Úvod do teórie kódovania

Daniel Olejár
Martin Stanek

22. mája 2011
Verzia 2.0

Obsah

Kapitola 1

Úvod

> The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point.
>
> (Claude Shannon, 1948)

Komunikácia (výmena informácie medzi dvoma alebo viacerými zúčastnenými stranami) je nutným predpokladom existencie a vývoja akéhokoľvek systému, ktorého časti musia navzájom spolupracovať pri plnení spoločných úloh. Aby bola možná koordinácia spolupracujúcich častí, komunikácia, ktorá medzi nimi prebieha, musí byť efektívna a spoľahlivá. Efektívnosť komunikácie znamená, že výmena informácie musí prebehnúť vo vymedzenom čase (nesmie trvať príliš dlho) a s vynaložením ohraničeného úsilia (napríklad finančných zdrojov, energie). Aby sme mohli komunikáciu považovať za spoľahlivú, správa sa počas prenosu nesmie modifikovať tak, aby ju príjemca nedokázal správne interpretovať; t.j. príjemca musí dokázať na základe prijatej správy dostatočne presne zrekonštruovať odvysielanú správu. Okrem toho sa na komunikáciu často kladie aj tretia požiadavka—bezpečnosť.[1] Bezpečnosť komunikácie je predmetom záujmu disciplíny nazývanej informačná bezpečnosť a v tejto práci sa ňou nebudeme zaoberať.

Požiadavky na efektívnosť a spoľahlivosť komunikácie sú už na prvý pohľad v priamom protiklade (náklady na prenos informácie a výkon vysielača) a tak býva často problém nájsť pre komunikačný systém nejaké uspokojivé riešenie. Ešte v nedávnej minulosti sa kompromis medzi efektívnosťou a spoľahlivosťou komunikácie riešil (a celkom úspešne) empiricky. V roku 1948 vyšiel článok Claude Shannona *A Mathematical Theory of Communication*, ktorý položil základy dvoch matematických teórií—teórie informácie a teórie kódovania. Tieto teórie pomohli exaktne sformulovať známe problémy efektívnosti a spoľahlivosti komunikácie a vytvoriť rámec pre ich riešenie. Teória informácie skúma ohraničenia na prenos informácie a teória kódovania sa usiluje nájsť

[1] Bezpečnosť komunikácie má viacero aspektov, z ktorých najdôležitejšie sú dôvernosť, integrita a autentickosť prenášanej informácie. Bezpečnosť komunikácie sa zaisťuje pomocou rozličných prostriedkov (kryptologických, technických, organizačných, právnych a pod.).

taký spôsob zápisu (kódovanie) informácie, ktorý by umožnil dosiahnuť hranice stanovené teóriou informácie. Vznik a rozvoj teórie informácie a teórie kódovania bol motivovaný potrebou riešenia praktických problémov komunikácie[2]. Napriek tomu, že sa obe teórie vyvíjali ako matematické teórie (teória informácie využíva najmä analytické a pravdepodobnostné metódy, kým teória kódovania používa algebraické metódy), neodtrhli sa od problémov reálnej komunikácie. Najmä komunikácia s kozmickými sondami prinášala trvalé podnety pre hľadanie efektívnych samoopravných kódov. V súčasnosti zažíva aplikačný boom najmä teória kódovania. Postupujúca informatizácia spoločnosti prináša digitalizáciu informácie; elektronické dokumenty nahrádzajú papierové, digitalizuje sa telefonická komunikácia, rozhlasové a televízne vysielanie, analógový zápis zvuku, obrazu a videa je rýchle nahradzovaný digitálnym. Rôzne aplikácie využívajúce digitálne zapísanú informáciu kladú čoraz náročnejšie požiadavky na spoľahlivosť a efektívnosť komunikácie. (Napríklad digitálny zápis filmov na DVD si vyžaduje zápis veľkého množstva informácie vo forme, ktorá umožní on-line dekódovanie a je zároveň dostatočne odolná voči chybám.) Používatelia videa, MP3 prehrávačov, mobilných telefónov, ani laickí používatelia informačných a komunikačných systémov nepotrebujú študovať teóriu kódovania, ani teóriu informácie na to, aby dokázali rozličné informačné a komunikačné zariadenia úspešne používať. V inej situácii sú informatici, ktorí budú pracovať s rozsiahlymi údajmi a budú potrebovať zvoliť na ich spracovanie čo najefektívnejšie metódy. Dokonca aj v prípade, keď budú mať k dispozícii hotové programy, budú potrebovať aspoň základné znalosti teórie informácie a teórie kódovania na to, aby dokázali posúdiť, či sú dané programy vhodné na spracovanie ich údajov a ak nie, či vôbec existujú efektívne metódy riešenia.

Od uverejnenia Shannonovho článku vyšlo množstvo dobrých kníh z teórie kódovania (a samozrejme aj teórie informácie). Viacero z nich je dostupných na Internete. Na Internete možno nájsť aj univerzitné prednášky, informácie o štandardoch, spôsoboch kódovania rozličných druhov informácie aj najnovšie vedecké výsledky teórie kódovania a teórie informácie. S výnimkou elementárnych učebných textov si však uvedené informácie od čitateľa vyžadujú aspoň základné poznatky, ktorých získanie samostatným štúdiom nemusí byť ani jednoduché, ani efektívne. V slovenskej odbornej literatúre bola teórii kódovania venovaná jedna kapitola v Jablonského knihe Úvod do diskrétnej matematiky z roku 1982 a Adámkova Teorie kódování, ktorá vyšla v roku 1988, ale ucelená učebnica alebo monografia z teórie kódovania chýba. Touto knihou chceme spomínanú medzeru v slovenskej odbornej literatúre zaplniť. Kniha je určená predovšetkým univerzitným študentom informatiky, informatikom, matematikom a všetkým, ktorí majú záujem o teóriu kódovania. U čitateľa predpokladáme aspoň základné znalosti z matematickej analýzy, algebry, lineárnej a teórie pravdepodobnosti v rozsahu úvodných kurzov magisterského, resp. inžinierskeho štúdia. Cieľom knihy je oboznámiť čitateľa so základnými problémami, ktoré teória kódovania rieši, metódami návrhu dobrých kódov, efektívnnymi metódami kódovania a dekódovania informácie, ako aj hranicami, ktoré pre konštrukciu kódov vyplývajú z teórie informácie. Preštudovanie tejto knihy by mu mohlo pomôcť využívať výsledky teórie kódovania na riešenie vlastných problémov súvisiacich s kódovaním informácie. Ak sme u čitateľa vzbudili záujem o samotnú teóriu kódovania, v závere knihy mu odporúčame literatúru pre ďalšie štúdium. Rozsah súčas-

[2]Claude Shannon a Richard Hamming, ktorý rozpracoval základy teórie kódovania, pracovali v tom čase v American Telephone and Telegraph's Bell Laboratories

ného poznania v teórii kódovania vyžaduje prijať rozhodnutie, ktorými časťami teórie kódovania sa v knihe zaoberať nebudeme. Rozhodli sme sa obetovať teoretickejšie časti, ktoré sa bezprostredne nedajú použiť na konštrukciu kódov, resp. na posudzovanie ich vlastností. Čitateľa, ktorému by tieto časti chýbali, odkazujeme na literatúru uvedenú v zozname, resp. odporúčania pre ďalšie štúdium uvedené v záverečnej kapitole.

Táto kniha vznikla na základe prednášok z teórie kódovania, ktoré sme na Fakulte matematiky, fyziky a informatiky Univerzity Komenského prednášali pre študentov informatiky od polovice 80-tych rokov. Pôvodne sme vychádzali z klasických prác [12], [3], resp. [6] a prednáška bola koncipovaná viac teoreticky ako aplikačne. Vzhľadom na zameranie poslucháčstva, medzi ktorým prevládali informatici (a časovým obmedzeniam) sa postupne ťažisko prednášky presunulo od matematickej teórie k algoritmom. Inšpiráciu sme našli v Blahutovej knihe [2], ktorý nielen našiel rozumný kompromis medzi nevyhnutnou, pomerne abstraktnou teóriou a efektívnymi algoritmami, ale dokázal túto náročnú problematiku podať veľmi prístupným spôsobom. Z tejto knihy sme intenzívne čerpali podnety pre prednášku aj pre túto knihu. Lintova kniha [15] nám poskytla informácie o aktuálnych teoretických výsledkoch, prehľadný dôkaz Shannonovej vety a zaujímavý pohľad na vzťah medzi technickými prostriedkami a samoopravnými kódmi pri zaistení spoľahlivosti komunikácie. Pozreli sme si množstvo prednášok z teórie kódovania na špičkových svetových univerzitách; za všetkých spomenieme najmä metodicky pekne spracované učebné texty J.I.Halla, [7] z Michigen State University a veľmi obsažné prednášky Mahdu Sudana z MIT. Veľmi inšpiratívne boli Shannonovské prednášky Roberta J. McEliecea z Caltechu o úlohe samoopravných kódov pri kozmickom výskume. Teória kódovania a teória informácie sa od svojho vzniku uberali vlastnými cestami. Hamming [3] ukázal na súvislosti výsledkov oboch teórií; v podobnom duchu, s aktuálnym obsahom a širším záberom je napísaná kniha [11]. Títo a ďalší kolegovia, matematici a inžinieri pracujúci v teórii kódovania prispeli k rozvoju nášho poznania hĺbky a krásy tejto teórie a významu jej aplikácií, za čo im patrí naša úprimná vďaka. Ďakujeme aj tvorcom programu Maple, vďaka ktorému sme mohli do knihy zaradiť viacero príkladov, ktorých vypracovanie presiahlo možnosti ručných výpočtov; tvorcom programov TEXa LATEX, pomocou ktorých sme mali možnosť upraviť podľa vlastných predstáv grafickú podobu tejto knihy.

Počas práci na knihe sa jej pôvodné zameranie menilo a postupne presiahlo aj rozsah základnej prednášky z teórie kódovania. Snažili sme sa preto o taký výklad problematiky, ktorý by umožnil použiť časti knihy ako učebné texty pre rozličné kurzy. Niektoré z nich uvádzame na nasledujúcich schémach

TO DO

Pracovné verzie knihy slúžili ako študijné texty pre túto prednášku. Vďaka tomu sme dostali množstvo pripomienok, upozornení na existujúce chyby i návrhov na doplnenie a prepracovanie niektorých častí. Za všetky pripomienky a námety, ktoré prispeli k zlepšeniu obsahu i formy prezentácie úprimne ďakujeme. Osobitne by sme chceli poďakovať Broni Brejovej a Tomášovi Vinařovi za príspevok k Reed Mullerovým kódom, Jánovi Mazákovi a Edite Rollovej za podrobné errata a Monike Steinovej za spracovanie príkladu o nebinárnych BCH kódoch.

V aktuálnej verzii 2.0. sú opravené tlačové a obsahové chyby, ktoré našla Edita Rol-

lová, v porovnaní s predchádzajúcou verziou je doplnená časť Lineárna algebra. Nie sú zatiaľ opravené chyby v príklade o ternárnom BCH kóde, ani pripomienky študentov k výkladu Berlekampovho-Masseyovho algoritmu.

Daniel Olejár a Martin Stanek

Poznámka. Poznámka. Tento text je pracovnou verziou knihy z teórie kódovania. Je určený ako študijný text pre poslucháčov informatiky na Univerzite Komenského v Bratislave. Akékoľvek iné použitie si vyžaduje písomný súhlas autorov.

Kapitola 2

Základné pojmy a označenia

Mnohé pojmy teórie kódovania sa stali súčasťou bežného jazyka a ľudia ich často používajú bez toho, aby si uvedomovali ich presný význam. V bežnej komunikácii to natoľko neprekáža, ale pri odbornom výklade by rozličná interpretácia základných pojmov mohla viesť k nedorozumeniam. Aby sme sa v ďalšom výklade vyhli zbytočným nedorozumeniam, vybudujeme exaktne potrebný pojmový aparát.

2.1 Abecedy, slová a jazyky

Abeceda je ľubovoľná konečná neprázdna množina. Prvky abecedy budeme nazývať *znakmi* alebo *symbolmi*. Abecedu budeme označovať symbolom Σ; ak bude potrebné rozlišovať rozličné abecedy, budeme symbol Σ indexovať ($\Sigma_1, \Sigma_2, \dots$). Ľubovoľná konečná postupnosť znakov z abecedy Σ sa nazýva *slovom nad abecedou* Σ. Ak nebude podstatné o akú abecedu ide alebo z kontextu bude známe, o ktorú abecedu sa jedná, budeme kvôli stručnosti slová nad abecedou Σ vynechávať. Zjednodušíme aj zapisovanie slov; symboly v postupnosti nebudeme oddeľovať čiarkami a slovo (napr.) a, b, e, c, e, d, a budeme zapisovať v tvare, ako sa slová v textoch štandardne zapisujú; t.j. $abeceda$. Nech je w slovo nad abecedou Σ, potom počet znakov slova w nazveme *dĺžkou slova* w. Dĺžku slova w budeme označovať symbolom $l(w)$. Tak napríklad $l(slovo) = 5$, $l(abeceda) = 7$, $l(a) = 1$. Postupnosť znakov nad abecedou Σ môže byť aj prázdna. Takáto postupnosť sa nazýva *prázdnym slovom* a označuje sa symbolom ε. Pre dĺžku prázdneho slova platí $l(\varepsilon) = 0$. Teraz definujeme operácie nad slovami, pomocou ktorých bude možné zo známych slov vytvárať nové slová. Nech sú u, v dve slová nad abecedou Σ; $u = a_1 \dots a_n; \quad v = b_1 \dots b_m$. *Zreťazením slov* u, v je slovo $w = uv = a_1 \dots a_n b_1 \dots b_m$ nad abecedou Σ. (Je zrejmé, že operácia zreťazovania slov je asociatívna, ale vo všeobecnosti nie je komutatívna; prázdne slovo ε je obojstranným neutrálnym prvkom vzhľadom na operáciu zreťazenia slov: pre ľubovoľné slovo w platí $w\varepsilon = \varepsilon w = w$). Slovo možno zreťaziť aj so sebou samým, napr. $uu = a_1 \dots a_n a_1 \dots a_n$. Pre ľubovoľné slovo w a ľubovoľné číslo $k \in \mathcal{N}$ definujeme:

1. $w^0 = \varepsilon$,
2. $w^{k+1} = w^k w$.

Nech $u = a_1 \dots a_n$ je ľubovoľné slovo, potom súvislú podpostupnosť $z = a_i a_{i+1} \dots a_{i+k-1}$; $1 \leq i, i+k < n$ nazveme *podslovom slova* u. Ak $0 < k < n$, slovo z nazveme *vlastným podslovom slova* u. Slová u a ε sú triviálnymi podslovami slova u a v kódovaní sa nimi zvlášť zaoberať nebudeme. Zato však v teórii kódovania zohrávajú dôležitú úlohu podslová, ktoré sú začiatkom alebo koncom nejakého slova. Zavedieme pre ne špeciálne pomenovania. Nech $u = a_1 \dots a_n$ je ľubovoľné slovo, slovo $z = a_1 \dots a_k, \quad 0 < k \leq n$ nazveme *počiatočným podslovom (prefixom)* slova u a slovo $x = a_i a_{i+1} \dots a_n; \quad 1 \leq i = n$ nazveme *koncovým podslovom (sufixom) slova* u. Znaky v slove možno aj preusporiadať. Dôležitým prípadom preusporiadania znakov je otočenie slova: *zrkadlovým obrazom* slova $u = a_1 \dots a_n$ nazveme slovo $u^R = a_n \dots a_1$.

Slová môžeme zoskupovať do množín. Takéto množiny slov budeme nazývať jazykmi. Presnejšie, ľubovoľnú množinu slov nad abecedou Σ nazveme *jazykom nad abecedou* Σ. Keďže jazyky sú množiny slov, možno z existujúcich jazykov vytvárať nové jazyky pomocou množinových operácií, ako sú zjednotenie, doplnok, rozdiel, prienik, symetrická diferencia množín a prípadne iné. Pre slová sme zaviedli operáciu zreťazovania (slov). Zavedieme teraz užitočné operácie s jazykmi, ktoré sú založené na zreťazovaní slov. Nech sú $\mathcal{L}_1, \mathcal{L}_2$ jazyky nad abecedou Σ, potom $\mathcal{L} = \mathcal{L}_1\mathcal{L}_2$ je jazyk nad abecedou Σ definovaný nasledovne: $\mathcal{L} = \{uv;\ u \in \mathcal{L}_1, v \in \mathcal{L}_2\}$. Jazyk možno zreťazovať so sebou samým; pre ľubovoľný jazyk $\mathcal{L}$ a ľubovoľné číslo $k \in \mathcal{N}$ definujeme:

1. $\mathcal{L}^0 = \{\varepsilon\}$,
2. $\mathcal{L}^{k+1} = \mathcal{L}^k\mathcal{L}$.

Na záver uvedieme ešte dve operácie nad jazykmi, ktoré nám umožnia popísať množinu všetkých možných slov, ktoré sa dajú vytvoriť pomocou operácie zreťazovania jazyka. Nech $\mathcal{L}$ je ľubovoľný jazyk, potom jazyky $\mathcal{L}^+ = \bigcup_{i>0}^{\infty} \mathcal{L}^i$ a $\mathcal{L}^* = \bigcup_{i\geq 0}^{\infty} \mathcal{L}^i$ sa nazývajú *kladná,* resp. *nezáporná iterácia jazyka* $\mathcal{L}^i$. Všimnite si, že abecedu Σ možno chápať aj ako jazyk pozostávajúci zo všetkých slov dĺžky 1 nad abecedou Σ a Σ^* predstavuje množinu všetkých slov nad abecedou Σ.

Ilustrujeme zavedené pojmy na príkladoch.

Príklad.

1. Binárna abeceda Σ_1 je ľubovoľná dvojprvková množina. Znaky binárnej abecedy najčastejšie označujeme číslicami $0, 1$; binárnu abecedu budeme v tomto prípade chápať ako množinu $\Sigma_1 = \{0, 1\}$.
2. Na zápis prirodzených čísel vystačíme s abecedou $0, 1$; $\Sigma_2 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.
3. Racionálne čísla možno zapísať v podobe slov nad abecedou $\Sigma_3 = \Sigma_2 \cup \{"+", "-", "."\}$.
4. $\Sigma_4 = \{a, b, c, d, e, f, g, i, j, k, l, m, n, o, p, q, r, s, t, v, w, x, y, z\}$ je abeceda pozostávajúca z malých písmen anglickej abecedy.

5. Abecedu $\Sigma_5 = \{A, B, C, D, E, F, G, I, J, K, L, M, N, O, P, Q, R, S, T, V, W, X, Y, Z\}$ tvoria veľké písmená anglickej abecedy.

6. $\Sigma_6 = \Sigma_4 \bigcup \Sigma_5$.

7. Abecedu $\Sigma_7 = \{\alpha, \beta, \gamma, \delta, \epsilon, \varepsilon, \zeta, \eta, \theta, \vartheta, \iota, \kappa, \lambda, \mu, \nu, \xi, o, \pi, \varpi, \rho, \varrho, \sigma, \varsigma, \tau, \upsilon, \phi, \varphi, \chi, \psi, \omega\}$ tvoria malé písmená gréckej abecedy.

8. Ďalšími užitočnými abecedami by mohli byť rozličné znakové sady, napr. všetky znaky kódov ASCII. V teórii kódovania budeme často pracovať s abecedami, ktorých symboly sú prvkami konečných polí. Tieto symboly budeme zapisovať pomocou prirodzených čísel; $\Sigma_8 = Z_m = \{0, 1, \ldots m-1\}$.

9. Slovo 2.78128 je slovom nad Σ_3, ale nie je slovom nad abecedou Σ_2 (pretože obsahuje symbol ".", ktorý sa v abecede Σ_2 nenachádza).

10. Zreťazením slov $w_1 = pismeno$ a $w_2 = male$ dostávame slová (napr. nad abecedou Σ_4) $w_1w_2 = pismenomale$ a $w_2w_1 = malepismeno$.

11. Nech je dané slovo $w_1 = pismeno$ nad abecedou Σ_4, počiatočné a koncové podslová tohto slova sú uvedené v nasledujúcej tabuľke:

prefix	sufix
ε	pismeno
p	ismeno
pi	smeno
pis	meno
pism	eno
pisme	no
pismen	o
pismeno	ε

12. Nech je dané slovo $w_1 = pismeno$ nad abecedou Σ_4, zrkadlový obraz slova w_1 je slovo $w_1^R = onemsip$ nad abecedou Σ_4.

13. Nech sú $\mathcal{L}_1 = \{ne, pre, po, vy\}$, $\mathcal{L}_2 = \{mysli, hovor, pis, padni\}$ jazyky nad abecedou Σ_4. Jazyk $\mathcal{L}_1\mathcal{L}_2$ je uvedený v nasledujúcej tabuľke:

$\mathcal{L}_1/\mathcal{L}_2$	ne	pre	po	vy
mysli	nemysli	premysli	pomysli	vymysli
hovor	nehovor	prehovor	pohovor	vyhovor
pis	nepis	prepis	popis	vypis
padni	napadni	prepadni	popadni	vypadni

14. Uvažujme binárnu abecedu $\Sigma_1 = \{0, 1\}$. Uvedieme množiny slov Σ_1^k pre niekoľko

počiatočných hodnôt k:

k	Σ_1^k
0	$\{\varepsilon\}$
1	$\{0, 1\}$
2	$\{00, 01, 10, 11\}$
3	$\{000, 001, 010, 011, 100, 101, 110, 111\}$
4	$\{0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111\}$
5	$\{00000, 00001, 00010, 00011, 00100, 00101, 00110, 00111, 01000, 01001, 01010, 01011, 01100, 01101, 01110, 01111, 10000, 10001, 10010, 10011, 10100, 10101, 10110, 10111, 11000, 11001, 11010, 11011, 11100, 11101, 11110, 11111\}$

2.2 Údaje, informácia a komunikácia

V bežnom živote sa pojem informácie používa voľne a v rozličných významoch; hovorí sa o rozličných druhoch informácie (obrazová, knižná, zvuková, genetická, novinová) a informácii sa pripisujú rozličné atribúty (overená, čerstvá, aktuálna, pochybná, škandalózna a i.) V teórii kódovania nás nebude zaujímať pôvod, význam ani hodnotenie informácie; jediné čo pre nás bude podstatné je množstvo informácie. Budeme pracovať s údajmi a správami, ktoré budú obsahovať nejakú informáciu, tieto údaje budeme spracovávať a budeme sa snažiť nájsť pre zápis informácie, ktorú údaje obsahujú formu ktorá je z hľadiska spracovania údajov/informácie najvhodnejšia.

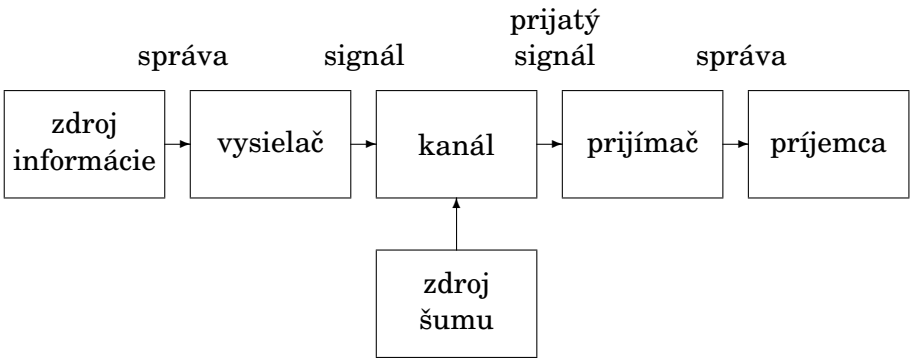
V hovorovom jazyku sa pojmy informácia, správa a údaje chápu ako synonymá; ukážeme, že tieto pojmy majú odlišný význam. Ilustrujeme rozdiel medzi pojmami údaje a informácia na príklade. Predstavme si

1. binárny reťazec 0000000000001010,
2. výraz $0^{12}(01)^2$,
3. slovné spojenie „dvanásť núl, jednotka, nula, jednotka, nula“,
4. 10,
5. A,
6. 000A.

Vo všetkých prípadoch ide o jednoznačné určenie tej istej binárnej postupnosti dĺžky 16; v prvom prípade je popisom explicitné vymenovanie členov postupnosti, v druhom jej zápis pomocou regulárneho výrazu, v treťom slovný popis vo štvrtom vyjadrenie číselnej hodnoty binárneho čísla v desiatkovej sústave (predpokladáme, že informácia o dĺžke slova je známa), v piatom ide o hexadecimálny zápis toho istého čísla (s vynechaním počiatočných núl) a napokon posledný výraz je hexadecimálny zápis binárneho reťazca, vrátane prvých troch nulových hexadecimálnych číslic. Všetky popisy majú spoločné to, že umožňujú v množine všetkých binárnych reťazcov (v našom prípade dĺžky

16) jednoznačne identifikovať daný reťazec; t.j. obsahujú rovnakú informáciu. *Údaje* teda predstavujú záznam informácie; *informácia* je obsahom údajov. Niekedy sa pojem informácia spája aj so sémantikou (významom) údajov, ale toto spojenie chápanie informácie komplikuje, pretože do pojmu informácia zavádza subjektívny aspekt (toho, kto údaje interpretuje a kontext). Preto sa budeme pridržiavať vyššie uvedeného chápania informácie ako obsahu údajov.[1] Pojem *správa* sa zvykne používať na označenie údajov, ktoré majú istý formát a sú prenášané z jedného miesta na druhé. Okrem prenosu údajov v priestore sa údaje často prenášajú aj v čase: zapíšu sa na nejaké médium a po čase sa z neho čítajú. Pod *komunikáciou* budeme rozumieť činnosť dvoch alebo viacerých entít (účastníkov komunikácie), ktorá pozostáva z prenosu údajov/správ od jedného účastníka (odosielateľa) k druhému/iným (príjemca/príjemcovia). Existuje mnoho spôsobov komunikácie, ktoré závisia tak od použitých komunikačných prostriedkov (pošta, telefón, telegraf, televízia, rozhlas, a i.), typu údajov, ktoré sa pri komunikácii prenášajú aj účelu komunikácie. Nebudeme ich rozoberať, namiesto toho zavedieme pomerne všeobecný model komunikačného systému, popíšeme úlohu jeho jednotlivých subsystémov a ukážeme, aké úlohy sa musia pri komunikácii riešiť. Uvedený model použijeme tak na popis prenosu údajov v priestore, ako aj v čase.

Na obrázku 2.1 je uvedený Shannonov model komunikačného systému. Hoci je tento model veľmi všeobecný, hodí sa na popis mnohých komunikačných systémov a pre ďalšie (napríklad komunikačný systém so spätnou väzbou) môže Shannonov model poslúžiť ako základ, ktorý sa dá vhodne upraviť. Podrobnejší model komunikačného systému, odvodený zo Shannonovho modelu je uvedený na obrázku 2.2



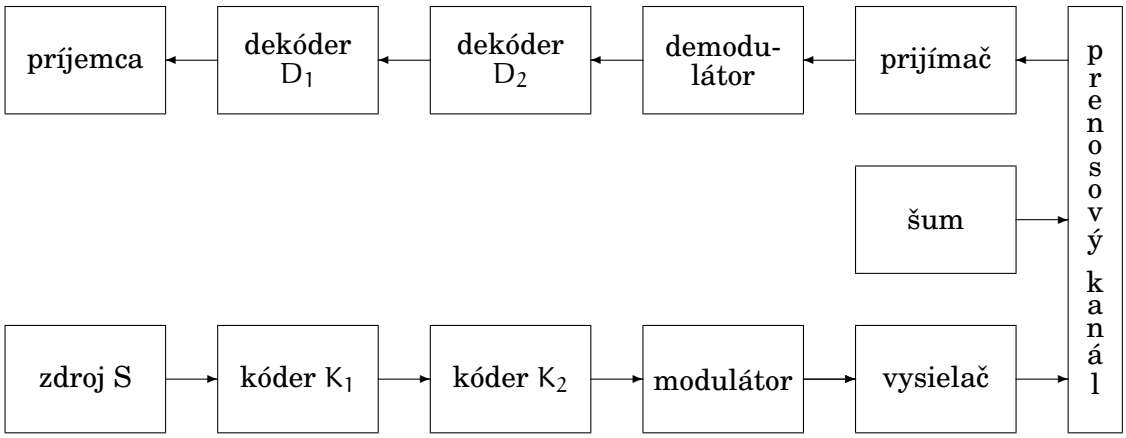
Obr. 2.1: Shannonov model komunikačného systému

Zdroj informácie/údajov. Aby sme sa nemuseli zaoberať tým, odkiaľ údaje (informácia) pochádzajú, budeme predpokladať, že existuje nejaký *zdroj informácie* (údajov), S (Source). Zdroju prislúcha nejaká abeceda, Σ_S, ktorú budeme nazývať *zdrojovou abecedou*, alebo *abecedou zdroja*. Ďalej Σ_S budeme predpokladať, že zdroj S generuje postupnosť znakov $x_i, x_{i+1}, \ldots; x_{i+j} \in \Sigma_S$; napríklad tak že v diskrétnych časových okamihoch (taktoch) sa na výstupe zdroja budú objavovať symboly zo zdrojovej abecedy. Postupnosť znakov zdrojovej abecedy môžeme spracovávať po znakoch, alebo rozdeliť na slová konečnej dĺžky. Bez ujmy na všeobecnosti môžeme predpokladať, že údaje, ktoré budeme

[1] Základné poznatky o meraní množstva informácie v údajoch sú uvedené v kapitole 16

spracovávať, majú formu postupnosti slov[2] nad abecedou Σ_S.

Kóder K_1 – kódovanie zdrojových údajov. Forma, v ktorej sú zapísané zdrojové údaje, nemusí byť vhodná pre ďašie spracovanie, a preto je postupnosť slov vytvorená zdrojom pred ďalším spracovaním zakódovaná. Toto kódovanie sa nazýva *kódovanie zdrojovej informácie*, alebo *kódovanie zdroja* a realizuje ho kóder K_1. Výsledkom kódovania zdrojovej informácie je postupnosť symbolov kódovej abecedy Σ_C, ktorú budeme nazývať *správou*. Keďže pôvodnú informáciu získavame priamo zo zdroja, kódovanie zdroja nemusí riešiť ochranu údajov pred prípadnými chybami, ale plní inú úlohu—zaisťuje dosiahnutie efektívnosti zápisu zdrojovej informácie. Výsledkom kódovania zdrojových údajov je (v ideálnom prípade) najkratšia správa nad kódovou abecedou, na základe ktorej možno v plnom rozsahu zrekonštruovať zdrojové údaje v pôvodnej podobe. Požiadavka na efektívnosť kódovania správy sa dá vyjadriť tak, že vo výslednej (kódovanej) správe sa ľubovoľná k-tica znakov kódovej abecedy bude vyskytovať s rovnakou pravdepodobnosťou[3].



Obr. 2.2: Zovšeobecnený model komunikačného systému

Na tomto mieste sa na chvíľu zastavíme. Shannonov model komunikačného systému predpokladá, že v ideálnom prípade sa príjemcovi podarí zrekonštruovať správu v pôvodnom tvare. Aj keď sa v reálnych systémoch používa kódovanie zdroja, ktoré realizuje tzv. *bezstratovú kompresiu (data compaction)*, údaje generované zdrojom častokrát obsahujú informáciu, ktorú príjemca nedokáže využiť. Bezstratová kompresia takýchto údajov by viedla ku správam, ktoré by boli zbytočne rozsiahle. Ak dokážeme určiť, ktorá informácia obsiahnutá v zdrojových údajoch je podstatná a ktorá nie, môžeme na kódovanie zdrojových údajov použiť efektívnejšie kódovanie, založené na odfiltrovaní tak redundancie, ako aj nepodstatnej informácie obsiahnutej v zdrojových údajoch. Takéto kódovanie zdroja, pri ktorom dochádza k istej strate informácie sa nazýva *(kompresia so stratou informácie, data compression)*. Príkladom môže byť kódovanie hudby, ktoré využíva skutočnosť, že údaje obsahujú informáciu ktorú príjemca nedokáže využiť (nepočuteľné zvuky); táto informácia sa pri kódovaní zdroja jednoducho odfiltruje a tým

[2] v krajnom prípade slov dĺžky 1, teda znakov zdrojovej abecedy

[3] Zmyslom kódovania zdroja je odstrániť *redundanciu* (nadbytočnosť) pôvodného zápisu. Požiadavka na rovnakú pravdepodobnosť výskytu všetkých k-tic kódovej abecedy znamená, že v kódovanej správe už nebude možné objaviť nejakú zákonitosť, ktorá by sa dala využiť na ďalšie zefektívnenie zápisu.

zvýši efektívnosť zápisu (porovnajte nejakú hudobnú skladbu zapísanú na audio CD a zápis tej istej skladby vo formáte MP3, príp. iných). Na druhej strane mechanické použitie kompresie so stratou informácie nebude asi použiteľné pri spracovávaní exe súborov (hoci inteligentná revízia zdrojových textov tých istých programov by nepochybne odhalila možnosti optimalizácie textu.)

Správa Kódovanú správu rozdelíme na bloky vhodnej dĺžky k. (O výbere k budeme hovoriť neskôr.) Pripomíname, že ak bol kóder K_1 dostatočne kvalitný a kódovaná správa dostatočne dlhá, všetky slová dĺžky k by sa v nej mali vyskytovať s rovnakými pravdepodobnosťami.

Kóder K_2 Na rozdiel od kódovania zdroja, kde nebolo treba rátať so šumom a úlohou kódera K_1 bolo redukovať redundanciu zdrojových údajov, správu budeme čoskoro posielať cez komunikačný kanál, na ktorý pôsobí šum. Úlohou druhého kódera je transformovať slová dĺžky k nad kódovou abecedou Σ_C na slová dĺžky n (kvôli jednoduchosti predpokladajme, že nad tou istou kódovou abecedou Σ_C) tak, aby sa len mierne zvýšila redundancia a príjemca bol schopný odhaliť/opraviť chyby, ktoré vzniknú pri prenose prenosovým kanálom. Najprv budeme uvažovať kóder bez pamäte. Tento kóder realizuje injektívne zobrazenie

$$\mathrm{ENC} : \Sigma_C^k \to \Sigma_C^n.$$

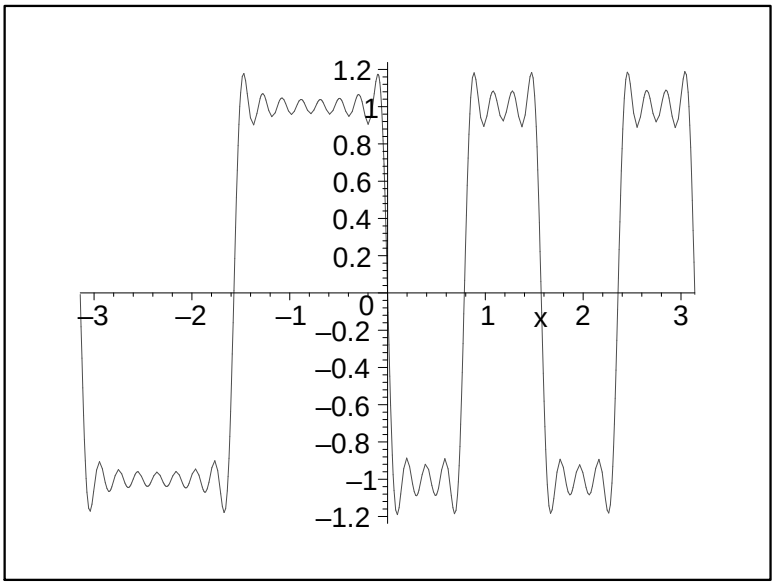
Kódery bez pamäte sa používajú na kódovanie pomocou blokových kódov a vyznačujú sa tým, že nezohľadňujú žiadne vzťahy medzi k-ticami vstupných údajov; to isté slovo (dĺžky k) sa zakaždým zobrazí na to isté slovo (dĺžky n). Existujú aj kódery s pamäťou, ktoré pri kódovaní znaku (zväčša kódujú znak po znaku) zohľadňujú aj predchádzajúce symboly. Tieto kódery sa používajú pri tzv. *konvolučných kódoch*.

Modulátor Správy sa prenášajú z jedného miesta na druhé pomocou fyzikálnych veličín, ktoré sa dokážu šíriť cez vhodné prostredie. Fyzikálna reprezentácia správy sa nazýva signál. (My budeme pomocou jedného signálu reprezentovať menšie časti správy, napríklad slová, alebo znaky kódovej abecedy.) Zariadenie, ktoré transformuje fyzikálnu veličinu tak, aby predstavovala príslušný signál, sa nazýva *modulátor*. Predstavme si napríklad rádiovú vlnu so sínusovým priebehom a amplitúdou 1 a binárnu kódovú abecedu $\Sigma = \{0, 1\}$. Symbolu 0 priradíme hodnotu -1 a symbolu 1 hodnotu $+1$. Postupnosť $0, 0, 1, 1, 0, 1, 0, 1$ bude reprezentovaná signálom, ktorého priebeh je uvedený na odrázku 2.3. Pre zaujímavosť uvedieme aj hodnoty signálov reprezentujúcich jednotlivé bity:

$$\begin{array}{cccc} -0.9479054106 & -0.9450567393 & 0.9450567393 & 0.9479054110 \\ -0.9180252682 & 0.9222918778 & -0.9222918783 & 0.9180252668 \end{array}$$

Vysielač je ďalším prvkom komunikačného systému. Jeho úlohou je generovať signály dostatočne silné na to, aby prekonali cestu k príjemcovi.

Prenosový kanál Signály sa môžu šíriť v rôznorodých prostrediach; napríklad kozmickým priestorom, po kovovom kábli, optickom vlákne a pod. Médium umožňujúce prenos signálov budeme nazývať *prenosovým kanálom*. Predpokladáme, že prenosový kanál je vystavený vplyvom okolitého prostredia, ktoré ovplyvňujú správy prenášané kanálom. Faktorov, ktoré môžu pôsobiť na prenosový kanál je tak veľa, že sa dosť dobre nedá



Obr. 2.3: Signál

skúmať vplyv jednotlivých faktorov, ale namiesto toho sa skúmajú dôsledky ich spoločného pôsobenia. Rôzne rušivé faktory vplývajúce na prenosový kanál, budeme nazývať zdrojmi šumu a výsledok ich pôsobenia—šumom . Budeme predpokladať, že šum má podobu signálov, ktoré ovplyvňujú signály prenášajúce správu (napríklad sa s nimi skladajú), v dôsledku čoho dôchádza k zmenám signálov, ktoré sa v prenášanej správe prejavujú ako chyby troch základných typov:

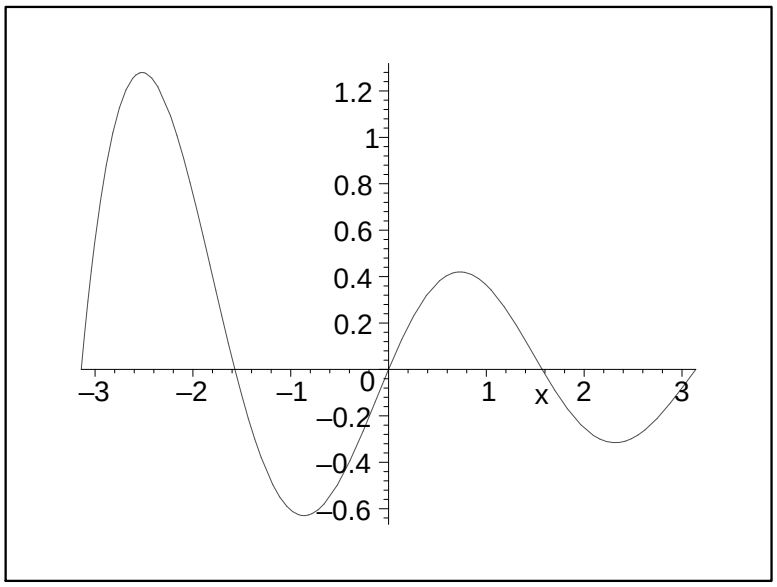
1. nahradenie jedného symbolu prenášanej správy iným symbolom (kódovej abecedy);
2. zmazaním symbolu (čo môžeme chápať tak, že symbol prenášanej správy je nahradený symbolom, ktorý nepatrí do kódovej abecedy);
3. výpadkom/doplnením nového symbolu (kódovej abecedy) do prenášanej správy (porucha synchronizácie).

Šumový signál je zobrazený na obrázku 2.4

V tejto knihe sa budeme zaoberať kódmi, ktoré umožnia riešiť chyby prvého a druhého druhu; t.j. odhaľovať a opravovať chyby. Poruchami synchronizácie sa nebudeme zaoberať, čitateľovi odporúčame

TO DO

Signály prenášané prenosovým kanálom zachytáva prijímacia strana pomocou **prijímača** (napríklad anténa mobilného telefónu). Abstrahujeme od transformácií signálov, ktoré realizuje prijímač a predpokladáme, že prijaté signály vstupujú do **demodulátora**, ktorý transformuje signály na postupnosť znakov kódovej abecedy. Demodulátor už v podstate robí prvú korekciu chýb. V dôsledku pôsobenia šumu na kanál prijaté



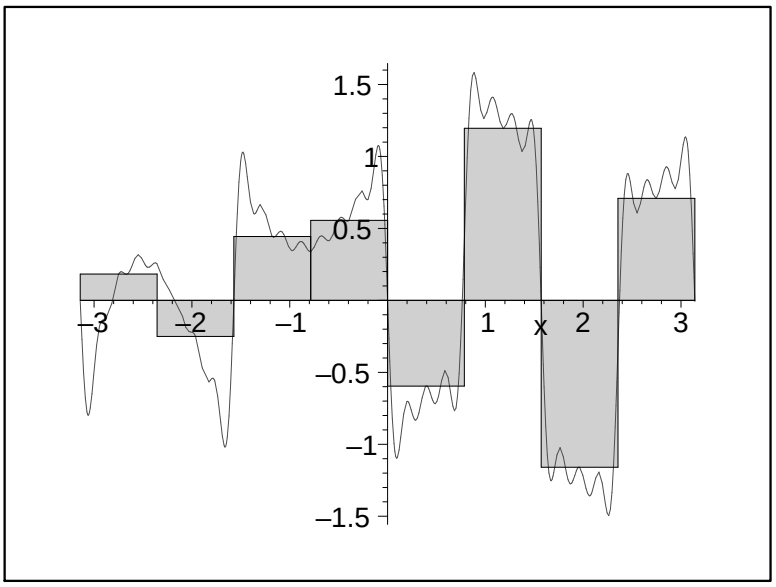
Obr. 2.4: Šum

signály nebudú mať zďaleka ideálny priebeh, obr. 2.5. Postupnosť $0, 0, 1, 1, 0, 1, 0, 1$ je reprezentovaná postupnosťou hodnôt[4]

bit	pôvodný signál	prijatý signál	interpretácia	
			tvrdá	mäkká
0	−0.9479054106	0.1824649004	1	?
0	−0.9450567393	−0.2506249324	0	?
1	0.9450567393	0.4439007163	1	1
1	0.9479054110	0.5558633849	1	1
0	−0.9180252682	−0.5960788882	0	0
1	0.9222918778	1.195406403	1	1
0	−0.9222918783	−1.159436952	0	0
1	0.9180252668	0.7084777992	1	1

Ani jedna z prijatých hodnôt (signálov) nepatrí do množiny $\{-1, 1\}$. Aby dekóder mohol transformovať prijaté signály na znaky kódovej abecedy, musí použiť pružnejšie pravidlo; napríklad, signály s nezápornými hodnotami budú reprezentovať 1 a ostatné signály budú reprezentovať kódový znak 0. Pri použití tohto pravidla sa značne deformované signály transformujú na postupnosť $1, 0, 1, 1, 0, 1, 0, 1$. Demodulátor môže byť navrhnutý tak, aby zakaždým prijal rozhodnutie o interpretácii signálu (*hard quantization*). To by však mohlo viesť k nesprávnej interpretácii signálov blízkych k 0 demodulátorom a problém (identifikáciu a opravenie chyby) by musel riešiť dekóder. Napriek tomu, že pri "tvrdej"transformácii spojitého signálu na diskrétne hodnoty sa dosahuje najvyššia pravdepodobnosť správneho priradenia, často sa používa alternatívne riešenia, tzv. *soft quantization*. V krajnom prípade demodulátor neinterpretuje žiadne signály ako

[4]tieto predstavujú hodnoty signálu v strede príslušných intervalov.



Obr. 2.5: Prijatý signál

kódové znaky a posunie dekóderu vypočítané číselné hodnoty signálov na jednotlivých časových intervaloch. Tým sa prakticky celé spracovanie prijatej správy presunie na dekóder, čo však zvyšuje nároky na jeho výkonnosť (zložitosť a zrejme aj cenu). Rozumné je preto kompromisné riešenie, kedy demodulátor predspracuje prijaté signály napríklad tak, že jednoznačne interpretuje tie signály, ktorých hodnoty dostatočne dobre zodpovedajú hodnotám reprezentujúcim jednotlivé znaky kódovej abecedy, problematické hodnoty signálu bude reprezentovať nejakým novým symbolom a tieto výsledky odovzdá dekóderu. Presnejšie, nech s_k označuje priemernú hodnotu signálu v takte k, a a_k je hodnota symbolu kódovej abecedy zodpovedajúca signálu s_k. Potom pravidlo pre binárnu kódovú abecedu by mohlo vyzerať napríklad takto

$$a_k = \begin{cases} 1 & s_k \geq 0.3, \\ 0 & s_k \leq -0.3, \\ ? & -0.3 < s_k < 0.3. \end{cases}$$

V našom prípade by demodulátor postupnosť prijatých signálov interpretoval ako postupnosť $?, ?, 1, 1, 0, 1, 0, 1$. Ak počas prenosu došlo ku chybe (nahradenie jedného znaku kódového slova iným), informácia, ktorú dostal dekóder od demodulátora by mu umožnila odhadnúť najpravdepodobnejšie miesta, na ktorých mohlo dôjsť ku chybe (v našom príklade sú podozrivé prvé dva symboly), čím by sa (ako uvidíme neskôr) značne zjednodušilo dekódovanie.

Dekódery D_2 a D_1 Hlavnou úlohou dekódera D_2 je čo najlepšie rekonštruovať odvysielanú správu. Predpokladajme, že poznáme všetky kódové slová (budeme ich označovať ako $\mathbf{u}_i$), všetky možné prijaté slová $\mathbf{v}_j$ a podmienené pravdepodobnosti $p_{i,j} = p(\mathbf{v}_j|\mathbf{u}_i)$. Pravdepodobnosť $p_{i,j}$ vyjadruje pravdepodobnosť toho, že po odvysielaní kódového slova $\mathbf{u}_i$ bolo prijaté (nejaké) slovo $\mathbf{v}_j$. Základom pre rozhodovanie dekódera D_2 je nasledujúci

princíp *dekódovania na základe maximálnej pravdepodobnosti* (*Maximum Likelihood Decoding, MLD*):

> Prijaté slovo $\mathbf{v}_j$ dekódujeme na také kódové slovo $\mathbf{u}_i$, pre ktoré je podmienená pravdepodobnosť $p(\mathbf{v}_j|\mathbf{u}_i)$ maximálna.

Dekódovanie na základe maximálnej pravdepodobnosti vždy dáva výsledok (kódové slovo). Takéto dekódovanie sa nazýva *úplné dekódovanie* . Ako sa dá očakávať, okrem úplného dekódovania bude existovať aj nejaké alternatívne riešenie, ktoré budeme nazývať *neúplným dekódovaním*. Pri neúplnom dekódovaní môžu nastať dva prípady:

1. dekóder dekóduje prijaté slovo t.j. priradí prijatému slovu kódové slovo,
2. dekóder namiesto kódového slova vypíše nejaký dohodnutý symbol (napríklad ∞).

Druhý prípad nastane vtedy, keď dekóder našiel v prijatom slove chybu, ale nebol ju schopný opraviť. Takúto situáciu nemôžeme vylúčiť, pretože dekóder nie je schopný opraviť slová, ktoré boli výrazne modifikované. V takom prípade je lepšie požiadať o opätovné zaslanie informácie, ako sa pokúšať opraviť prijaté slovo a dekódovať ho nesprávne. Ani takýto prístup však nezaručuje, že dekódované slovo sa zhoduje s odvysielaným kódovým slovom. Keďže dekóder rozhoduje na základe syntaxe (napr. zoznamu kódových slov) a nie sémantiky prijatých správ, ak počas prenosu kódového slova nastala chyba, ktorá ho transformovala na iné **kódové slovo**, dekóder takúto chybu nedokáže identifikovať. Preto dekóder postavený na princípe MLD bude mať síce najvyššiu pravdepodobnosť správneho dekódovania, ale ak sa pomýli, tak je dekódovaná správa zaťažená chybou, ktorú je ťažko odhaliť. Preto väčšina dekóderov, ktorými sa budeme zaoberať, vychádza z trocha slabšieho princípu, nazývaného *IMLD* (*Incomplete Maximum Likelihood Decoding*); *neúplné dekódovanie na základe maximálnej pravdepodobnosti:*

> Prijaté slovo $\mathbf{v}_j$ dekódujeme buď na také kódové slovo $\mathbf{u}_i$, pre ktoré je podmienená pravdepodobnosť $p(\mathbf{v}_j|\mathbf{u}_i)$ maximálna, alebo na symbol ∞; bola odhalená chyba.

Napriek použitiu samoopravných kódov nedokážeme garantovať správne dekódovanie prijatej správy. Budeme rozlišovať dva problematické prípady: *chyba dekódera* (*decoder error*) nastáva vtedy, keď dekóder nesprávne dekódoval prijaté slovo; t.j. interpretoval ho ako iné kódové slovo, ako bolo to, ktoré bolo odvysielané. *Zlyhanie dekódera* zahŕňa tak chybu dekódera, ako aj ten prípad, keď dekóder odhalil chybu ale nebol ju schopný opraviť.

Dekóder D_1 transformuje dekódovanú správu do podoby, v ktorej ju môže ďalej spracovávať príjemca. Dobrým príkladom je dekódovanie binárne zapísanej zvukovej informácie (hudby) do počúvateľnej podoby, alebo dekódovanie digitálne zapísaných filmov na DVD.

Všimneme si, že hoci sme v modeli prenosového kanála hovorili o prenose správ, tento model možno priamo použiť aj na popis uchovávania a opätovného čítania údajov. Zaznamenávanie údajov a ich opätovné čítanie možno chápať ako prenos informácie v čase, zatiaľ čo pri prenose správ sa jedná o prenos informácie v priestore. Jeden podstatný rozdiel medzi prenosom údajov v čase a priestore však je. Ak pri prenose informácie v priestore dôjde k odhaliteľnej ale neopraviteľnej chybe, príjemca má možnosť požiadať odosielateľa o opätovné zaslatie správy. Ak však dôjde k poškodeniu údajov zapísaných na nejakom pamäťovom médiu, opätovné čítanie neumožní prečítať správne údaje. O to dôležitejšie je pri uchovávaní údajov na pamäťových médiách ochrana ich integrity napríklad pomocou samoopravných kódov. Z hľadiska úloh, ktoré rieši teória kódovania nie je rozdiel informácie v čase a priestore podstatný, a preto sa v ďalšom sa sústredíme na problémy vznikajúcimi pri prenose informácie v priestore.

2.3 Kódovanie

Vráťme sa k modelu komunikačného systému z predchádzajúcej časti. Zostáva vyriešiť problém, ako zapisovať správy, ktoré generuje zdroj S v podobe postupnosti znakov nad abecedou Σ_S pomocou kódovej abecedy Σ_C. Existuje viacero riešení tohto problému. Začneme tým najjednoduchším—kódovaním jednotlivých znakov zdrojovej abecedy. Nech $\Sigma_S = \{s_0, \ldots, s_{m-1}\}$ je zdrojová abeceda a $\Sigma_C = \{b_0, \ldots, b_r\}$ je kódová abeceda a nech sú $\nu_0, \ldots, \nu_{m-1}$ navzájom rôzne slová nad kódovou abecedou Σ_C. Potom zobrazenie

$$\begin{aligned} s_0 &\rightarrow \nu_0 \\ s_1 &\rightarrow \nu_1 \\ &\vdots \\ s_{m-1} &\rightarrow \nu_{m-1} \end{aligned}$$

budeme nazývať *kódovaním symbolov zdrojovej abecedy* slovami nad abecedou Σ_C. Množina $V = \{\nu_0, \ldots, \nu_{m-1}\}$ sa nazýva *kód* a prvky množiny V sa nazývajú *kódovými slovami*. Všimneme si, že kódovanie po písmenách je totálnym (všade definovaným) zobrazením a keďže zdroj S generuje len postupnosti znakov nad abecedou Σ_S, každá správa vytvorená zdrojom S sa dá vyjadriť pomocou postupnosti kódových slov kódu V. Problém však vzniká pri dekódovaní kódovaných správ. Predpokladajme, že je daná nejaká správa $s_{i_1}, \ldots, s_{i_n}$ nad zdrojovou abecedou a jej prislúchajúca kódovaná správa vyjadrená ako postupnosť kódových slov $\nu_{i_1}, \ldots, \nu_{i_n}$. Postupnosť $\nu_{i_1}, \ldots, \nu_{i_n}$ sa však prenáša po znakoch a pred dekódovaním je potrebné ju rozdeliť na kódové slová. Ak sa to podarí, nie je problém dekódovať jednotlivé kódové slová a získať pôvodnú správu $s_{i_1}, \ldots, s_{i_n}$. Nasledujúci príklad ukazuje, že existujú také kódy, pre ktoré sa nie každá postupnosť kódových symbolov dá jednoznačne rozdeliť na kódové slová.

Príklad. Abeceda zdroja $\Sigma_S = \{0, 1, 2, 3\}$ pozostáva zo štyroch symbolov (prvých štyroch desiatkových číslic) a kódová abeceda je binárna; $\Sigma_C = \{0, 1\}$. Kódovanie priraďuje de-

siatkovej číslici jej binárny zápis:

$$\begin{array}{cccccc} 0 & \to & 0 & 1 & \to & 1 \\ 2 & \to & 10 & 3 & \to & 11 \end{array}$$

Uvažujme napr. binárnu postupnosť 001011. Táto postupnosť sa dá interpretovať viacerými spôsobmi, a síce ako binárny zápis desiatkových postupností 001011, 00103, 00211, 0023.

Jednoznačnosť dekódovania je prirodzenou požiadavkou, ktorá sa kladie na kódovanie. Nutným predpokladom jednoznačnosti dekódovania je tzv. *rozdeliteľnosť kódu*.

Definícia 2.3.1. *Kód* $V = \{v_0, \ldots, v_{m-1}\}$ *nad abecedou* Σ_C *sa nazýva rozdeliteľným, ak pre ľubovoľnú rovnosť postupností kódových slov*

$$v_{i_1} \ldots v_{i_k} = v_{j_1} \ldots v_{j_l}$$

platí $l = k, \quad i_1 = j_1, \ldots, i_k = j_k$.

Čo vlastne vyjadruje rozdeliteľnosť kódu? Ak je kód V rozdeliteľný, znamená to, že ľubovoľnú postupnosť nad Σ_C^* buď môžeme rozdeliť na postupnosť kódových slov jednoznačným spôsobom, alebo ju nemôžeme rozdeliť vôbec. Pre rozdeliteľný kód nemôže nastať taká situácia, kedy by sme nejakú postupnosť kódových symbolov mohli rozdeliť na postupnosť kódových slov dvoma rozličnými spôsobmi. Jednoduchým riešením problému rozdeliteľnosti sú *blokové* alebo *rovnomerné kódy*. Blokový kód sa vyznačuje tým, že všetky jeho kódové slová majú rovnakú dĺžku.

Príklad. Rozšírime predchádzajúci príklad a uvedieme dva spôsoby binárneho kódovania desiatkových číslic—rovnomerné a nerovnomerné:

desiatkový zápis	binárny zápis	blokový kód
0	0	0000
1	1	0001
2	10	0010
3	11	0011
4	100	0100
5	101	0101
6	110	0110
7	111	0111
8	1000	1000
9	1001	1001

Dekódovanie postupnosti znakov kódovej abecedy bude v prípade blokového kódu relatívne jednoduché: postupnosť sa najprv rozdelí na slová dĺžky rovnej dĺžke bloku a potom sa (napríklad na základe tabuľky) jednotlivým kódovým slovám priradia symboly zdrojovej abecedy.

Príklad. Postupnosť 100001001100100100110010 rozdelíme na kódové slová:
1000 0100 1100 1001 0011 0010 a dekódujeme pomocou tabuľky z predchádzjúceho príkladu: 843932. Všimneme si, že existujú aj binárne postupnosti, ktoré sa nedajú dekódovať, nakoľko slová 1111, 1110, 1101, 1100, 1011, 1010 nie sú kódové slová.

S rozdeliteľnosťou vznikajú problémy pri použití niektorých kódov, ktoré obsahujú slová nerovnakej dĺžky; tzv. *nerovnomerných kódoch*. Postupnosť kódových symbolov v tomto prípade nemožno mechanicky rozdeliť na bloky rovnakej dĺžky, ale je potrebné určiť kódové slová. To sa v prípade nerovnomerných kódov vo všeobecnosti nemusí dať spraviť (alebo nedá spraviť jednoznačne). Ale aj medzi nerovnomernými kódmi existujú rozdeliteľné kódy. Nakoľko tieto kódy umožňujú zapisovať informáciu častokrát úspornejšie ako blokové kódy, používajú sa najmä na (bezstratovú) kompresiu údajov. Podrobnejšie sa nimi budeme zaoberať v nasledujúcich kapitolách. Vráťme sa teraz ku kódovaniu zdrojovej informácie. Zatiaľ sme kódovali jednotlivé znaky kódovej abecedy, teraz pojem kódovania znakov zdrojovej abecedy zovšeobecníme.

Definícia 2.3.2. *Nech je* Σ_S *zdrojová abeceda, nech je množina* $U = \{u_0, \ldots, u_M\}$ *nejakých slov nad zdrojovou abecedou a nech sú* $v_0, \ldots, v_M$ *slová nad kódovou abecedou* Σ_C. *Zobrazenie*

$$\begin{aligned} u_0 &\rightarrow v_0 \\ u_1 &\rightarrow v_1 \\ &\vdots \\ u_M &\rightarrow v_M \end{aligned}$$

budeme nazývať kódovaním množiny U kódom V.

Všimneme si, že táto definícia kódovania zahŕňa aj kódovanie znakov zdrojovej abecedy; stačí položiť $U = \Sigma_S$.

Príklad. Nech je $\mathcal{U}$ rovná množine všetkých podmnožín množiny prirodzených čísel $\{0, \ldots .99\}$, $V = \{0,1\}^{100}$ je množina binárnych vektorov dĺžky 100. Podmnožine $\{i_0, \ldots, i_k\}$ z $\mathcal{U}$ je priradené slovo v_j[5], ktoré má jednotkové hodnoty na pozíciách $i_0, \ldots, i_k$ a nuly na ostatných pozíciách. Je zrejmé, že V kóduje množinu $\mathcal{U}$ a že toto kódovanie je bijekciou. Poznajúc mohutnosť kódu V vieme určiť aj mohutnosť množiny $\mathcal{U}$: $|\mathcal{U}| = 2^{100}$.

[5] tzv. charakteristický vektor

Časť I

Kódovanie zdroja

Kapitola 3

Nerovnomerné kódy

Vhodný kód na kódovanie daného zdroja informácie môžeme spravidla vybrať z viacerých kandidátov. To ktorý z nich nakoniec použijeme, závisí od účelu ktorý chceme kódovaním zdroja dosiahnuť. Ako sme už spomenuli v predchádzajúcej kapitole, jednou z prirodzených požiadaviek na kódovanie zdroja je, aby zakódovaná správa bola čo najkratšia (a zároveň jednoznačne dekódovateľná), aby sa pri kódovaní nepoužívali zbytočne dlhé kódové slová; resp. aby kódovanie bolo efektívne[1]. Ak má množina (znakov alebo slov) ktorú potrebujeme kódovať mohutnosť n, tak na rozlíšenie prvkov kódovanej množiny budeme potrebovať blokový kód so slovami dĺžky aspoň $\lceil \log_m n \rceil$, kde m je mohutnosť kódovej abecedy. V prípade, keď zdroj informácie generuje všetky znaky približne rovnako často (a nie sú známe iné využiteľné vzťahy medzi znakmi/slovami zdrojových správ)[2], je celkom efektívne kódovanie zdoja pomocou blokových kódov. Iná situácia však nastane, keď sa niektoré zo symbolov zdrojovej abecedy (slov nad zdrojovou abecedou) vyskytujú v správach výrazne častejšie ako iné; to znamená, ak sa množstvo informácie obsiahnuté v jednotlivých zdrojových symboloch (slovách nad zdrojovou abecedou) výrazne odlišuje. V takomto prípade by kódovanie správ pomocou nerovnomerných kódov, v ktorých by boli častejšie sa vyskytujúcim symbolom (slovám) priradené kratšie kódové slová efektívnejšie[3], ako použitie blokových (rovnomerných) kódov.

Základným predpokladom praktickej použiteľnosti nerovnomerných kódov je rozdeliteľnosť. V tejto kapitole sa budeme zaoberať rozdeliteľnými nerovnomernými kódami. Najprv zavedieme veľmi užitočnú triedu efektívne dekódovateľných nerovnomerných kódov, tzv. *prefixové kódy*. Potom dokážeme Kraftovu-McMillanovu nerovnosť, ktorá predstavuje kritérium pre existenciu (nerovnomerného) rozdeliteľného kódu s danými dĺžkami kódových slov. Nakoniec zavedieme pojem ceny kódu, skonštruujeme dolný odhad ceny kódu a budeme sa zaoberať konštrukciami optimálnych a kvázioptimálnych kódov. Kvôli zjednodušeniu výkladu budeme v priebehu tejto kapitoly predpokladať, že kódová abeceda je binárna a ak nebude explicitne povedané inak, budeme kódovať znaky zdrojovej abecedy; to znamená správy vytvorené zdrojom informácie budeme kódovať

[1] Zatiaľ vystačíme s intuitívnym chápaním efektívnosti kódovania, neskôr ho upresníme pomocou pojmu ceny kódu.

[2] jednotlivé zdrojové symboly obsahujú približne rovnaké množstvo informácie

[3] pre m-prvkovú kódovú abecedu a n prvkovú kódovanú množinu bude priemerný počet znakov kódovej abecedy potrebný na zakódovanie jedného prvku kódovanej množiny nižší ako $\lceil \log_m n \rceil$

po znakoch.

3.1 Rozdeliteľné kódy

3.1.1 Prefixové kódy

Štúdium nerovnomerných rozdeliteľných kódov začneme skúmaním základných vlastností prefixových kódov. Prefixové kódy totiž predstavujú rozsiahlu triedu nerovnomerných kódov s dobrými vlastnosťami (vyznačujú sa najmä rozdeliteľnosťou a jednoduchosťou dekódovania), ktoré budeme používať priamo na kódovanie zdrojových údajov, ale aj na konštrukciu iných kódov a pri skúmaní parametrov nerovnomerných kódov. Definujeme *prefixový kód* formálne.

Definícia 3.1.1. *Kód* $V = \{v_0, \ldots, v_{m-1}\}$ *sa nazýva prefixovým kódom, ak pre ľubovoľné* $v_i, v_j \in V$, $i \neq j$; v_i *nie je prefixom slova* v_j.

Prefixový kód sa teda vyznačuje tým, že žiadne jeho slovo nemôže byť počiatočným podslovom iného kódového slova. Kód z príkladu 2.3 nebol prefixový; kódové slovo 1 bolo prefixom kódového slova 10. „Prefixovosť“ kódu je taká silná vlastnosť, že z nej vyplýva rozdeliteľnosť kódu; ináč povedané, prefixovosť je postačujúcou podmienkou pre rozdeliteľnosť kódu. Sformulujeme a dokážeme toto tvrdenie formálne.

Veta 3.1.1. *Nech je* $V = \{v_0, \ldots, v_{m-1}\}$ *(binárny) prefixový kód, potom je* V *(binárny) rozdeliteľný kód.*

Dôkaz. Predpokladajme, že V je prefixový, ale nie rozdeliteľný kód. Potom existuje aspoň jedna binárna postupnosť, ktorá je rozdeliteľná na postupnosť kódových slov aspoň dvoma rozličnými spôsobmi. Vyberieme zo všetkých takých binárnych postupností postupnosť β s minimálnou dĺžkou. Pre postupnosť β teda platí :

$$v_{i_1} \ldots v_{i_k} = v_{j_1} \ldots v_{j_l}.$$

Z toho, že postupnosť β má minimálnu dĺžku vyplýva, že $v_{i_1} \neq v_{j_1}$. V opačnom prípade by bolo totiž možné slovo v_{i_1} z postupnosti β vynechať a dostali by sme kratšiu binárnu postupnosť, pre ktorú by platilo:

$$v_{i_2} \ldots v_{i_k} = v_{j_2} \ldots v_{j_l}.$$

To je však v spore s predpokladom o minimálnej dĺžke postupnosti β. Ak však $v_{i_1} \neq v_{j_1}$, potom buď slovo v_{i_1} je prefixom slova v_{j_1} alebo slovo v_{j_1} je prefixom slova v_{i_1}. To je zasa v spore s predpokladom o tom, že kód V je prefixový. To znamená, že postupnosť kódových symbolov, ktorá sa dá rozdeliť na postupnosť kódových slov aspoň dvoma rôznymi spôsobmi nemôže existovať, a teda kód V je rozdeliteľný. □

Tvrdenie sme síce dokázali pre binárny prípad, ale platí všeobecne pre ľubovoľnú kódovú abecedu, ktorá obsahuje aspoň dva symboly. Prefixovosť teda postačuje na to, aby bol kód rozdeliteľný; prirodzenou otázkou je, či je prefixovosť zároveň nutným predpokladom rozdeliteľnosti kódu, alebo ináč povedané, či existujú aj iné rozdeliteľné kódy okrem prefixových. Ukazuje sa, že nie. Uvedieme príklad rozdeliteľného kódu, ktorý nie je prefixový.

Príklad 3.1. *Kód* $V = \{0, 01, 11\}$ *síce nie je prefixový, ale napriek tomu to je rozdeliteľný kód.*

Kód z predchádzajúceho príkladu je tzv. *sufixový kód*, ktorý sa vyznačuje tým, že žiadne kódové slovo nie je sufixom iného kódového slova. Vytvorili sme ho tak, že sme „otočili“ slová prefixového kódu $\{0, 10, 11\}$. Postupnosť kódových symbolov správy kódovanej pomocou sufixového kódu budeme rozdeľovať na kódové slová „odzadu“; t.j. až vtedy, keď máme k dispozícii celú kódovanú správu. Príkladom takejto postupnosti, ktorá sa nedá rozdeliť na postupnosť kódových slov, kým sa nedočíta do konca, je postupnosť:

$$0111\ldots1$$

Ak táto postupnosť obsahuje párny počet jednotiek (napr. $2k$), dá sa rozdeliť nasledovne: $0(11)^k$; ak obsahuje nepárny počet jednotiek ($2k+1$), tak sa rozdelí na kódové slová nasledovne: $01(11)^k$.

TO DO: silne rozdeliteľné kódy

3.1.2 Kraftova - McMillanova nerovnosť

Aby sme získali čo najkratší zápis správy, snažíme sa na kódovanie používať kódy s krátkymi kódovými slovami. Ak je mohutnosť abecedy zdroja menšia alebo rovná mohutnosti kódovej abecedy, tak potom možno znaky zdrojovej zbecedy kódovať slovami dĺžky 1 (znakmi kódovej abecedy). V opačnom prípade (a tých je prevažná väčšina) budeme potrebovať použiť kód s väčšími dĺžkami kódových slov. Je zrejmé, že si dĺžky kódových slov nemôžeme voliť ľubovoľne; keďže existujú len dve binárne slová dĺžky 1 (0 a 1) a štyri binárne slová dĺžky 2 (00,01,10,11) binárne kódy s tromi slovami dĺžky 1 alebo piatimi slovami dĺžky 2 zrejme nemôžu existovať. Ale existuje napríklad binárny kód so štyrmi kódovými slovami dĺžok $1, 2, 2, 2$; resp. existuje rozdeliteľný kód nad abecedou mohutnosti $q \geq 2$ s dĺžkami kódových slov $l_i = l(v_i);\ i = 0, \ldots, m-1$? Na túto otázku dáva odpoveď veta 3.1.2, ktorú vyslovíme a dokážeme v tejto časti. Kvôli zjednodušeniu výkladu budeme v ďalšom predpokladať, že kódová abeceda je binárna a že zdrojová abeceda Σ_S obsahuje aspoň dva symboly, t.j. $m \geq 2$.

Veta 3.1.2 (Kraftova-McMillanova nerovnosť)**.** *Nech sú* $l_0, \ldots, l_{m-1}$ *ľubovoľné nenulové prirodzené čísla. Potom rozdeliteľný kód* $V = \{v_0, \ldots, v_{m-1}\}$ *s dĺžkami kódových slov* $l_i = l(v_i);\ i = 0, \ldots, m-1$ *existuje práve vtedy, ak platí nasledujúca nerovnosť*

$$\sum_{i=0}^{m-1} 2^{-l_i} \leq 1. \tag{3.1}$$

Dôkaz. Najprv dokážeme, že podmienka je nutná. Nech je $V = \{v_0, \ldots, v_{m-1}\}$ ľubovoľný kód. Priradíme mu generujúcu funkciu (enumerátor dĺžok kódových slov) definovanú nasledujúcim spôsobom:

$$h_V(x) = \sum_{i=0}^{m-1} x^{-l(v_i)}.$$

Zavedieme teraz n-násobné zreťazenie (rozšírenie) kódu V;

$$V^n = \{w_i;\ w_i = v_{i_1} \ldots v_{i_n},\ v_{i_j} \in V,\ j = 1, \ldots, n\}$$

Bude nás zaujímať, aký je vzťah medzi vytvárajúcimi funkciami kódu V a jeho rozšírenia, V^n. Kvôli lepšiemu pochopeniu si tento problém najprv ilustrujeme na jednoduchom príklade.

Príklad 3.2. *Uvažujme kód* $V = \{0, 10, 11\}$. *Jeho vytvárajúca funkcia je*

$$h_V(x) = x^{-1} + x^{-2} + x^{-2} = x^{-1} + 2x^{-2}.$$

Dvojnásobným zreťazaním kódu V *dostávame kód*

$$V^2 = \{00, 010, 011, 100, 1010, 1011, 110, 1110, 1111\}$$

s enumerátorom

$$h_{V^2}(x) = x^{-2} + 4x^{-3} + 4x^{-4} = (x^{-1} + 2x^{-2})^2.$$

Pokračovanie dôkazu. To, čo sme videli na príklade, platí aj vo všeobecnosti a dá sa dokázať matematickou indukciou; t.j.

$$h_{V^n}(x) = \left(\sum_{i=0}^{m-1} x^{-l(v_i)} \right)^n. \tag{3.2}$$

Predpokladajme teraz, že $V = \{v_0, \ldots, v_{m-1}\}$ je rozdeliteľný kód s dĺžkami kódových slov $l_i = l(v_i),\ i = 0, \ldots, m-1$. Symbolom M_i označíme počet slov dĺžky i v rozšírenom kóde V^n a pomocou hodnôt M_i zapíšeme enumerátor kódu V^n. Označíme maximálnu dĺžku slova v kóde V symbolom l_{max}. Potom dĺžka ľubovoľného slova kódu V^n nepresiahne $n \cdot l_{max}$ (v kóde aspoň jedno slovo takej dĺžky existuje, a je to slovo, ktoré sme dostali n-násobným zreťazením slova maximálnej dĺžky kódu V). To znamená, že $M_i = 0$ pre $i > n \cdot l_{max}$ a generujúcu funkciu kódu V^n môžeme vyjadriť nasledovne

$$h_{V^n}(x) = \sum_{i=0}^{n \cdot l_{max}} M_i x^{-i}. \tag{3.3}$$

Dosadíme v (3.3) namiesto premennej x hodnotu 2 a dostávame:

$$h_{V^n}(2) = \sum_{i=0}^{n \cdot l_{max}} 2^{-i} M_i. \tag{3.4}$$

Všimneme si, že suma v (3.4) môže obsahovať nulové členy; ak totiž V neobsahuje slovo nulovej dĺžky ε, tak každé slovo v kóde V^n bude mať dĺžku minimálne $n \cdot l_{min}$, kde l_{min} je minimálna dĺžka kódového slova kódu V. Potom $M_0 = M_1 = M_2 = \cdots = M_{n \cdot l_{min}-1} = 0$. Podobne, ak by kód V obsahoval len slová párnej dĺžky, tak ani kód V^n nemôže obsahovať slová nepárnej dĺžky. Využijeme teraz skutočnosť, že kód V je rozdeliteľný. Z toho vyplýva, že všetky slová kódu V^n sú rôzne, a keďže V^n je binárny kód, znamená to, že $M_i \leq 2^i$. V opačnom prípade by sa aspoň jedna binárna postupnosť dĺžky i musela dať poskladať zo slov kódu V rôznymi spôsobmi, čo je v spore s predpokladom o rozdeliteľnosti kódu V. Na druhej strane niektoré binárne postupnosti sa nemusia dať poskladať zo slov kódu V a v tomto prípade $M_i < 2^i$. Dosadíme horný odhad hodnoty M_i do vzťahu (3.4) a po jednoduchých úpravách dostávame:

$$h_{V^n}(2) = \sum_{i=1}^{n \cdot l_{max}} 2^{-i} M_i \leq \sum_{i=1}^{n \cdot l_{max}} 2^{-i} 2^i = n \cdot l_{max}. \tag{3.5}$$

Na druhej strane, zo vzťahov (3.2), (3.5) vyplýva

$$h_{V^n}(2) = \left(\sum_{i=0}^{m-1} 2^{-l(v_i)} \right)^n \leq n \cdot l_{max}. \tag{3.6}$$

Ale nerovnosť (3.6) platí pre ľubovoľné n. Ak by teda

$$\sum_{i=0}^{m-1} 2^{-l(v_i)} = a > 1,$$

tak by existovalo také n_0, že pre všetky $n > n_0$ by

$$a^n > n \cdot l_{max},$$

pretože exponenciálna funkcia so základom $a > 1$ rastie rýchlejšie ako polynomická. To však je v spore so vzťahom (3.6), a teda platí

$$\sum_{i=0}^{m-1} 2^{-l(v_i)} \leq 1.$$

Postačujúcosť. Predpokladajme, že dĺžky kódových slov sú usporiadané vzostupne; $l_0 \leq l_1 \leq \cdots \leq l_{m-2} \leq l_{m-1}$ a že pre ne platí Kraftova-McMillanova nerovnosť. Ukážeme, že je možné zostrojiť rozdeliteľný kód s dĺžkami kódových slov $l_0, \ldots, l_{m-1}$.
1. konštrukcia [1]. Použijeme matematickú indukciu.

1. Vyberieme ľubovoľné binárne slovo dĺžky l_0 ako kódové slovo v_0.

2. Predpokladáme, že sme už vybrali slová $v_0, \ldots, v_{k-1}$, $k \leq m-1$, dĺžok $l_0, \ldots, l_{k-1}$ ktoré tvoria rozdeliteľný (prefixový) kód.

3. V množine binárnych vektorov dĺžky l_k nájdeme také slovo v_k (dĺžky l_k), že žiadne zo slov $v_0, \ldots, v_{k-1}$ nie je jeho prefixom. Ukážeme, že také slovo existuje. Všetkých binárnych slov dĺžky l_k, ktoré majú prefix v_0 dĺžky l_0 je $2^{l_k - l_0}$. (Prvých l_0 bitov sa zhoduje so slovom v_0, ostatných $l_k - l_0$ bitov možno vybrať ľubovoľným spôsobom.) Všetkých binárnych slov dĺžky l_k, ktorých prefixom je niektoré zo slov $v_0, \ldots, v_{k-1}$ je

$$\sum_{i=0}^{k-1} 2^{l_k - l_i}. \tag{3.7}$$

Teraz využijeme predpoklad, že pre doteraz zostrojený kód platí Kraftova-McMillanova nerovnosť a že $k < m$:

$$\sum_{i=0}^{m-1} 2^{-l_i} \leq 1.$$

Rozdelíme sumu z poslednej nerovnosti na dve časti:

$$\sum_{i=0}^{m-1} 2^{-l_i} = \sum_{i=0}^{k-1} 2^{-l_i} + \sum_{i=k}^{m-1} 2^{-l_i} \leq 1. \tag{3.8}$$

Keďže že všetky sčítance v sume (3.8) sú kladné čísla, vynechaním niektorých členov druhej sumy z (3.8) sa nerovnosť zachová (existuje aj taká možnosť, že druhá suma bude obsahovať len jediný člen, 2^{-l_k}):

$$\sum_{i=0}^{k-1} 2^{-l_i} + 2^{-l_k} \leq 1. \tag{3.9}$$

Vynásobíme nerovnosť (3.9) hodnotou 2^{l_k} a upravíme

$$\sum_{i=0}^{k-1} 2^{l_k - l_i} \leq 2^{l_k} - 1. \tag{3.10}$$

Zo nerovnosti (3.10) vyplýva, že, existuje aspoň jeden binárny vektor dĺžky l_k, ktorého prefixom nie je žiadne zo slov $v_0, \ldots, v_{k-1}$. Vyberieme tento vektor ako kódové slovo v_k. Takýmto spôsobom napokon zostrojíme prefixový kód $V = \{v_0, \ldots, v_{m-1}\}$ s dĺžkami kódových slov $\{l_0, \ldots, l_{m-1}\}$, čím sme dokázali tvrdenie vety.

Predchádzajúci dôkaz mal skôr existenčný ako konštruktívny charakter. Dokážeme ešte raz, že ak platí Kraftova-McMillanova nerovnosť, tak potom možno zostrojiť prefixový kód požadovaných vlastností. Využijeme na to konštrukciu Shannonovho kódu[4].

2. konštrukcia [6]. Rovnako ako v predchádzajúcom dôkaze budeme predpokladať, že dĺžky kódových slov sú usporiadané vzostupne; $l_0 \leq l_1 \leq \cdots \leq l_{m-1}$ a že pre ne platí

[4] Ku koštrukcii Shannonovho kódu sa ešte vrátime vo vete 3.3.2

Kraftova-McMillanova nerovnosť. Zavedieme čísla $q_k,\ k = 0,\dots,m-1$ odvodené od dĺžok kódových slov. Čísla q_k definujeme nasledovne:

$$q_0 = 0, \quad q_k = \sum_{i=0}^{k-1} 2^{-l_i}; \quad k = 1,\dots,m-1. \tag{3.11}$$

Z Kraftovej-McMillanovej nerovnosti vyplýva, že čísla $q_k;\ k = 0,\dots,m-1$ spĺňajú podmienky $0 \le q_k < 1$. Zapíšeme teraz čísla q_k v binárnom tvare. Zo spôsobu vytvárania q_k a zo skutočnosti, že $l_0 \le l_1 \le \cdots \le l_{m-1}$, vyplýva, že q_k sa dá zapísať ako

$$q_k = (0.b_{k,1}\dots b_{k,l_{k-1}})_2,$$

kde $b_{k,i} \in \{0,1\}$. Kód $V = \{\nu_0,\dots,\nu_{m-1}\}$ vytvoríme potom z binárnej reprezentácie čísel q_k nasledujúcim spôsobom:

$$\nu_i = \underbrace{b_{i,1}\dots b_{i,l_{i-1}}0\dots0}_{l_i};$$

t.j. kódové slovo ν_i pozostáva z prvých l_i binárnych číslic nasledujúcich po rádovej čiarke v binárnom rozvoji čísla q_i. Tvrdíme, že takto zostrojený kód je prefixový, a teda aj rozdeliteľný. Predpokladajme, že kód V nie je prefixový. Potom obsahuje kódové slová, z ktorých jedno je prefixom druhého. Nech h je najmenšie také číslo, že pre slovo ν_h existuje kódové slovo (označme ho symbolom ν_i), ktoré je jeho prefixom. Keďže ν_i je prefixom ν_h, $l_i < l_h$, a teda aj $i < h$. Z toho že ν_i je prefixom ν_h a zo spôsobu konštrukcie kódových slov vyplýva, že ν_i je prefixom slov $\nu_{i+1},\dots,\nu_{h-1},\nu_h$. Keďže ν_h je prvé kódové slovo, ktoré má prefix ν_i, $h = i+1$. Pozrieme sa teraz na slová $\nu_i,\ \nu_{i+1}$ podrobnejšie, preskúmame čísla $q_i,\ q_{i+1}$.

$$\begin{aligned} q_i &= 0.\overbrace{\underbrace{b_{i,1}\dots b_{i,l_{i-1}}}_{l_{i-1}}0\dots0}^{l_i} \\ q_{i+1} &= q_i + 2^{-l_i} = 0.\underbrace{b_{i,1}\dots b_{i,l_{i-1}}0\dots10\dots0}_{l_{i+1}} \end{aligned}$$

Môžu nastať dve možnosti:

1. $l_i < l_{i+1}$; v tomto prípade má slovo ν_i na mieste l_i znak 0 a slovo ν_{i+1} znak 1;
2. $l_i = l_{i+1}$. Pripočítaním hodnoty 2^{-l_i} ku q_i sa zmení niektorá z prvých l_i číslic čísla q_i, a teda slová ν_i a ν_{i+1} sa odlišujú aspoň v jednom z prvých l_i znakov.

To znamená, že ν_i nemôže byť prefixom slova ν_{i+1}, a teda kód V je prefixový. □

Dôsledok 1. *Pre ľubovoľný rozdeliteľný kód* $V = \{v_0, \ldots, v_{m-1}\}$ *existuje prefixový kód* $W = \{w_0, \ldots, w_{m-1}\}$, *taký, že* $l(v_i) = l(v_i),\ i = 0, \ldots, m-1$.

Z uvedeného dôsledku vyplýva, že ak nám nezáleží na konkrétnej podobe kódových slov, môžeme rozdeliteľný kód s dĺžkami kódových slov $l_0, \ldots, l_{m-1}$ nahradiť prefixovým kódom s tými istými dĺžkami kódových slov; $l_0, \ldots, l_{m-1}$. Túto možnosť budeme v ďalších úvahách využívať a budeme často predpokladať, že rozdeliteľný kód je zároveň aj prefixový kód. Skôr ako budeme pokračovať v skúmaní vlastností nerovnomerných kódov, uvedieme príklad Shannonovho kódu, ktorý sme použili v dôkaze Kraftovej-McMillanovej nerovnosti.

Príklad 3.3. *Uvažujme nasledujúce dĺžky kódových slov: 2,3,3,4,5,5,6,6. Keďže* $2^{-2} + 2^{-3}+2^{-3}+2^{-4}+2^{-5}+2^{-5}+2^{-6}+2^{-6} = 0.65625 < 1$, *z vety 3.1.2 vyplýva, že existuje prefixový kód s týmito dĺžkami kódových slov. Vypočítame hodnoty* q_i *a vyjadríme príslušné kódové slová.*

$$\begin{array}{lll} q_0 = 0.00 & l_0 = 2 & v_0 = 00 \\ q_1 = 0.01 & l_1 = 3 & v_1 = 010 \\ q_2 = 0.011 & l_2 = 3 & v_2 = 011 \\ q_3 = 0.1 & l_3 = 4 & v_3 = 1000 \\ q_4 = 0.1001 & l_4 = 5 & v_4 = 10010 \\ q_5 = 0.10011 & l_5 = 5 & v_5 = 10011 \\ q_6 = 0.101 & l_6 = 6 & v_6 = 101000 \\ q_7 = 0.101001 & l_7 = 6 & v_7 = 101001 \end{array}$$

3.1.3 Úplné kódy

Kód z príkladu 3.3 je síce prefixový, ale má jeden vážny nedostatok. Existujú binárne postupnosti, ktoré sa nedajú rozbiť na kódové slová. Okrem triviálnych postupností dĺžky 1 sú to napríklad postupnosti začínajúce dvojicou symbolov 11. Nemá však zmysel požadovať, aby platilo $V^* = B^*$, pretože to je možné len v prípade, ak $B \subseteq V$. Intuitívnej požiadavke, aby každá binárna postupnosť predstavovala alebo sa dala doplniť na postupnosť kódových slov, vyhovujú tzv. *úplné kódy*.

Definícia 3.1.2. *Binárny rozdeliteľný kód* V *sa nazýva úplným kódom, ak pre ľubovoľnú binárnu postupnosť* $\beta \in B^*$ *existuje také kódové slovo* $v_i \in V$, *že buď postupnosť* β *je prefixom slova* v_i, *alebo slovo* v_i *je prefixom postupnosti* β.

Príklad 3.4. *Uvažujme blokový kód* $V = \{00, 01, 10, 11\}$. *Keďže kód* V *obsahuje všetky binárne slová dĺžky 2, spĺňa podmienky definície 3.1.2 a je úplný. Každú binárnu postupnosť párnej dĺžky možno jednoznačne rozdeliť na postupnosť kódových slov.*

Overovať, či nejaký kód s veľkým počtom kódových slov spĺňa podmienky definície 3.1.2, by nemuselo byť jednoduché. Našťastie úplnosť kódu úzko súvisí s Kraftovou-McMillanovou nerovnosťou a prefixovými kódmi.

Veta 3.1.3. *Binárny rozdeliteľný kód* $V = \{v_0, \ldots, v_{m-1}\}$ *je úplný práve vtedy, ak je prefixový a platí*

$$\sum_{i=0}^{m-1} 2^{-l_i} = 1. \tag{3.12}$$

Dôkaz. Predpokladajme, že $V = \{v_0, \ldots, v_{m-1}\}$ je binárny prefixový kód, pre ktorý platí rovnosť (3.12), pritom však V nie je úplný. To znamená, že existuje binárna postupnosť $\beta \in B^*$ taká, že žiadne kódové slovo $v_i \in V$ nie je prefixom postupnosti β a postupnosť β nie je prefixom žiadneho kódového slova kódu V. Potom však môžeme zostrojiť nový binárny prefixový kód $V' = V \cup \{\beta\}$, pre ktorý platí

$$\sum_{i=0}^{m-1} 2^{-l_i} + 2^{-l(\beta)} = 1 + 2^{-l(\beta)} > 1. \tag{3.13}$$

Ale nerovnosť (3.13) je v spore s Kraftovou-McMillanovou nerovnosťou (3.1). To znamená, že postupnosť β požadovaných vlastností nemôže existovať, a teda kód V je úplný.

Dokážeme druhú časť tvrdenia sporom. Nech je V úplný rozdeliteľný kód. Predpokladajme, že V nie je prefixový, alebo pre V neplatí rovnosť (3.12). Keďže z rozdeliteľnosti kódu V vyplýva platnosť Kraftovej-McMillanovej nerovnosti (3.1), znamená to, že pre kód V platí

$$\sum_{i=0}^{m-1} 2^{-l_i} < 1. \tag{3.14}$$

Zhrnieme naše predpoklady: kód V je úplný a platí $\sum_{i=0}^{m-1} 2^{-l_i} < 1$. Z úplnosti kódu V vyplýva, že každá binárna postupnosť dĺžky $n > l_{max}$, kde

$$l_{max} = \max_{v_i \in V}\{l(v_i)\}$$

musí mať ako prefix nejaké kódové slovo. Spočítame počet takýchto postupností:

$$\sum_{i=0}^{m-1} 2^{n-l_i} \geq 2^n. \tag{3.15}$$

Ak by kód V bol prefixový, potom je kódové slovo, ktoré je prefixom nejakej binárnej postupnosti dĺžky n dané jednoznačne. Potom by však platilo

$$\sum_{i=0}^{m-1} 2^{n-l_i} = 2^n, \tag{3.16}$$

a

$$\sum_{i=0}^{m-1} 2^{-l_i} = 1, \tag{3.17}$$

čo je v spore s predpokladom (3.14). To znamená, že platí $\sum_{i=0}^{m-1} 2^{-l_i} < 1$, kód V je úplný ale nie je prefixový. Potom však existujú kódové slová $v_i \neq v_j$ také, že (napr.) v_i je prefixom v_j. Z úplnosti kódu V vyplýva, že každá binárna postupnosť dĺžky $n > l_{max}$ musí začínať nejakým kódovým slovom kódu V. Potom

$$\sum_{i=0}^{m-1} 2^{n-l_i} > 2^n, \tag{3.18}$$

lebo postupnosti začínajúce slovom v_j sú už zarátané v sume (3.18) ako postupnosti začínajúce slovom v_i. Na druhej strane nemôže platiť nerovnosť

$$\sum_{k=0}^{m-1} 2^{n-l_k} - 2^{n-l(v_j)} < 2^n, \tag{3.19}$$

pretože to by znamenalo, že odstránením slova v_j z kódu V sa stratí úplnosť kódu; t.j. potom bude existovať binárna postupnosť β dĺžky $n > l_{max}$, ktorej prefixom nie je žiadne kódové slovo kódu V. Ale to znamená, že jej prefixom nemohlo byť odstránené slovo v_j, pretože v tom prípade by prefixom postupnosti β bolo slovo v_i, a teda kód V by nebol úplný. To znamená, že platí nerovnosť (3.18). Platnosť nerovnosti (3.18) je však v rozpore s tvrdením vety 3.1.2. Dostávame spor, ktorý dokazuje platnosť nášho tvrdenia. □

3.1.4 Kódové stromy

Na skúmanie vlastností nie príliš rozsiahlych nerovnomerných kódov je možné výhodne používať orientovaný ohodnotený graf, nazývaný *kódovým stromom*. Uvažujme orientovaný binárny strom $\mathcal{T}$ hĺbky n s hranami a vrcholmi ohodnotenými nasledujúcim spôsobom: najprv ohodnotíme jeho hrany, pričom budeme postupovať od koreňa k listom; ak z vrcholu vychádzajú dve (neohodnotené) hrany tak jednej z nich priradíme hodnotu 0 a druhej hodnotu 1. Ak z vrcholu vychádza jediná (neohodnotená) hrana, priradíme jej jednu z hodnôt $\{0,1\}$. Po ohodnotení hrán ohodnotíme vrcholy binárneho stromu $\mathcal{T}$: koreňu priradíme prázdne slovo ε a vrcholu v priradíme postupnosť binárnych hodnôt, ktoré boli priradené hranám ležiacim na ceste, spájajúcej koreň s vrcholom v.[5] Keďže $\mathcal{T}$ je súvislý acyklický graf, medzi ľubovoľnými dvoma vrcholmi v ňom existuje jediná cesta, a teda binárna postupnosť priradená vrcholu je určená jednoznačne. Binárny kódový strom $\mathcal{T}(\mathcal{V})$ binárneho kódu V dostaneme tak, že z binárneho stromu $\mathcal{T}$ hĺbky $n \geq l_{max}$, kde $l_{max} = \max_{v_i \in V}\{l(v_i)\}$ a binárny strom $\mathcal{T}$ je ohodnotený spôsobom uvedeným vyššie, odstránime všetky podstromy, ktoré neobsahujú vrchol s ohodnotením zodpovedajúcim niektorému kódovému slovu kódu V. Na obr. 3.1 je zobrazený binárny (ohodnotený) strom hĺbky 2.

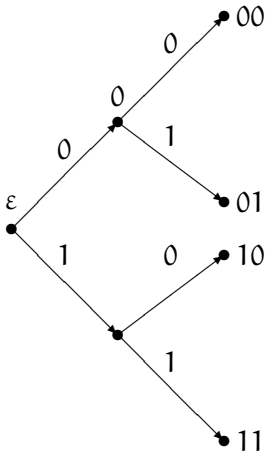
Binárny kódový strom kódu V z príkladu 3.3 je zobrazený na obr. 3.2 Všimneme si, že všetky vrcholy zodpovedajúce kódovým slovám, sú listy (vrcholy, z ktorých nevychádzajú žiadne hrany). To nie je náhoda. Ak by nejaké slovo v_i bolo prefixom iného slova v_j, vrchol v_i by musel ležať na ceste spájajúcej koreň s vrcholom v_j, a teda by musel byť vnútorným vrcholom kódového stromu.

Veta 3.1.4. *Nech je V prefixový kód. Potom v kódovom strome $\mathcal{T}(\mathcal{V})$ zodpovedajú kódovým slovám listy.*

Dôkaz. Prenechávame čitateľovi.

Pomocou kódového stromu je možné ľahšie formulovať aj podmienku úplnosti kódu. Ako sme už ukázali, kód V z príkladu 3.3 nie je úplný; problémy spôsobujú postupnosti

[5] V ďalšom budeme vrchol v označovať binárnym slovom, ktoré mu je priradené.



Obr. 3.1: Ohodnotený binárny strom

začínajúce dvojicou 11. Pri skúmaní kódového stromu kódu V zistíme, že z vrcholu 1 vychádza len jedna hrana, ktorej je priradená hodnota 0. Ak túto hranu odstránime a vrchol 1 stotožníme s pôvodným vrcholom 10, dostaneme kódový strom $\mathcal{T}(\mathcal{V}')$ prefixového kódu $V' = \{00, 010, 011, 100, 1010, 1011, 11000, 11001\}$.

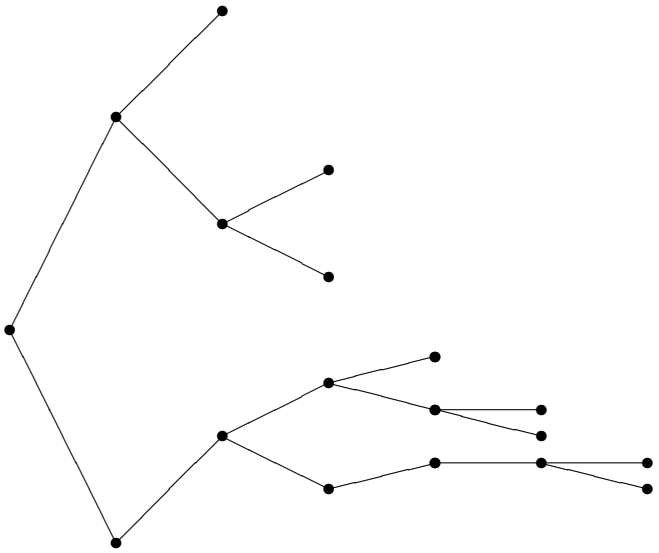
Kódový strom $\mathcal{T}(\mathcal{V}')$ obsahuje ešte dva vnútorné vrcholy $(11, 110)$ stupňa 1. Odstránením hrán vychádzajúcich z týchto vrcholov, vrcholu 110 a stotožnením vrcholov 11 a 1100 stromu $\mathcal{T}(\mathcal{V}')$ dostávame kódový strom (Obr. 3.3) $\mathcal{T}(\mathcal{V}'')$ prefixového kódu $V'' = \{00, 010, 011, 100, 1010, 1011, 110, 111\}$. Pre kód V'' platí $\sum_{v_i \in V''} 2^{l(v_i)} = 1$. Kód V'' je úplný. Každý binárny[6] prefixový kód, ktorý nie je úplný, možno týmto spôsobom transformovať na úplný kód.

3.1.5 Automatové dekódovanie.

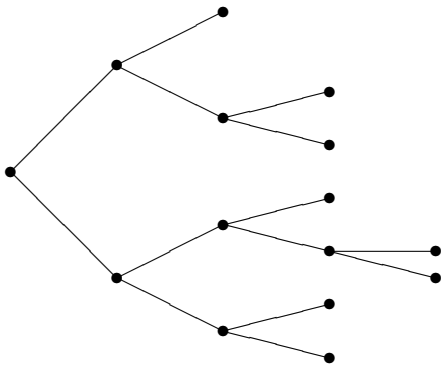
Veľkou prednosťou prefixových kódov je to, že okamžite po dočítaní posledného symbolu kódového slova dokážeme určiť, o aké kódové slovo ide. (Pre porovnanie pripomíname sufixový rozdeliteľný kód z príkladu 3.1, pre ktorý existovali správy, ktoré bolo možné dekódovať až po prijatí posledného symbolu správy.) Prefixové kódy sa vďaka možnosti priebežného dekódovania správy nazývajú aj okamžitými kódmi alebo automatovými kódmi. Ten druhý názov získali vďaka tomu, že na ich dekódovanie možno použiť konečný automat.

Definícia 3.1.3. *Konečný automat je usporiadaná šestica* $\mathcal{A} = (\Sigma_i, \Sigma_o, Q, \Phi, \Psi, q)$*, kde* Σ_i *je vstupná,* Σ_o *výstupná abeceda,* Q *je konečná množina stavov,* $\Phi : \Sigma_i \times Q \to Q$ *je*

[6] Ako uvidíme neskôr, mnohé z vlastností nerovnomerných kódov nezávisia od počtu znakov kódovej abecedy. Transformácia prefixového kódu na úplný prefixový kód, ktorú sme popísali vyššie, podstatne využíva to, že kódová abeceda je binárna; a nedá sa priamo zovšeobecniť na prípad kódovej abecedy s väčším počtom kódových symbolov.



Obr. 3.2: Kódový strom Shannonovho kódu



Obr. 3.3: Kódový strom skráteného Shannonovho kódu

prechodová funkcia, $\Psi : \Sigma_i \times Q \to \Sigma_o$ *je výstupná funkcia a* q *je počiatočný stav konečného automatu* $\mathcal{A}$.

Konečný automat si môžeme predstaviť ako zariadenie so vstupnou a výstupnou páskou, riadiacou jednotkou, čítacou a zapisovacou hlavou, obr. 3.4. Vstupná páska je rozdelená na políčka, v každom políčku je zapísaný symbol vstupnej abecedy. Podobne je výstupná páska rozdelená na políčka a v políčku je zapísaný jeden zo symbolov výstupnej abecedy, alebo je políčko prázdne. Čítacia hlava sa pohybuje po vstupnej páske zľava doprava, v každom kroku číta jeden symbol z políčka vstupnej pásky a po prečítaní sa presunie o jedno políčko doprava. Zapisovacia hlava v každom kroku zapisuje na políčko výstupnej pásky jeden symbol výstupnej abecedy a posunie sa o jedno políčko doprava, alebo nezapíše nič a zostáva na tom istom políčku aj v nasledujúcom kroku. Automat začína pracovať v počiatočnom stave q a skončí, keď prečíta celý vstup zo vstupnej pásky.

Pri dekódovaní binárnych prefixových kódov pomocou konečného automatu bude vstupná abeceda $\Sigma_i = \{0, 1\}$, výstupná abeceda sa bude zhodovať so zdrojovou abecedou; $\Sigma_o = \Sigma_S$ a prechodovú a výstupnú funkciu definujeme pomocou tabuľky. Ilustrujeme dekódovanie binárnárneho prefixového kódu na príklade.

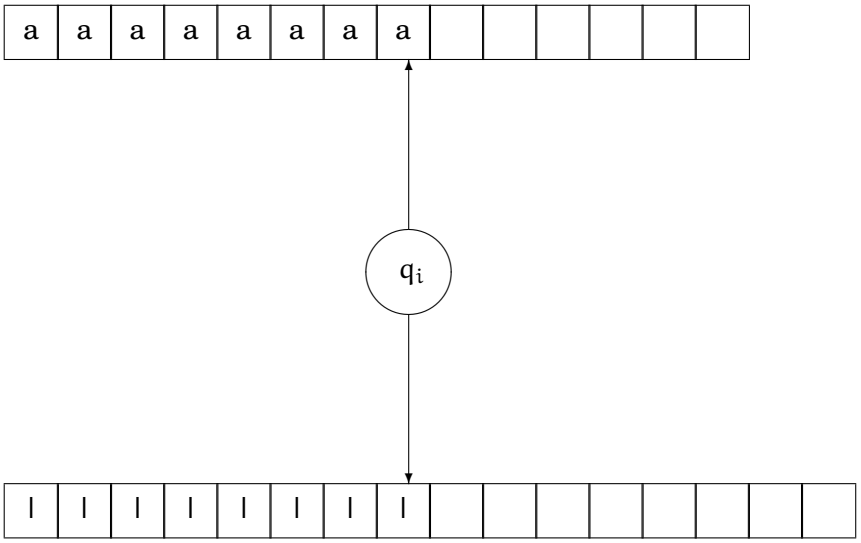
Príklad 3.5. *Uvažujme kód* V'' *z predchádzajúceho príkladu. Predpokladáme, že kódové slová slúžia na zápis prvých písmen anglickej abecedy:*

a	*00*	*e*	*1010*
b	*010*	*f*	*1011*
c	*011*	*g*	*110*
d	*100*	*h*	*111*

Vstupná abeceda konečného (dekódovacieho) automatu $\mathcal{A}$ *je binárna:* $\Sigma_i = \{0, 1\}$, *výstupná abeceda* $\Sigma_o = \{a, b, c, d, e, f, g, h, \lambda\}$, *množina stavov* $Q = \{q, q_0, q_1, q_{01}, q_{10}, q_{11}, q_{101}\}$ *a prechodová a výstupná funkcia sú uvedené v tabuľke. Počiatočným stavom je* q.

stav	*vstup*	
	0	1
q	q_0, λ	q_1, λ
q_0	q, a	q_{01}, λ
q_{01}	q, b	q, c
q_1	q_{10}, λ	q_{11}, λ
q_{10}	q, d	q_{101}, λ
q_{101}	q, e	q, f
q_{11}	q, g	q, h

Je daná binárne kódovaná správa 010011111. *Ukážeme, ako ju automat* $\mathcal{A}$ *dekóduje. Kvôli jednoduchosti budeme pozíciu čítacej hlavy na vstupnej páske a stav automatu* $\mathcal{A}$ *zapisovať tak, že stav automatu zapíšeme pred symbol, ktorý v danom kroku automat* $\mathcal{A}$ *číta. Symboly, ktoré by sa zapisovali na výstupnej páske budeme zapisovať pod dekódované slová binárnej správy.*



Obr. 3.4: Konečný automat

$$q010011111 \mapsto 0q_0 10011111 \mapsto 01q_{01}0011111 \mapsto \underbrace{010}_{b} q011111 \mapsto$$
$$\underbrace{010}_{b} 0q_0 11111 \mapsto \underbrace{010}_{b} 01q_{01}1111 \mapsto \underbrace{010}_{b}\underbrace{011}_{c} q111 \mapsto \underbrace{010}_{b}\underbrace{011}_{c} 1q_1 11 \mapsto$$
$$\underbrace{010}_{b}\underbrace{011}_{c} 11q_{11}1 \mapsto \underbrace{010}_{b}\underbrace{011}_{c}\underbrace{111}_{h} q$$

3.2 Cena kódu

Nerovnomerné rozdeliteľné kódy sa dajú výhodne použiť v takých prípadoch, keď sa slová (alebo znaky), ktoré sa kódujú, vyskytujú nerovnako často. Vtedy je možné často sa vyskytujúcim slovám (znakom) priradiť kratšie kódové slová a tak dosiahnuť, že kódovaná správa bude v priemernom prípade kratšia, ako keby sa na kódovanie používali napríklad blokové kódy. V ďalšom túto intuitívnu predstavu upresníme. Kvôli jednoduchosti budeme kódovať znaky zdrojovej abecedy $\Sigma_S = \{a_0, \ldots, a_{m-1}\}$. Zavedieme prvý, značne zjednodušený matematický model zdroja S. Budeme predpokladať, že zdroj S je náhodný generátor, ktorý generuje znaky zdrojovej abecedy náhodne a nezávisle na sebe. Zdroj S je charakterizovaný rozdelením pravdepodobností $P = \{p_0, \ldots, p_{m-1}\}$; $p_i \geq 0$, $i = 0, \ldots, m-1$; $\sum_{i=0}^{m-1} p_i = 1$ výskytu jednotlivých symbolov zdrojovej abecedy. (Z matematického hľadiska je zdroj S náhodná premenná, nadobúdajúca hodnotu a_i s pravdepodobnosťou p_i, $i = 0, \ldots, m-1$.) Je zrejmé, že existuje viacero spôsobov kódovania znakov zdrojovej abecedy. Aby sme mohli porovnať efektívnosť jednotlivých kódov, zavedieme pojem *ceny kódu*.

Definícia 3.2.1. *Nech* $P = \{p_0, \ldots, p_{m-1}\}$ *je rozdelenie pravdepodobností znakov zdrojovej abecedy* $\Sigma_S = \{a_0, \ldots, a_{m-1}\}$*; nech* $V = \{v_0, \ldots, v_{m-1}\}$ *je kód kódujúci znaky kódovej*

abecedy, $a_i \to v_i,\ i = 0, \ldots, m-1$ *a nech* $l_i = l(v_i)$ *sú dĺžky kódových slov kódu* V. *Potom cenou kódu* V *pri rozdelení pravdepodobností nazveme*

$$\mathcal{L}(P, V) = \sum_{i=0}^{m-1} l_i p_i.$$

Cena kódu V pri rozdelení pravdepodobností P nie je z matematického hľadiska nič iné, než stredná hodnota dĺžky kódového slova, počet symbolov kódovej abecedy pripadajúcich na zakódovanie jedného znaku zdrojovej abecedy. (V prípade kódovania slov z nejakej množiny M by to bol počet symbolov kódovej abecedy pripadajúcich na zakódovanie jedného slova z množiny M.)

Aká je minimálna hodnota $\mathcal{L}(P, V)$ pri danom rozdelení pravdepodobností? Existujú kódy dosahujúce túto minimálnu hodnotu a ak áno, sú známe metódy ich zostrojovania? Na tieto i ďalšie otázky dáme odpoveď v nasledujúcich častiach tejto kapitoly.

3.3 Kvázioptimálne kódy a optimálny kód

Kód $V = \{v_0, \ldots, v_{m-1}\}$ s dĺžkami kódových slov $l(v_i) = l_i,\ i = 0, \ldots, m-1$ nazveme optimálnym kódom pre rozdelenie pravdepodobností $P = \{p_0, \ldots, p_{m-1}\}$, ak pre ľubovoľný kód $W = \{w_0, \ldots, w_{m-1}\}$ platí

$$\mathcal{L}(P, V) \leq \mathcal{L}(P, W).$$

Cenu optimálneho kódu pri rozdelení pravdepodobností P označíme $\mathcal{L}(P)$. Prirodzená otázka je, aká je cena optimálneho kódu.

Veta 3.3.1. *Nech je* $P = \{p_0, \ldots, p_{m-1}\}$ *ľubovoľné rozdelenie pravdepodobností,* $p_0 \geq p_1 \geq \cdots \geq p_{m-1} > 0$. *Potom platí*

$$\sum_{i=0}^{m-1} p_i \cdot \lg \frac{1}{p_i} \leq \mathcal{L}(P) \leq \sum_{i=0}^{m-1} p_i \cdot \lg \frac{1}{p_i} + 1.$$

Rovnosť

$$\sum_{i=0}^{m-1} p_i \cdot \lg \frac{1}{p_i} = \mathcal{L}(P) \tag{3.20}$$

platí práve vtedy, ak $p_i = 2^{-l_i},\ l_i \in N,\ i = 0, \ldots, m-1$.

Dôkaz. Dolný odhad. Predpokladajme, že $V = \{v_0, \ldots, v_{m-1}\}$ je ľubovoľný prefixový kód s dĺžkami kódových slov $l(v_i) = l_i,\ i = 0, \ldots, m-1$. Porovnáme cenu kódu V s entropiou zdroja $H_2(\mathcal{P}) = \sum_{i=0}^{m-1} p_i \cdot \lg \frac{1}{p_i}$:

$$\sum_{i=0}^{m-1} p_i \cdot \lg \frac{1}{p_i} - \sum_{i=0}^{m-1} l_i p_i = \sum_{i=0}^{m-1} p_i \cdot \left[\lg \frac{1}{p_i} - \lg 2^{l_i}\right] = \sum_{i=0}^{m-1} p_i \cdot \lg \frac{2^{-l_i}}{p_i}.$$

Teraz prevedieme binárny logaritmus na prirodzený, využijeme nerovnosť $\ln x \leq x - 1$ a upravíme:

$$\sum_{i=0}^{m-1} p_i \cdot \lg \frac{2^{-l_i}}{p_i} = \frac{1}{\ln 2} \sum_{i=0}^{m-1} p_i \cdot \ln \frac{2^{-l_i}}{p_i} \leq \frac{1}{\ln 2} \sum_{i=0}^{m-1} p_i \cdot \left[\frac{2^{-l_i}}{p_i} - 1 \right] = \tag{3.21}$$

$$= \frac{1}{\ln 2} \left[\sum_{i=0}^{m-1} 2^{-l_i} - \sum_{i=0}^{m-1} p_i \right] = \frac{1}{\ln 2} \left[\sum_{i=0}^{m-1} 2^{-l_i} - 1 \right].$$

Kód V je prefixový a teda z Kraftovej-McMillanovej nerovnosti vyplýva, že $\sum_{i=0}^{m-1} 2^{-l_i} \leq 1$. Keďže $2 > 1,\ \ln 2 > 0$ platí

$$\frac{1}{\ln 2} \left[\sum_{i=0}^{m-1} 2^{-l_i} - 1 \right] \leq 0.$$

To však znamená, že pre ľubovoľný prefixový kód $V = \{v_0, \ldots, v_{m-1}\}$ platí

$$H_2(\mathcal{P}) \leq \mathcal{L}(P, V).$$

Horný odhad. Dokážeme, že existuje (prefixový) kód, ktorý dosahuje cenu $H_2(\mathcal{P}) + 1$. Položíme $l_i = \lceil \lg \frac{1}{p_i} \rceil$. Potom platí[7]:

$$\sum_{i=0}^{m-1} 2^{-l_i} = \sum_{i=0}^{m-1} 2^{-\lceil \lg \frac{1}{p_i} \rceil} \leq \sum_{i=0}^{m-1} 2^{-\lg \frac{1}{p_i}} = \sum_{i=0}^{m-1} p_i = 1;$$

a teda existuje prefixový kód s dĺžkami kódových slov $l_i = \lceil \lg \frac{1}{p_i} \rceil,\ i = 0, \ldots, m-1$. Cena tohto kódu je

$$\mathcal{L}(P, V) = \sum_{i=0}^{m-1} p_i l_i = \sum_{i=0}^{m-1} p_i \lceil \lg \frac{1}{p_i} \rceil \leq \sum_{i=0}^{m-1} p_i \left[\lg \frac{1}{p_i} + 1 \right] = \sum_{i=0}^{m-1} p_i \lg \frac{1}{p_i} + 1.$$

Vráťme sa ešte k dôkazu rovnosti (3.20). Ak $P = \{p_i = 2^{-l_i},\ l_i \in N,\ i = 0, \ldots, m-1\}$ je rozdelenie pravdepodobností, potom podľa vety 3.1.2 existuje prefixový kód s dĺžkami kódových slov

$$\lceil \lg \frac{1}{p_i} \rceil = \lceil \lg \frac{1}{2^{-l_i}} \rceil = \lceil \lg 2^{l_i} \rceil = \lceil l_i \rceil = l_i,$$

ktorý má cenu

$$\sum_{i=0}^{m-1} p_i l_i = \sum_{i=0}^{m-1} p_i \lg \frac{1}{p_i}.$$

Na druhej strane, ak

$$\sum_{i=0}^{m-1} p_i l_i = \sum_{i=0}^{m-1} p_i \lg \frac{1}{p_i}$$

[7] Pripomíname, že výraz $\lg x$ označuje binárny logaritmus čísla x.

to znamená, že v odvodení 3.21 nastala rovnosť. To však znamená, že $\frac{2^{-l_i}}{p_i} = 1$, resp. $p_i = 2^{-l_i}$ ($\ln x = x - 1$, pre $x = 1$). □

Jeden kód, ktorého cena sa veľmi nelíši od ceny optimálneho kódu už poznáme. Je to Shannonov kód. Konštrukcia, ktorú sme použili v dôkaze Kraftovej-McMillanovej nerovnosti však vychádzala zo znalosti dĺžok kódových slov a nie z rozdelenia pravdepodobností zdrojových symbolov. Ukážeme, ako možno zostrojiť Shannonov prefixový kód pre dané rozdelenie pravdepodobností zdrojových symbolov.

3.3.1 Shannonov kód

Veta 3.3.2 (Shannonov kód). *Nech je* $P = \{p_0, \dots, p_{m-1}\}$ *ľubovoľné rozdelenie pravdepodobností,* $p_0 \geq p_1 \geq \cdots \geq p_{m-1} > 0$; *čísla* $q_k \quad 0 \leq q_k < 1$ *sú definované nasledovne*

$$\begin{aligned} q_0 &= 0, \\ q_k &= \sum_{j=0}^{k-1} p_j, \qquad k = 1, \dots, m-1; \end{aligned}$$

a nech pre prirodzené čísla (dĺžky kódových slov) platí

$$l_k = \lceil \lg 1/p_k \rceil \quad \textit{pre } k = 0, \dots, m-1.$$

Ďalej, nech

$$q_k = 0.b_1^{(k)} \dots b_{l_k}^{(k)} \dots$$

je binárny zápis čísla q_k, $(k = 0, \dots, m-1)$. *Pre* $k = 0, \dots, m-1$ *definujeme kódové slovo*

$$w_k = b_1^{(k)} \dots b_{l_k}^{(k)}.$$

Potom kód $W = \{w_0, \dots, w_{m-1}\}$ *je prefixový kód, nazývaný Shannonovým kódom.*

Dôkaz. Dokážeme, že kód $W = \{w_0, \dots, w_{m-1}\}$ je prefixový. Pozrieme sa najprv na dĺžky kódových slov. Zapíšeme pravdepodobnosti zdrojových symbolov binárne:

$$p_k = a_1^{(k)} \dots a_{l_k}^{(k)} \dots \text{ pre } k = 0 \dots m-1.$$

Pripomenieme, že pre dĺžky kódových slov platí $l_k = \lceil \lg 1/p_k \rceil$, $k = 0, \dots, m-1$. To znamená, že

$$\begin{array}{ll} \frac{1}{2} \leq p_k < 1 & l_k = 1 \\ \frac{1}{4} \leq p_k < \frac{1}{2} & l_k = 2 \\ \frac{1}{8} \leq p_k < \frac{1}{4} & l_k = 3 \\ \dots & \dots \end{array}$$

t.j. hodnota l_k je jednoznačne určená pozíciou prvej jednotky v binárnom zápise čísla p_k; ak

$$p_k = 0.\underbrace{0\ldots01}_{s}\ldots$$

tak potom $2^{-s} \leq p_k < 2^{-s+1}$, $2^{s-1} < 1/p_k \leq 2^s$ a teda $l_k = s$. Predpokladajme teraz, že kód W nie je prefixový. To znamená, že existujú kódové slová slová w_r, w_t; $r < t$ také, že w_r je prefixom slova w_t. Kódové slová w_r, w_t boli vytvorené z čísel

$$\begin{aligned} q_r &= 0.b_1^{(r)}\ldots b_{l_r}^{(r)}\ldots \\ q_t &= 0.b_1^{(t)}\ldots b_{l_r}^{(t)}\ldots b_{l_t}^{(t)}\ldots; \\ q_t &= q_r + p_r + \cdots + p_{t-1}. \end{aligned}$$

Pre pravdepodobnosť p_r však platí $2^{-l_r} \leq p_r < 2^{-l_r+1}$. To znamená, že

$$2^{-l_r} \leq p_r + \cdots + p_{t-1}.$$

Potom sa však q_t odlišuje aspoň na jednom z prvých l_r miest po rádovej čiarke od q_r, lebo už pre $t = r + 1$ platí

$$\begin{array}{rrcll} & q_r & = & 0.b_1^{(r)}\ldots & b_{l_r}^{(r)} \ldots \\ + & p_r & = & 0.0\ldots & 1 \ldots \\ = & q_{r+1} & \neq & 0.b_1^{(r)}\ldots & b_{l_r}^{(r)} \ldots \end{array}$$

a teda w_r nie je prefixom slova w_t. □

Skôr, ako ukážeme, ako sa konštruuje optimálny kód, uvedieme ešte jednu jednoduchú metódu konštrukcie kódu, ktorého cena je blízka k cene optimálneho kódu, Fanov kód. (Shannonov a Fanov kód sa nazývajú *kvázioptimálne kódy.*)

3.3.2 Fanov kód

Fanova konštrukcia kvázioptimálneho kódu. Predpokladáme, že je dané rozdelenie pravdepodobností $P = \{p_0, \ldots, p_{m-1}\}$ symbolov zdrojovej abecedy,

1. usporiadame pravdepodobnosti zostupne (napríklad) $p_0 \geq p_1 \geq \cdots \geq p_{m-1} > 0$ a zapíšeme do 1. stĺpca tabuľky. Jednotlivým pravdepodobnostiam (zastupujúcim symboly zdrojovej abecedy) priradíme prázdne slová ε.
2. Ak tabuľka obsahuje aspoň dva riadky, rozdelíme ju na 2 časti tak, aby sa súčet pravdepodobností v hornej časti tabuľky líšil čo najmenej od súčtu pravdepodobností v dolnej časti tabuľky a pokračujeme krokom 3. Ak tabuľka obsahuje jediný riadok, jej spracovanie ukončíme.
3. Slová v_i priradené pravdepodobnostiam v hornej polovici tabuľky zreťazíme sprava so znakom 0, a slová z dolnej polovice tabuľky zreťazíme sprava so znakom 1. Pokračujeme v spracovaní hornej a dolnej časti tabuľky podľa kroku 2.

Keďže tabuľka obsahuje m riadkov, krok 2 sa uplatní najviac $m - 1$-krát. Ilustrujeme konštrukciu Fanovho a Shannonovho kódu na nasledujúcom príklade.

Príklad 3.6.

p_0	0.25	0	00		
p_1	0.20	0	01		
p_2	0.13	1	10	100	
p_3	0.12	1	10	101	
p_4	0.10	1	11	110	1100
p_5	0.08	1	11	110	1101
p_6	0.07	1	11	111	1110
p_7	0.05	1	11	111	1111

Fanov kód

p_i	q_i	l_i	ν_i
0.25	0.0	2	00
0.20	0.010	3	010
0.13	0.0111	4	0111
0.12	0.1001	4	1001
0.10	0.1011	4	1011
0.08	0.1100	4	1100
0.07	0.1110	4	1110
0.05	0.11110	5	11110

Shannonov kód

Fanov kód má cenu 2.85 a Shannonov 3.35 a entropia je 2.822. Shannonov kód ešte možno upraviť (skrátiť). Keďže slovo 011 nie je prefixom iného kódového slova okrem slova 0111, môžeme slovo 0111 nahradiť jeho prefixom 011, podobne 100 je prefixom jediného kódového slova, a preto možno toto slovo 1001 nahradiť slovom 100; rovnako možno skrátiť kódové slovo 1011 na 101; kódové slovo 1100 na 110 a napokon kódové slovo 11110 na 1111. Takto upravený (skrátený) Shannonov kód má cenu 2.87.

3.3.3 Huffmanov optimálny kód

Uvedieme teraz metódu konštrukcie optimálneho kódu. Podstata Huffmanovej metódy spočíva v tom, že sa zostrojenie optimálneho kódu pre m znakov redukuje na konštrukciu optimálneho kódu pre $m-1$ znakov. Pri dôkaze budeme potrebovať nasledujúcu vetu.

Veta 3.3.3. *Nech je* $P = \{p_0, \dots, p_{m-1}\}$ *ľubovoľné rozdelenie pravdepodobností,* $p_0 \geq p_1 \geq \cdots \geq p_{m-1} > 0$. *Potom existuje prefixový kód* $V = \{\nu_0, \dots, \nu_{m-1}\}$ *s dĺžkami kódových slov* $l_i = l(\nu_i)$, $i = 0, \dots, m-1$, *optimálny pre rozdelenie pravdepodobností* P, *taký, že minimálnym pravdepodobnostiam* p_{m-2}, p_{m-1} *zodpovedajú slová* ν_{m-2}, ν_{m-1} *maximálnej dĺžky* l_{m-1}, *ktoré majú spoločný prefix dĺžky* $l_{m-1} - 1$.

Dôkaz. Ak by v kóde V neboli priradené slová maximálnej dĺžky minimálnym pravdepodobnostiam, kód by nebol optimálny. To znamená, že minimálnym pravdepodobnostiam p_{m-2}, p_{m-1} musia byť priradené slová maximálnej dĺžky. Predpokladajme, že slová v_{m-2}, v_{m-1} nemajú spoločný prefix dĺžky $l_{m-1} - 1$. To znamená, že existuje slovo v_j maximálnej dĺžky l_{m-1}, ktoré má spoločný prefix dĺžky $l_{m-1} - 1$ so slovom v_{m-1}. V opačnom prípade by slovo v_{m-1} bolo možné nahradiť jeho prefixom dĺžky $l_{m-1} - 1$, čo je v spore s optimálnosťou kódu V. (Z podobných dôvodov musí existovať slovo v_k maximálnej dĺžky l_{m-1}, ktoré má spoločný prefix dĺžky $l_{m-1} - 1$ so slovom v_{m-2}.) „Zámenou“ slov v_{m-2} a v_j dostávame kód s rovnakou cenou, ako bola cena pôvodného kódu, spĺňajúci podmienky vety. □

Teraz už môžeme vysloviť a dokázať vetu, ktorá je teoretickým zdôvodnením Huffmanovej konštrukcie optimálneho kódu.

Veta 3.3.4. Huffmanov kód. *Nech* $V = \{v_0, \ldots, v_{m-1}\}$, $m > 1$ *je optimálny prefixový kód pre rozdelenie pravdepodobností* $P = \{p_0, \ldots, p_{m-1}\}$, *pričom* $p_j = q_0 + q_1$ *a* $p_0 \geq p_1 \geq \cdots \geq p_{m-1} \geq q_0 \geq q_1 > 0$. *Potom kód* $V' = \{v_0 \ldots, v_{j-1}, v_{j+1}, \ldots, v_{m-1}, v_j 0, v_j 1\}$ *je optimálny kód pre rozdelenie pravdepodobností* $P' = \{p_0 \ldots, p_{j-1}, p_{j+1}, \ldots, p_{m-1}, q_0, q_1\}$.

Dôkaz Kód V' je tiež prefixový a jeho cena je $\mathcal{L}(P', V') = \mathcal{L}(P, V) + p_j$. Aby sme ukázali, že V' je optimálny kód, musíme dokázať, že pre ľubovoľný kód $W' = \{w_0, \ldots, w_m\}$ pre rozdelenie pravdepodobností P' platí $\mathcal{L}(P', W') \geq \mathcal{L}(P', V') = \mathcal{L}(P, V) + p_j$. Predpokladajme, že W' je optimálny kód pre rozdelenie pravdepodobností P', ktorý naviac spĺňa podmienky vety 3.3.3. To znamená, že minimálnym pravdepodobnostiam q_0, q_1 zodpovedajú slová maximálnej dĺžky $w1, w0$. Uvažujme teraz kód $W = \{w_0, \ldots, w_{j-1}, w, w_{j+1}, \ldots, w_{m-1}\}$ pre rozdelenie pravdepodobností P. Keďže pre rozdelenie pravdepodobností P je optimálny kód V, platí $\mathcal{L}(P, V) \leq \mathcal{L}(P, W)$. Ale potom

$$\mathcal{L}(P', V') = \mathcal{L}(P, V) + p_j \leq \mathcal{L}(P, W) + p_j = \mathcal{L}(P', W'),$$

a teda kód V' je optimálny pre rozdelenie pravdepodobností P'. □

Popíšeme metódu konštrukcie Huffmanovho kódu pre rozdelenie pravdepodobností $P = \{p_0, \ldots, p_{m-1}\}$.

Konštrukcia optimálneho kódu.

1. usporiadaj pravdepodobnosti $p_0, \ldots, p_{m-1}$ do zoznamu zostupne. Ak $m > 1$ pokračuj krokom 2, inák choď na krok 3.
2. Opakuj $m - 2$ krát nasledujúcu činnosť:
   - sčítaj posledné dve (minimálne) pravdepodobnosti usporiadaného zoznamu;
   - odstráň tieto dve pravdepodobnosti zo zoznamu a zaraď do zoznamu ich súčet tak, aby bol nový zoznam usporiadaný zostupne;
   - zapamätaj si miesto v zozname, na ktoré bola zaradená nová hodnota.

3. Zoznam obsahuje dve pravdepodobnosti; priraď (napríklad) väčšej z nich slovo 0 a menšej slovo 1; ak $m = 1$ skonči, ináč pokračuj krokom 4.

4. Opakuj $m - 2$ krát nasledujúcu činnosť a potom skonči:

   - urči tú pravdepodobnosť p_j v aktuálnom usporiadanom zozname, ktorá bola vytvorená ako posledná súčtom nejakých dvoch minimálnych pravdepodobností q_0, q_1;
   - odstráň pravdepodobnosť p_j zo zoznamu, doplň doň pravdepodobnosti q_0, q_1 a usporiadaj ho;
   - ak bolo pravdepodobnosti p_j priradené slovo v, priraď pravdepodobnostiam q_0, q_1 slová $v0, v1$.

Ilustrujeme Huffmanovu konštrukciu na príklade.

Príklad 3.7.

0.25	0.25	0.25	0.25	0.31*	0.44*	0.56*	0*	1*	00*	01	01	01	01
0.20	0.20	0.20	0.24*	0.25	0.31	0.44	1	00	01	10*	11	11	11
0.13	0.13	0.18*	0.20	0.24	0.25			01	10	11	000*	001	001
0.12	0.12	0.13	0.18	0.20					11	000	001	100	100
0.10	0.12*	0.12	0.13							001	100	101*	0000
0.08	0.10	0.12									101	0000	0001
0.07	0.08											0001	1010
0.05													1011

Huffmanov kód

Pravdepodobnosti, ktoré vznikli sčítaním minimálnych pravdepodobností v predchádzajúcom kroku, sú označené hviezdičkou. Kvôli jednoduchosti sú hviezdičkou označené aj slová prislúchajúce týmto pravdepodobnostiam.

Huffmanov kód má cenu 2.85. Je zaujímavé, že aj keď sú Fanov a Huffmanov kód rôzne, majú rovnakú cenu. Vo všeobecnosti však Huffmanova metóda umožňuje získať lepšie kódy ako Fanova metóda vďaka tomu, že „preusporiadavaním“ pravdepodobností lepšie „vyvažuje“ tabuľku pravdepodobností. Čo však v tom prípade, keď je tabuľka pravdepodobností nevyvážená už na samom začiatku; ak sa jeden symbol vyskutuje veľmi často a ostatné zriedkavo? Uvedieme extrémny prípad a ukážeme, ako sa dá riešiť.

3.3.4 Rozšírenie kódu

Uvažujeme nasledujúci prípad. Zdrojová abeceda pozostáva zo symbolov $\{a, b\}$ a rozdelenie pravdepodobností je $P = \{0.9, 0.1\}$. Optimálny kód $V = \{0, 1\}$ má cenu $\mathcal{L} = 1.0$ a entropia zdroja je 0.4689955936. Rozdiel medzi entropiou a cenou optimálneho kódu je príliš veľký. Podstata problému je v tom, že zdrojová abeceda je príliš malá a nemáme možnosť rozlíšiť často (a) a zriedkavo (b) sa vyskytujúce symboly, ale obom sme priradili slová rovnakej dĺžky. Pri kódovaní zdrojovej abecedy s takým extrémnym rozdelením pravdepodobností uplatníme nasledujúci postup. Namiesto jednotlivých znakov kódovej abecedy budeme kódovať n-tice znakov zdrojovej abecedy. Využijeme pritom predpoklady o

charaktere zdroja: znaky generuje nezávisle na sebe a s nemennými pravdepodobnosťami. „Abeceda“ Σ_s^2, jej rozdelenie pravdepodobností a príslušný Huffmanov kód V_2 sú uvedené v nasledujúcej tabuľke.

aa	0.81	0
ab	0.09	11
ba	0.09	100
bb	0.01	101

Cena kódu V_2 je 1.29. Treba si však uvedomiť, že to je počet binárnych symbolov pripadajúcich na jedno zdrojové slovo, ktoré má dĺžku 2, a preto cena kódu, meraná počtom kódových symbolov potrebných na zakódovanie jedného symbolu zdrojovej abecedy je $\mathcal{L}(P', V_2) = 1.29/2 = 0.645$. Táto hodnota je už podstatne bližšia k entropii zdroja, ako cena pôvodného kódu. Ďalšie rozšírenie zdrojovej abecedy (kódovanie trojznakových slov nad zdrojovou abecedou) už neprinesie takú podstatnú redukciu ceny kódu:

aaa	0.729	0
aab	0.081	100
aba	0.081	101
baa	0.081	110
abb	0.009	11100
bab	0.009	11101
bba	0.009	11110
bbb	0.001	11111

$\mathcal{L}(P'', V_3) = 1.599/3 = 0.533$. Aby sme si spravili predstavu o to, ako rýchlo sa približuje cena kódu pre narastajúcu hodnotu n k entropii, vypočítame cenu neskráteného Shannonovho kódu pre rozličné hodnoty n:

n	3	4	5	10	20	30	50	100	200	1000	2000
$\mathcal{L}$	0.6333	0.5509	0.5163	0.5070	0.5006	0.4863	0.4789	0.4741	0.4713	0.4695	0.4692

Upozorňujeme, že n-násobným rozšírením (dvojprvkovej) zdrojovej abecedy dostaneme množinu slov mohutnosti 2^n, a tak je použitie tejto metódy pre konštrukciu kódov s cenou blízkou k dolnej hranici danej entropiou pre väčšie hodnoty n a/alebo rozsiahlejšie abecedy zdroja prakticky nepoužiteľné.

Metóda konštrukcie Huffmanovho kódu nie je jednoznačná. Ak v zozname pravdepodobností už existuje hodnota rovná tej, ktorú sme dostali v niektorom kroku súčtom minimálnych pravdepodobností, máme možnosť zaradiť vypočítanú pravdepodobnosť za alebo pred pravdepodobnosť v pôvodnom usporiadanom zozname. Uplatnením rozličných stratégií zaraďovania rovnakých pravdepodobností do zoznamu, dostaneme kódy, ktoré majú rovnakú cenu, ale môžu mať rozličné dĺžky kódových slov.

Príklad 3.8.

0.375	0.375	0.375	0.375	0.625∗	0	1	1	1	1
0.250	0.250	0.250	0.375∗	0.375	1	00	01	01	01
0.125	0.125	0.250∗	0.25			01	000	001	001
0.125	0.125	0.125					001	0000	0000
0.0625	0.125∗							0001	00010
0.0625									00011

Huffmanov kód V

0.375	0.375	0.375	0.375∗	0.625∗	0	1	00	00	00
0.250	0.250	0.250∗	0.375	0.375	1	00	01	10	10
0.125	0.125∗	0.250	0.25			01	10	11	010
0.125	0.125	0.125					11	010	011
0.0625	0.125							011	110
0.0625									111

Huffmanov kód V′.

ľahko sa pritom presvedčíme o tom, že $\mathcal{L}(V, P) = \mathcal{L}(V', P) = 2.375$.

Ktorý zo zostrojených kódov je lepší? Ak sa v nejakom texte vyskytujú zdrojové symboly s pravdepodobnosťami zodpovedajúcimi pravdepodobnostiam z rozdelenia pravdepodobností P, oba kódy zakódujú daný text rovnako efektívne. Ak však kód zostrojujeme na základe predpokladaného rozdelenia pravdepodobností, ktoré sa od skutočného líši, môže sa cena skonštruovaného kódu viac alebo menej odlišovať od ceny optimálneho kódu a rozdiel medzi skutočnou a minimálnou cenou bude závisieť aj od spôsobu konštrukcie kódu.

3.3.5 Chyby v pravdepodobnostiach výskytu zdrojových symbolov

Predpokladajme, že je dané rozdelenie pravdepodobností $P = \{p_0, \ldots, p_{m-1}\}$, na základe ktorého sme zostrojili Huffmanov kód $V = \{v_0, \ldots, v_{m-1}\}$ s dĺžkami kódových slov $\{l_0, \ldots, l_{m-1}\}$. Nech je $P' = \{p'_0, \ldots, p'_{m-1}\}$, skutočné rozdelenie pravdepodobností zdrojových symbolov; $p'_i = p_i + e_i,\ i = 0, \ldots, m-1$, kde e_i je chyba v odhade pravdepodobnosti výskytu i-teho symbolu. Keďže P', P sú rozdelenia pravdepodobností, platí: $\sum_{i=0}^{m-1} p'_i = \sum_{i=0}^{m-1} p_i + e_i = 1 + \sum_{i=0}^{m-1} e_i$; a teda $\sum_{i=0}^{m-1} e_i = 0$. Zistíme, aký bude rozdiel cien kódu V pri rozdelení pravdepodobností P' a P:

$$\mathcal{L}(V, P') = \sum_{i=0}^{m-1} p'_i l_i = \sum_{i=0}^{m-1} (p_i + e_i) l_i = \sum_{i=0}^{m-1} p_i l_i + \sum_{i=0}^{m-1} l_i e_i = \mathcal{L}(V, P) + \sum_{i=0}^{m-1} l_i e_i$$

Zistíme, kedy nadobúda $\sum_{i=0}^{m-1} l_i e_i$ extrémne hodnoty. Použijeme na to metódu Lagrangeových neurčitých koeficientov [13]. Vyjadríme najprv podmienky, za ktorých budeme

hľadať extrémy funkcie $\sum_{i=0}^{m-1} l_i e_i$. Odchýlky e_i od pravdepodobností výskytu symbolov budeme chápať ako výsledky náhodnej premennej e; pričom $P(e = e_i) = \frac{1}{m}$, $i = 0, \ldots, m-1$. Potom stredná hodnota chyby je

$$E(e) = \sum_{i=0}^{m-1} e_i \frac{1}{m} = \frac{1}{m} \sum_{i=0}^{m-1} e_i = 0. \tag{3.22}$$

Vyjadríme disperziu chýb:

$$Var(e) = \sum_{i=0}^{m-1} e_i^2 \frac{1}{m} - \left(\frac{1}{m} \sum_{i=0}^{m-1} e_i \right)^2 = \frac{1}{m} \sum_{i=0}^{m-1} e_i^2 - 0 = \frac{1}{m} \sum_{i=0}^{m-1} e_i^2 = \sigma^2. \tag{3.23}$$

Využijeme (3.22) a (3.23) a zostrojíme Lagrangeovu funkciu pre veličinu $\sum_{i=0}^{m-1} l_i e_i$ (λ, μ sú Lagrangeove neurčité koeficienty):

$$\mathcal{F} = \frac{1}{m} \sum_{i=0}^{m-1} l_i e_i + \lambda \left(\frac{1}{m} \sum_{i=0}^{m-1} e_i \right) + \mu \left(\frac{1}{m} \sum_{i=0}^{m-1} e_i^2 - \sigma^2 \right). \tag{3.24}$$

Vypočítame parciálne derivácie funkcie $\mathcal{F}$ podľa e_i, $i = 0, \ldots, m-1$ a položíme ich rovnými nule:

$$\frac{\partial \mathcal{F}}{\partial e_i} = \frac{1}{m} (l_i - \lambda - 2\mu e_i) = 0; \quad i = 0, \ldots, m-1. \tag{3.25}$$

Sčítame rovnice (3.25) a vyjadríme koeficient λ.

$$\sum_{i=0}^{m-1} \frac{1}{m} (l_i - \lambda - 2\mu e_i) = \frac{1}{m} \sum_{i=0}^{m-1} l_i - \lambda - \frac{2\mu}{m} \sum_{i=0}^{m-1} e_i = 0;$$

a teda

$$\lambda = \frac{1}{m} \sum_{i=0}^{m-1} l_i. \tag{3.26}$$

Vrátime sa k sústave rovníc (3.25). Jednotlivé rovnice vynásobíme zodpovedajúcimi hodnotami e_i a výsledok spočítame cez všetky hodnoty i. Dostávame

$$\frac{1}{m} \sum_{i=0}^{m-1} \left(l_i e_i - \lambda e_i - 2\mu e_i^2 \right) = \frac{1}{m} \sum_{i=0}^{m-1} l_i e_i - \frac{\lambda}{m} \sum_{i=0}^{m-1} e_i - \frac{2\mu}{m} \sum_{i=0}^{m-1} e_i^2 = 0.$$

Upravíme

$$\frac{1}{m} \sum_{i=0}^{m-1} l_i e_i = \frac{2\mu}{m} \sum_{i=0}^{m-1} e_i^2,$$

a určíme koeficient μ:

$$\mu = \frac{1}{2m\sigma^2} \sum_{i=0}^{m-1} l_i e_i. \tag{3.27}$$

Napokon vynásobíme jednotlivé rovnice sústavy (3.25) príslušnými hodnotami l_i a výsledky násobenia sčítame cez všetky i:

$$\frac{1}{m}\sum_{i=0}^{m-1}\left(l_i^2-\lambda l_i-2\mu l_i e_i\right)=\frac{1}{m}\sum_{i=0}^{m-1}l_i^2-\frac{\lambda}{m}\sum_{i=0}^{m-1}l_i-\frac{2\mu}{m}\sum_{i=0}^{m-1}l_i e_i=0. \tag{3.28}$$

Dosadíme hodnoty konštánt μ,λ do (3.28) a upravíme

$$\begin{aligned}\frac{1}{m}\sum_{i=0}^{m-1}l_i^2-\left(\frac{1}{m}\sum_{i=0}^{m-1}l_i\right)^2-\frac{1}{\sigma^2m^2}\left(\sum_{i=0}^{m-1}l_i e_i\right)^2 &= 0;\\ \frac{1}{m}\sum_{i=0}^{m-1}l_i^2-\left(\frac{1}{m}\sum_{i=0}^{m-1}l_i\right)^2=\frac{1}{\sigma^2}\left(\frac{1}{m}\sum_{i=0}^{m-1}l_i e_i\right)^2&.\end{aligned}$$

$$\left[\frac{1}{m}\sum_{i=0}^{m-1}l_i^2-\left(\frac{1}{m}\sum_{i=0}^{m-1}l_i\right)^2\right]\cdot\sigma^2=Var(l)Var(e)=\left(\frac{1}{m}\sum_{i=0}^{m-1}l_i e_i\right)^2.$$

To znamená, že pre fixovanú hodnotu disperzie chýb, σ^2, sa extrémne odchýlky ceny kódu (v kladnom alebo zápornom smere) dosahujú pre kódy, ktoré majú veľkú disperziu dĺžok kódových slov. Ináč povedané, čím väčšie sú rozdiely v dĺžkach kódových slov, tým väčšiu ochýlku (zlepšenie alebo zhoršenie) ceny kódu môžu spôsobiť chyby v pravdepodobnostiach jednotlivých zdrojových symbolov. Príkladom kódu so stabilnou cenou je blokový kód, pre ktorý sa žiadne odchýlky v pravdepodobnostiach neprejavia zmenou ceny kódu. (Otázne však je, či pre skutočné rozdelenie pravdepodobností bude pôvodný kód optimálny. Tento problém naše odvodenie nerieši.)

Príklad 3.9. *Ilustrujeme predchádzajúce úvahy na Huffmanových kódoch z príkladu 3.8. Pripomíname, že sme zostrojili dva optimálne Huffmanove kódy pre rozdelenie pravdepodobností* P*, pričom kód* V *mal kódové slová dĺžok* $\{1,2,3,4,5,5\}$ *a kód* V′ *mal kódové slová dĺžok* $\{2,2,3,3,3,3\}$. *V prvom prípade bola disperzia dĺžok kódových slov* $Var(l)=\frac{20}{9}$*, v druhom* $Var(l')=\frac{2}{9}$. *V nasledujúcej tabuľke uvádzame príklad chýb v pravdepodobnostiach zdrojových symbolov, ktoré viedli k rozličným odchýlkam cien kódov.*

p_i	l_i	l'_i	e_i	$\triangle_i$	$\triangle'_i$
0.375	1	2	−0.1250	−0.1250	−0.250
0.250	2	2	0	0	0
0.125	3	3	0	0	0
0.125	4	3	0	0	0
0.0625	5	3	+0.0625	+0.3125	+0.1875
0.0625	5	3	+0.0625	+0.3125	+0.1875
			0	+0.500	+0.125

Vplyv chýb v pravdepodobnostiach symbolov na cenu Huffmanovho kódu.

Huffmanov kód je pomerne odolný voči malým odchýlkam v pravdepodobnostiach zdrojových symbolov. Ak chceme minimalizovať vplyv týchto odchýlok na cenu kódu, pri konštrukcii Huffmanovho kódu budeme zaraďovať vypočítanú pravdepodobnosť do zoznamu tak vysoko, ako sa len bude dať.

3.4 Kódovanie Markovovského zdroja

Matematický model, ktorý sma používali až doteraz na popis zdroja informácie, bol značne zjednodušený. V textoch zapísaných v prirodzenom jazyku sú medzi jednotlivými znakmi závislosti, ktoré sme doteraz zanedbávali. Napríklad v slovenčine sa po mäkkých spoluhláskach takmer nikdy nepíše ypsilon, po tvrdých spoluhláskach zasa mäkké i, v textoch sa nevyskytujú viac ako tri za sebou idúce samohlásky ani dlhé postupnosti zložené zo samotných spoluhlások a pod. Popísať však dostatočne presne takéto zákonitosti prirodzeného jazyka by bolo dosť náročné. Pre naše potreby vystačíme s omnoho jednoduchším matematickým modelom a zdroj budeme popisovať pomocou Markovovských reťazcov. Budeme predpokladať, že zdroj S v diskrétnych časovýcho okamihoch (taktoch, krokoch) generuje symboly zo zdrojovej abecedy $\Sigma_S = \{s_0, \ldots, s_{q-1}\}$; činnosť zdroja budeme popisovať pomocou postupnosti náhodných premenných $S_t,\ t = 0, \ldots,$[8] ktorá spĺňa nasledujúcu podmienku: pre ľubovoľné prirodzené číslo n a ľubovoľné čísla $i_0, \ldots, i_{n+1} \in \{0, \ldots, q-1\}$ platí

$$P(S_{n+1} = s_{i_{n+1}} | S_0 = s_{i_0}, \ldots, S_n = s_{i_n}) = P(S_{n+1} = s_{i_{n+1}} | S_n = s_{i_n}). \tag{3.29}$$

Podmienka (3.29) vyjadruje skutočnosť, že pravdepodobnosť výskytu symbolu v $(n+1)$-vom kroku závisí len od toho, aký symbol bol na výstupe zdroja v predchádzajúcom kroku n. Postupnosť náhodných premenných ktorá spĺňa podmienku (3.29) sa nazýva *Markovovský reťazec.* Analogicky, zdroj S ktorý spĺňa podmienku (3.29), budeme nazývať *Markovovským zdrojom.* Podmienené pravdepodobnosti $P(S_{n+1} = s_{i_{n+1}} | S_n = s_{i_n})$ sa nazývajú pravdepodobnosťami prechodu. Pravdepodobnosti prechodu vo všeobecnosti závisia od parametra n (znaky sa vyskytujú s inými pravdepodobnosťami napríklad v hlavičkách ako v textoch programov). Budeme však predpokladať, že pravdepodobnosti prechodu nezávisia od časového parametra n. Takýto Markovovský zdroj sa nazýva *homogénny.* Zdroj S popíšeme pomocou matice pravdepodobností prechodu. Kvôli zjednodušeniu zápisu budeme pravdepodobnosť $P(S_{n+1} = s_j | S_n = s_k)$ označovať symbolom $p_{j,k}$. Matica pravdepodobností prechodu zdroja S bude mať nasledujúci tvar:

$$M = \begin{pmatrix} p_{0,0} & p_{1,0} & \cdots & p_{q-1,0} \\ p_{0,1} & p_{1,1} & \cdots & p_{q-1,1} \\ & & & \\ p_{0,q-1} & p_{1,q-1} & \cdots & p_{q-1,q-1} \end{pmatrix}$$

Matica M má všetky prvky nezáporné a súčet prvkov v ľubovoľnom riadku je rovný 1. (Takáto matica sa nazýva stochastická.) Ak poznáme symbol, ktorý sa objavil na výstupe zdroja, pomocou matice pravdepodobností prechodu M vieme určiť pravdepodobnosti výskytu symbolov na výstupe zdroja v nasledujúcom i v ďalších časových okamihoch. Nech sa v 0-tom kroku objavil na výstupe zdroja symbol s_0. Potom sa v nasledujúcom kroku budú na výstupe zdroja objavovať symboly zo zdrojovej abecedy s pravdepodobnosťami

$$(1, 0, \ldots, 0) \times M = (p_{0,0}, p_{1,0}, \ldots, p_{q-1,0}).$$

[8] rozdelenie pravdepodobností symbolov $s_0, \ldots, s_{q-1}$ v t-tom kroku budeme označovať nasledovne: $\{p_0^{(t)}, \ldots, p_{q-1}^{(t)}\}$.

Rozdelenie pravdepodobností symbolov na výstupe zdroja v ďalšom kroku by sme vypočítali ako súčin rozdelenia pravdepodobností v 1. kroku a matice M:

$$(p_{0,0}, p_{1,0}, \ldots, p_{q-1,0}) \times M = (1, 0, \ldots, 0) \times M^2.$$

(Namiesto toho, aby sme v každom kroku prácne počítali súčin vektora a matice M, využijeme poznatok, že matica pravdepodobností prechodu po m krokoch sa rovná m-tej mocnine matice pravdepodobností prechodu po jednom kroku [14].) Ilustrujeme uvedené pojmy na príklade.

Príklad 3.10. *Uvažujeme Markovovský zdroj S so štvorprvkovou abecedou $\Sigma_S = \{a, b, c, d\}$. Vzťahy medzi symbolmi sú popísané pomocou nasledujúcej matice pravdepodobností prechodov:*

$$M = \begin{pmatrix} 0.1 & 0.4 & 0.2 & 0.3 \\ 0.5 & 0.1 & 0.2 & 0.2 \\ 0.5 & 0.2 & 0.2 & 0.1 \\ 0.6 & 0.1 & 0.2 & 0.1 \end{pmatrix}$$

Nech $p = (1, 0, 0, 0)$ je rozdelenie pravdepodobností symbolov v kroku 0. Potom rozdelenie pravdepodobností symbolov v kroku 1 bude $(0.1, 0.4, 0.2, 0.3)$. Vypočítame niekoľko mocnín matice M :

$$M^2 = \begin{pmatrix} 0.49 & 0.15 & 0.20 & 0.16 \\ 0.32 & 0.27 & 0.20 & 0.21 \\ 0.31 & 0.27 & 0.20 & 0.22 \\ 0.27 & 0.30 & 0.20 & 0.23 \end{pmatrix}$$

$$M^4 = \begin{pmatrix} 0.3933 & 0.2160 & 0.2000 & 0.1907 \\ 0.3619 & 0.2379 & 0.2000 & 0.2002 \\ 0.3597 & 0.2394 & 0.2000 & 0.2009 \\ 0.3524 & 0.2445 & 0.2000 & 0.2031 \end{pmatrix}$$

$$M^8 = \begin{pmatrix} 0.37199797 & 0.23084535 & 0.20000000 & 0.19715668 \\ 0.37092176 & 0.23159571 & 0.20000000 & 0.19748253 \\ 0.37084603 & 0.23164851 & 0.20000000 & 0.19750546 \\ 0.37059591 & 0.23182290 & 0.20000000 & 0.19758119 \end{pmatrix}$$

$$M^{16} = \begin{pmatrix} 0.3712427184 & 0.2313719296 & 0.2000000000 & 0.1973853520 \\ 0.3712414540 & 0.2313728112 & 0.2000000000 & 0.1973857348 \\ 0.3712413650 & 0.2313728732 & 0.2000000000 & 0.1973857617 \\ 0.3712410712 & 0.2313730782 & 0.2000000000 & 0.1973858506 \end{pmatrix}$$

V postupnosti matíc je vidieť istú zákonitosť—ako keby matice konvergovali k nejakej limitnej matici. Pozrieme sa na túto skutočnosť z iného hľadiska. Ako ovplyvní výskyt konkrétneho symbolu v 0-tom kroku rozdelenie pravdepodobností výskytu symbolu v n-tom kroku? V nasledujúcej tabuľke je uvedené rozdelenie pravdepodobností (náhodnej premennej) S_{16} za predpokladu, že $S_0 = a$ (b, c, d).

	p_a	p_b	p_c	p_d
$S_0 = a$	0.3712427184	0.2313719296	0.2000000000	0.1973853520
$S_0 = b$	0.3712414540	0.2313728112	0.2000000000	0.1973857348
$S_0 = c$	0.3712413650	0.2313728732	0.2000000000	0.1973857617
$S_0 = d$	0.3712410712	0.2313730782	0.2000000000	0.1973858506

Vidíme, že rozdelenie pravdepodobností symbolov v 16*-tom kroku (a zrejme ani ďalších krokoch) nezávisí od toho, aký symbol bol na výstupe zdroja v nultom kroku.*

Markovovský zdroj popísaný v príklade 3.10 je zvláštnym prípadom tzv. *ergodického Markovovského zdroja.* Definujeme ergodický Markovovský zdroj formálne.

Definícia 3.4.1. *Nech rozdelenie pravdepodobností náhodnej premennej* S_n *konverguje k limitnému rozdeleniu pravdepodobností; t.j.*

$$\lim_{n\to\infty} p_k^{(n)} = p_k; \quad k = 0, \ldots, q-1$$

a limitné rozdelenie pravdepodobností $\{p_0, \ldots, p_{q-1}\}$ *nezávisí od počiatočného rozdelenia pravdepodobností symbolov, potom sa Markovovský zdroj nazýva ergodickým Markovovským zdrojom.*

Z hľadiska kódovania nás zaujíma predovšetkým spomínané limitné rozdelenie pravdepodobností. Ak je zdroj S ergodický, tak takéto limitné rozdelenie pravdepodobností existuje a musí spĺňať nasledujúci vzťah:

$$(p_0, \ldots, p_{q-1}) \times M = (p_0, \ldots, p_{q-1}). \tag{3.30}$$

Podmienku, ktorú musí spĺňať zdroj na to, aby bol ergodický, stanovuje nasledujúca veta.

Veta 3.4.1. *(Markovova, [14]) Nech je* S *Markovovský zdroj s abecedou* $\Sigma_S = \{s_0, \ldots, s_{q-1}\}$ *a* $p_{j,k}^{(m)}$ *je pravdepodobnosť prechodu* $s_k \to s_j$ *po* m *krokoch. Ak existujú také prirodzené čísla* $t > 0$, $k_0 \geq 0$, *že* $\forall j$, $j = 0, \ldots, q-1$ *platí* $p_{j,k_0}^{(t)} > 0$; *t.j. v matici* M^t *existuje aspoň jeden stĺpec, ktorý má všetky prvky kladné, tak potom je Markovovský zdroj* S *ergodický, a teda existujú limitné pravdepodobnosti*

$$\lim_{n\to\infty} p_{j,k}^{(n)} = p_k, \quad k = 0, \ldots, q-1,$$

nezávislé na indexe j. *Postupnosť* $p_0, \ldots, p_{q-1}$ *je jediné nezáporné riešenie sústavy rovníc*

$$p_k = \sum_{j=0}^{q-1} p_j p_{j,k}, \quad k = 0, \ldots, q-1,$$

ktoré vyhovuje podmienke

$$\sum_{j=0}^{q-1} p_j = 1.$$

To znamená, že limitné rozdelenie $p_0, \ldots, p_{q-1}$ *je stacionárnym rozdelením pravdepodobností Markovovského zdroja.*

Poznámka. Stacionárne rozdelenie pravdepodobností je počiatočné rozdelenie pravdepodobností, pri ktorom majú všetky (náhodné premenné) S_n, $n = 0, \ldots$ rovnaké rozdelenie pravdepodobností.

Ak teda v matici M alebo jej niektorej nenulovej mocnine existuje stĺpec, v ktorom sú všetky prvky nenulové, potom je zdroj S ergodický a riešením sústavy (3.30) nájdeme stacionárne limitné rozdelenie pravdepodobností. Pripomíname, že matica M nie je regulárna, a preto sústavu (3.30) treba riešiť za predpokladu

$$p_0 + \cdots + p_{q-1} = 1.$$

Príklad 3.11. *Nájdeme stacionárne limitné rozdelenie pravdepodobností ergodického Markovovského zdroja S z príkladu 3.10. (Ergodickosť Markovovského zdroja S vyplýva z toho, že už v samotnej matici M sú všetky prvky kladné.) Riešime sústavu rovníc*

$$\begin{aligned} p_a &= 0.1 * p_a + 0.5 * p_b + 0.5 * p_c + 0.6 * p_d \\ p_b &= 0.4 * p_a + 0.1 * p_b + 0.2 * p_c + 0.1 * p_d \\ p_c &= 0.2 * p_a + 0.2 * p_b + 0.2 * p_c + 0.2 * p_d \\ p_d &= 0.3 * p_a + 0.2 * p_b + 0.1 * p_c + 0.1 * p_d \\ 1 &= p_a + p_b + p_c + p_d \end{aligned}$$

Riešením tejto sústavy je vektor

$$P = (p_a = 0.3712418301, p_b = 0.2313725490, p_c = 0.2000000000, p_d = 0.1973856209).$$

Poznanie limitného rozdelenia pravdepodobností možno využiť na zostrojenie Huffmanovho kódu Markovovského zdroja S. V našom prípade by Huffmanov kód bol blokový kód dĺžky 2 s cenou $\mathcal{L}(V, P) = 2$. Huffmanov kód Markovovského zdroja S však nevyužíval vzťahy medzi jednotlivými symbolmi. Navrhneme efektívnejšie kódovanie Markovovského zdroja S . Najprv zostrojíme Huffmanove kódy V_a, V_b, V_c, V_d pre rozdelenia pravdepodobností $P(\,|a)$, $P(\,|b)$, $P(\,|c)$, $P(\,|d)$. Jednotlivé kódy a ich ceny sú uvedené v tabuľke.

	a	b	c	d	$\mathcal{L}(V_x, P)$
V_a	001	1	000	01	1.9
V_b	1	001	01	000	1.8
V_c	1	01	000	001	1.8
V_d	0	100	11	101	1.6

Postupnosť symbolov $s_{i_0} s_{i_1} \ldots$ vytvorenú Markovovským zdrojom S budeme kódovať nasledovne:

1. prvý symbol, s_{i_0} zakódujeme pomocou pevne stanoveného kódu, napríklad V_{s_0};
2. j-ty symbol postupnosti zakódujeme pomocou kódu $V_{s_{j-1}}$; $j = 1, \ldots m$.

Pri dekódovaní najprv dekódujeme prvý symbol, s_{i_0}, zakódovaný pomocou kódu V_{s_0}; na základe poznania prvého symbolu (s_{i_0}) určíme kód $V_{s_{i_0}}$, ktorým je kódovaný druhý symbol, s_{i_1}, atď. Kód Markovovského zdroja budeme kvôli jednoduchosti nazývať Markovovským kódom.

Príklad 3.12. *Nech postupnosť* abababcadadca *vytvoril Markovovský zdroj z príkladu 3.10. Jeho kódovanie je popísané v nasledujúcej tabuľke.*

znak	a	b	a	b	c	a	d	a	d	c	a
použitý kód	V_a	V_a	V_b	V_a	V_b	V_c	V_a	V_d	V_a	V_d	V_c
kódové slovo	001	1	1	1	01	1	01	0	01	11	1

Na zakódovanie postupnosti dĺžky 11 *(nad štvorprvkovou abecedou) sme potrebovali* 17*-bitový reťazec.*

Cena kódu Markovovského zdroja závisí od cien čiastkových kódov V_{s_i}, $i = 0, \ldots, m-1$ a limitného rozdelenia pravdepodobností. Dá sa vypočítať na základe nasledujúceho vzťahu:

$$\mathcal{L}(M, V) = \sum_{i=0}^{q-1} p_i \cdot \mathcal{L}(P(\,|s_i), V_{s_i}).$$

Porovnáme na záver cenu Hufmannovho kódu (pre limitné rozdelenie pravdepodobností), entropiu limitného rozdelenia pravdepodobností a cenu kódu Markovovského zdroja z predchádzajúceho príkladu.

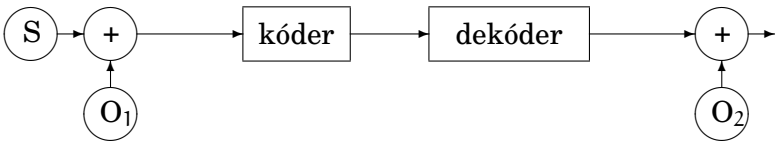
entropia limitného rozdelenia	1.945755388
cena Huffmanovho kódu	2.000000000
cena Markovovského kódu	1.797647058

Využitím závislostí medzi jednotlivými symbolmi sme dostali kód ktorého cena je výrazne nižšia ako entropia limitného rozdelenia pravdepodobností.[9]

3.5 Kódovanie pomocou orákula

Huffmanov kód využíval to, že sme poznali rozdelenie pravdepodobností symbolov v zdrojovom texte; Markovovský kód zasa vychádzal z poznania štatistických zákonitostí medzi symbolmi, ktoré nasledovali bezprostredne za sebou. Bolo by možné zostrojiť aj iné kódy, ktoré by využívali iné zákonitosti v zdrojových textoch. Uvedieme jednu všeobecnú teoretickú konštrukciu, ktorá nám umožní určiť, ako dokážeme zdrojový text stlačiť, ak poznáme akúkoľvek využiteľnú zákonitosť medzi znakmi v zdrojovom texte. Nebudeme sa snažiť túto zákonitosť bližšie špecifikovať, budeme ju charakterizovať tým, ako dobre nám umožní odhadovať, aký symbol sa objaví v nasledujúcom takte na výstupe zdoja. Podstata tejto metódy (budeme ju nazývať *kódovanie s predpoveďou*),

[9] ktorá závislosti medzi jednotlivými symbolmi nezohľadňuje.



Obr. 3.5: Kódovanie s predpoveďou

spočíva vo využití dvojice orákul[10], ktoré budú hádať, aký symbol zdroj v danom takte vytvoril. Schéma kódovania s predpoveďou je uvedená na obr. 3.5: Zdroj S generuje binárnu postupnosť $\alpha = a_0 a_1 \ldots$, orákulum O_1 sa pokúša „uhádnuť“ výstup zdroja S a generuje binárnu postupnosť $\beta = b_0 b_1 \ldots$. Obe postupnosti sa následne sčítavajú bit po bite modulo 2 a výsledná postupnosť $\{a_i \oplus b_i\}_{i\geq 0}$ vstupuje do kódera. Ak sa orákulu podarí „uhádnuť“ správne hodnotu symbolu a_i generovaného zdrojom, $a_i \oplus b_i = 0$. Postupnosť $\alpha \oplus \beta$ vstupujúca do kódera pozostáva zo súvislých postupností pozostávajúcich zo samých núl, ktoré sú oddelené jednotkami. Kóder zakóduje pozície jednotiek a pošle ich po prenosovom kanáli príjemcovi. Dekóder príjemcu transformuje kódovanú správu opäť do tvaru postupností núl oddelených jednotkami; $\{a_i \oplus b_i\}_{i\geq 0}$. Táto postupnosť vstupuje do člena realizujúceho sčítanie modulo 2, v ktorom sa sčítava s výstupom orákula O_2 generujúceho tú istú binárnu postupnosť $\beta = b_0 b_1 \ldots$ ako orákulum O_1. Výsledkom je postupnosť $\{(a_i \oplus b_i) \oplus b_i\}_{i\geq 0} = \{a_i\}_{i\geq 0}$; t.j postupnosť generovaná zdrojom S. Doplníme niektoré predpoklady a ukážeme, aké výsledky sa dajú dosiahnuť pomocou kódovania s predpoveďou. Predpokladáme, že orákulump O_1 „uhádne“ správny výsledok (jeden bit generovaný zdrojom S) s pravdepodobnosťou p a generuje opačnú hodnotu s pravdepodobnosťou $q = 1 - p$. Ďalej predpokladáme, že výsledok hádania symbolu v i-tom kroku neovplyvní výsledok hádania v ďalších krokoch.

Pozrime sa teraz na kódovanie postupnosti $\{a_i \oplus b_i\}_{i\geq 0}$. V závislosti od kvality orákula (vyjadrenej pravdepodobnosťou p) budú sa v binárnej postupnosti vyskytovať kratšie alebo dlhšie postupnosti núl ukončené jednotkami:

$$\alpha \oplus \beta = 000001001100000000000010000100000010010010100011...$$

Takúto postupnosť možno jednoznačne určiť postupnosťou prirodzených čísel, $n_0, n_1, \ldots$ vyjadrujúcich dĺžky nulových podpostupností. Pre vyššie uvedenú binárnu postupnosť $\alpha \oplus \beta$ bude postupnosť prirodzených čísel vyzerať nasledovne:

$$5, 2, 0, 12, 4, 6, 2, 2, 1, 3, 0, \ldots.$$

Existuje viacero možností kódovania postupnosti $n_0, n_1, \ldots$. Kvôli jednoduchosti použijeme na začiatok blokový kód dĺžky k. Keďže binárne slovo dĺžky k dokáže rozlíšiť 2^k hodnôt, postupnosti $0^n 1$ kde $n \geq 2^k$ sa už pomocou jedného kódového slova nedajú zakódovať. Označíme symbolom C kódovú transformáciu realizovanú kóderom, potom C

[10]Je zrejmé, že táto konštrukcia je čisto teoretická, pretože orákulum nie je prakticky realizovateľné.

možno definovať nasledovne:

$$C(0^n1) = \begin{cases} n & \text{ak } n < 2^k - 1, \\ 2^k - 1, C(0^{n-2^k+1}1) & \text{ak } n \geq 2^k - 1. \end{cases}$$

Binárna postupnosť

$$\underbrace{0\ldots0}_{2^k-1}\underbrace{0\ldots0}_{2^k-1}\underbrace{0\ldots0}_{2^k-2}1$$

bude kódovaná binárne zapísanou trojicou čísel $2^k-1, 2^k-1, 2^k-2$ dĺžky $3k$. Už z tohto jednoduchého príkladu je zrejmé, že efektívnosť kódovania s predpoveďou bude závisieť od výberu parametra k. Ukážeme, ako na základe p vybrať optimálnu hodnotu dĺžky bloku k. Postupnosti 0^j1 sa vyskytujú s pravdepodobnosťami p^jq $j = 0, 1, \ldots$. Keďže

$$\sum_{j\geq 0} p^j q = q \sum_{j\geq 0} p^j = \frac{q}{1-p} = 1,$$

množina postupností $\{0^j1\}_{j\geq 0}$ s pravdepodobnosťami $P(0^j1) = p^jq$ tvorí pravdepodobnostný priestor. Vypočítame dĺžky kódov jednotlivých postupností a potom určíme strednú hodnotu dĺžky kódovej postupnosti potrebnej na zakódovanie jednej postupnosti 0^j1. V nasledujúcej tabuľke sú uvedené dĺžky kódov postupností 0^j1 pre jednotlivé hodnoty j (označme kvôli zjednodušeniu zápisu symbolom m hodnotu $2^k - 1$):

dĺžka postupnosti	dĺžka kódu
$0 \ldots m-1$	k
$m \ldots 2m-1$	$2k$
$2m \ldots 3m-1$	$3k$

Stredná hodnotu dĺžky kódovej postupnosti potrebnej na zakódovanie jednej postupnosti 0^j1 je

$$k\left[q + pq + \cdots + p^{m-1}q\right] + 2k\left[p^mq + p^{m+1}q + \cdots + p^{2m-1}q\right] +$$
$$+3k\left[p^{2m}q + p^{2m+1}q + \cdots + p^{3m-1}q\right] + \cdots = kq\frac{1-p^m}{1-p}\left[1 + 2p^m + 3p^{2m} + \ldots\right] =$$
$$= \frac{k}{1-p^m}.$$

Stredná hodnota dĺžky postupnosti núl je

$$\sum_{k\geq 0} p^k \cdot q \cdot k = q \cdot \sum_{k\geq 0} k \cdot p^k = p/(1-p).$$

Pripočítame ešte 1 a dostávame $1/(1-p)$. Teraz spočítame kompresný pomer,

$$\kappa = \frac{k \cdot (1-p)}{1-p^m}$$

a nájdeme optimálnu hodnotu dĺžky bloku k. Keďže analytický výpočet kodnoty k, ktorá minimalizuje kompresný pomer by bol zložitý, pre známe p je jednoduchšie výpočítať hodnoty κ pre rôzne hodnoty k.

Pre zaujímavosť sme skúsili použiť namiesto blokových kódov nerovnomerný prefixový kód (neskrátený Shannonov kód) a odhadli kompresný pomer. Výsledky sú uvedené v predposlednom riadku tabuľky 3.1. Výsledky boli zrejme výrazne ovplyvnené spôsobom výpočtu dĺžky kódového slova (hornou celou časťou z prevrátenej hodnoty pravdepodobnosti výskytu postupnosti danej dĺžky). Pri konštrukcii Shannonovho kódu pre konečné rozdelenia pravdepodobností sme neraz výrazne redukovali cenu kódu skrátením niektorých slov. V nekonečnom kóde sa to z pochopiteľných dôvodov spraviť nedá. Aby sme odhadli výsledky úprav nekonečného Shannonovho kódu, v poslednom riadku tabuľky 3.1 sme uviedli odhady kompresného pomeru, pričom dĺžky kódových slov boli vyjadrené len ako prevrátené hodnoty pravdepodobnosti výskytu postupnosti danej dĺžky.

p	0.5	0.6	0.7	0.8	0.9	0.95	0.99	0.999
$k = 1$	**1.00**	**1.00**	1.00	1.00	1.00	1.00	1.00	1.00
$k = 2$	1.14	1.02	**0.913**	0.820	0.738	0.699	0.667	0.667
$k = 3$	1.51	1.23	0.980	**0.759**	0.575	0.497	0.441	0.429
$k = 4$	2.00	1.60	1.21	0.829	**0.504**	0.372	0.286	0.267
$k = 5$	2.50	2.00	1.50	1.00	0.520	0.314	0.187	0.161
$k = 6$	3.00	2.40	1.80	1.20	0.601	**0.312**	0.128	0.0984
$k = 7$	3.50	2.80	2.10	1.40	0.70	0.350	0.0971	0.0588
$k = 8$	4.00	3.20	2.40	1.60	0.80	0.400	**0.0867**	0.0356
$k = 9$	4.50	3.60	2.70	1.80	0.90	0.45	0.0905	0.0225
$k = 10$	5.00	4.00	3.00	2.00	1.00	0.5	0.100	0.0156
$k = 11$								0.0126
$k = 12$								**0.0122**
Shannon					1.1002	0.82858	0.33974	0.070041
Shannon					0.96300	0.74377	0.32288	0.067632

Tabuľka 3.1: Kompresný pomer pri kódovaní s predpoveďou

Kapitola 4

Metódy kompresie údajov

4.1 Slovníkové metódy kompresie dát

V tejto kapitole uvedieme niektoré slovníkové metódy kompresie dát. Tieto metódy sa snažia využiť na kompresiu skutočnosť, že v dátach sa častokrát opakujú rovnaké postupnosti znakov.

4.2 LZ77

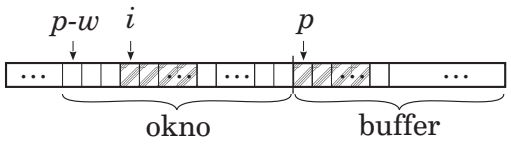
Autormi tohto algoritmu sú Lempel a Ziv (v roku 1977). Mnohé ďalšie slovníkové algoritmy boli inšpirované práve LZ77. Pri popise algoritmu budeme používať nasledujúce pojmy:

- Pozícia – aktuálna pozícia, na ktorej sa pri kódovaní/dekódovaní nachádzame. Začíname na prvom znaku a postupne pokračujeme až k poslednému znaku vstupného textu.
- Okno – posledných w spracovaných znakov (pri kompresii). Znak na pozícii sa do okna už nepočíta.
- Buffer – postupnosť znakov vo vstupnom texte začínajúca pozíciou.

4.2.1 Kompresia (kódovanie)

Označme vstupný (komprimovaný) text T a nech jeho dĺžka je n. Nech $p \in \{0, \ldots, n-1\}$ označuje pozíciu. To znamená, že okno je postupnosť (podreťazec) $T[p-w, \ldots, p-1]$ a buffer $T[p, \ldots, n-1]$. Ak $p < w$, tak je okno, prirodzene, kratšie. Hlavná myšlienka algoritmu spočíva v tom, že hľadáme najdlhší reťazec v okne, ktorý je zároveň prefixom buffra. Teda hľadáme maximálne $k \leq n-p$ také, že existuje $i \in \{p-w, \ldots, p-k\}$:

$$T[i, \ldots, i+k-1] = T[p, \ldots, p+k-1]. \tag{4.1}$$



Postup pri kompresii je nasledujúci:

1. $p = 1$
2. pokiaľ je $p < n$ opakujeme:
   (a) nájdeme i a k podľa (4.1)
   (b) výstupom je trojica
   $\langle i - (p - w) + 1, k, T[p + k]\rangle$
   (c) $p \leftarrow p + k + 1$

Prvý člen vo výstupnej trojici označuje index v okne, číslovaný pre aktuálne okno od 1, kde začína nájdený najdlhší zhodný reťazec. Tento spôsob zaručuje, že index je vždy prvok z $\{1, \ldots, w\}$. Uvedený postup nerieši niektoré situácie, ktoré pri kompresii môžu nastať. Prípad $k = 0$ nastane, ak sa v okne nenachádza znak $T[p]$. Vtedy dáme na výstup trojicu $\langle 0, 0, T[p]\rangle$. Ak najdlhší podreťazec „vyčerpá“ všetky znaky z buffra, t.j. $k = n - p$, tak zhodu skrátime o 1 a na výstup dáme trojicu $\langle i - (p - w) + 1, k - 1, T[n - 1]\rangle$.

Príklad: Nech $T =$ aababbcbababcb ($n = 14$). Tabuľka ukazuje priebeh činnosti algoritmu, pričom zhoda udáva najdlhší nájdený reťazec v okne. Vzhľadom na malý rozsah príkladu nie je veľkosť okna obmedzená.

p	T[p]	zhoda	výstup
0	a	–	$\langle 0, 0, a\rangle$
1	a	a	$\langle 1, 1, b\rangle$
3	a	ab	$\langle 2, 2, b\rangle$
6	c	–	$\langle 0, 0, c\rangle$
7	b	bab	$\langle 3, 3, a\rangle$
11	b	bcb	$\langle 6, 2, b\rangle$

4.2.2 Dekompresia (dekódovanie)

Dekóvanie je jednoduché. Podobne ako pri kódovaní si budujeme a udržiavame okno, v tomto prípade to bude posledných w znakov daných na výstup. Na začiatku je, prirodzene, prázdne. Postupne čítame trojice zo vstupu a na výstup dáme príslušný reťazec z okna (určený indexom a dĺžkou) doplnený znakom z trojice. Ak má vstupná trojica tvar $\langle 0, 0, x\rangle$, tak je výstupom samotný znak x.

4.2.3 Poznámky

LZ77 je relatívne rýchly algoritmus, pričom dekódovanie je oveľa rýchlejšie ako kódovanie, keďže nie je potrebné vyhľadávať najdlhší zhodný podreťazec a len sa „vypisuje“. Podstatnou z hľadiska rýchlosti kódovania a dosiahnutého kompresného pomeru je voľba veľkosti okna. Dlhé okno umožňuje nachádzať lepšie zhody, ale zároveň podstatne zvyšuje časovú zložitosť kódovania. Navyše, dlhšie okno zvyšuje počet bitov potrebných na zápis indexu a dĺžky vo výstupe (prvý a druhý prvok výstupnej trojice). Krátke okno naopak „zahadzuje“ potenciálne cenné informácie o prechádzajúcej podobe textu skoro, čím obmedzuje možnosť nájsť dlhšie reťazce zhody a predlžuje výstupný text. Na druhej strane krátke okno znižuje časovú zložitosť a dovoľuje použiť na zápis indexu a dĺžky menší počet bitov.

Použitie okna v LZ77 zabezpečuje, že algoritmus je orientovaný na využitie posledne videných znakov. Teda charakteristiky zo začiatku textu neberieme do úvahy. To môže byť pri niektorých typoch dát podstatnou výhodou.

LZ77 je možné rôzne modifikovať – použiť cyklické okno, variabilne meniť veľkosť okna a podobne. Medzi významnejšie úpravy patrí spracovanie výstupu ďalšími algoritmami. Dá sa očakávať, že žiadna z troch súradníc výstupnej trojice nie je rovnomerne distribuovanou náhodnou premennou. Preto môžeme použiť štatistické kódovanie (napr. Huffmanovo, aritmetické) na následnú transformáciu jednotlivých „stôp“ výstupu.

4.2.4 LZSS

Dôležitým algoritmom odvodeným z LZ77 je LZSS. LZSS rieši problém znakov, ktoré sa nevyskytujú v okne inak ako LZ77. LZSS si stanoví, akú minimálnu dĺžku musí mať reťazec zhody. Ak je nájdený reťazec kratší, tak na výstup ide samotný znak ($T[p]$) a posunieme pozíciu ďalej. Ak má reťazec dostatočnú dĺžku, dáva na výstup dvojicu $\langle i - (p - w), k\rangle$ (bez ďalšieho znaku, preto aj posun p je iný: $p \leftarrow p + k$). Teda LZSS dáva na výstup dva typy informácií. Aby ich bolo možné rozlíšiť pri dekódovaní, pridáme pred oba typy identifikačný bit.

Uveďme príklad činnosti LZSS pre vstupný text $T = \texttt{aababbcbababcb}$ (rovnaký ako v príklade pre LZ77). Minimálna požadovaná veľkosť zhody je 2.

p	$T[p]$	zhoda	výstup
0	a	–	$0, \texttt{a}$
1	a	a	$0, \texttt{a}$
2	b	–	$0, \texttt{b}$
3	a	ab	$1, \langle 1, 2\rangle$
5	b	b	$0, \texttt{b}$
6	c	–	$0, \texttt{c}$
7	b	bab	$1, \langle 2, 3\rangle$
10	a	ab	$1, \langle 1, 2\rangle$
12	c	cb	$1, \langle 6, 2\rangle$

LZSS vo všeobecnosti dosahuje lepšie kompresné pomery ako LZ77, s porovnateľ-

nými pamäťovými a časovými nárokmi. Dekódovanie je veľmi jednoduché a rýchle. Preto slúži ako základ pre ďalšie známe algoritmy – ARJ (kombinácia LZSS s Huffmanovým kódovaním), PKZip a pod. Odlišnosti iných algoritmov môžu spočívať vo veľkosti okna, v spracovaní výstupov (pozícií) a znakov štatistickým kódovaním (napr. Huffmanovým), v spôsobe posunu okna a jeho premenlivej veľkosti, v spôsobe určenia minimálnej dĺžky reťazca zhody a pod.

4.3 LZW

Autorom algoritmu je Welch (1984). LZW je v podstate vylepšením algoritmu LZ78. Špeciálna implementácia LZW sa používa na kompresiu v grafickom formáte GIF. Iný variant používa utilita `compress` v UNIXových systémoch. Algorimus si počas spracovávania vstupu buduje slovník, ktorý využíva na kódovanie. Pri popise algoritmu budeme používať nasledujúce pojmy:

- Slovník – postupnosť reťazcov
- Kódové slovo – index (pozícia) konkrétneho reťazca v slovníku

4.3.1 Kompresia (kódovanie)

Symbolom $+$ označíme zreťazenie dvoch reťazcov, teda $x + y$ označuje zreťazenie reťazcov x a y. Nech D je slovník kódových slov a nech $D(x)$ označuje kódové slovo reťazca x v slovníku D. Postup pri kódovaní je nasledujúci:

1. zaradíme všetky znaky abecedy do D
2. $p \leftarrow$ (inicializujeme p ako prázdny reťazec)
3. pokiaľ nie sme na konci vstupu:
   (a) $c \leftarrow$ ďalší znak zo vstupu
   (b) ak je $p + c$ v D:
   $p \leftarrow p + c$
   (c) inak:
   dáme na výstup $D(p)$
   pridáme $p + c$ do D
   $p \leftarrow c$
4. dáme na výstup $D(p)$

Poznamenajme, že uvedený postup nepočíta s prázdnym vstupom. Demonštrujme si postup kódovania LZW na príklade.

Príklad: Nech T = aababbcbababcb je vstupný text. Tabuľka ukazuje priebeh činnosti algoritmu, pričom stĺpec „slovník“ uvádza kódové slovo a prislúchajúci reťazec pridaný do slovníka v danom kroku výpočtu. Na začiatku obsahuje slovník tri znaky. Stĺpec pre hodnotu p udáva túto hodnotu *na konci* spracovania vstupného znaku v premennej c.

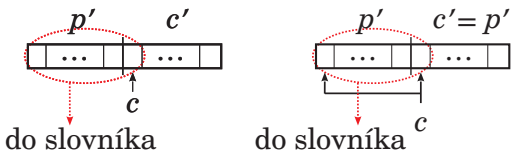
c	slovník	p	výstup
	0, a		
	1, b		
	2, c		
a	–	a	
a	3, aa	a	0
b	4, ab	b	0
a	5, ba	a	1
b	–	ab	
b	6, abb	b	4
c	7, bc	c	1
b	8, cb	b	2
a	–	ba	
b	9, bab	b	5
a	–	ba	
b	–	bab	
c	10, babc	c	9
b	–	cb	
	–		8

4.3.2 Dekompresia (dekódovanie)

Dekódovanie prebieha analogicky ako kódovanie. Konštruujeme slovník, ktorý následne používame na dekódovanie a reťazce zodpovedajúce kódovým slovám dávame na výstup. Na začiatku je opäť slovník naplnený všetkými znakmi abecedy.

Pri dekódovaní môžu nastať dva prípady: kódové slovo sa v slovníku nachádza alebo nie. Ak je vstupné kódové slovo už v slovníku, vieme dať na výstup príslušný reťazec. Zároveň vieme, že *pred* kódovaním tohto reťazca sme do slovníka zaradili reťazec, ktorý aj teraz potrebujeme do D dostať. Tento reťazec pozostáva z reťazca určeného prechádzajúcim kódovým slovom a prvým znakom reťazca určeného aktuálnym kódovým slovom.

Druhý prípad (vstupné kódové slovo nie je v slovníku) môže nastať, lebo budovanie slovníka je pri dekódovaní oneskorené. Nastane však len v tom prípade, keď pri kódovaní dáme na výstup kódové slovo, ktoré bolo do slovníka pridané ako posledné. To však znamená, že v texte sa vyskytoval za sebou dvakrát rovnaký reťazec. Preto je prvý znak reťazca prislúchajúceho vstupnému kódovému slovu zhodný s prvým znakom reťazca určeného predchádzajúcim kódovým slovom. Teda vieme, aký reťazec potrebujeme pridať do slovníka a následne aj vypísať na výstup. Nasledujúce obrázky ilustrujú oba uvedené prípady (označenia sú prebrané z nižšie uvedeného popisu algoritmu).



Pre zjednodušenie zápisu označme $D(x')$ reťazec prislúchajúci kódovému slovu x', teda opačné zobrazenie ako pri kódovaní. Postup pri dekódovaní je nasledujúci (premenné s čiarkou označujú kódové slová, bez čiarky sú to reťazce):

1. zaradíme všetky znaky abecedy do D

2. $c' \leftarrow$ prvé kódové slovo zo vstupu

3. dáme na výstup $D(c')$

4. pokiaľ nie sme na konci vstupu:

   (a) $p' \leftarrow c'$

   (b) $c' \leftarrow$ ďalšie kódové slovo zo vstupu

   (c) ak je $D(c')$ v D:
   dáme na výstup $D(c')$
   $p \leftarrow D(p')$
   $c \leftarrow$ prvý znak z $D(c')$
   pridáme $p + c$ do D

   (d) inak:
   $p \leftarrow D(p')$
   $c \leftarrow$ prvý znak z $D(p')$
   dáme na výstup $p + c$ (zhodné s $D(c')$)
   pridáme $p + c$ do D

Príklad: Nech abeceda je $\{a, b, c\}$ a nech postupnosť kódových slov $(0, 0, 1, 4, 1, 2, 8, 9, 3)$ je vstupný text. Poznamenajme, že postupnosť je na začiatku zhodná s výstupom predchádzajúceho príkladu (pre kontrolu) a na záver sa odlišuje (pre lepšiu demonštráciu postupu). Tabuľka ukazuje priebeh činnosti algoritmu pri dekódovaní. Stĺpce p, p' a c zobrazujú hodnoty premenných *na konci* spracovania vstupného kódového slova.

vstup	p'	p	c	slovník	výstup
				0, a	
				1, b	
				2, c	
0				—	a
0	0	a	a	3, aa	a
1	0	a	b	4, ab	b
4	1	b	a	5, ba	ab
1	4	ab	b	6, abb	b
2	1	b	c	7, bc	c
8	2	c	c	8, cc	cc
9	8	cc	c	9, ccc	ccc
3	9	ccc	a	10, ccca	aa

4.3.3 Poznámky

Výhoda LZW oproti LZ77 spočíva najmä v rýchlosti kódovania, pretože sa porovnáva menší počet reťazcov. Modifikácie LZW môžu zahŕňať premenlivú dĺžku zápisu kódových slov (v závislosti na aktuálnej veľkosti D), odstraňovanie starých reťazcov zo slovníka a podobne.

4.4 Aritmetické kódovanie

Aritmetické kódovanie je alternatívou k Huffmanovmu kódovaniu. Odstraňuje niektoré jeho nedostatky – oddeľuje pravdepodobnostný model zdroja od procesu kódovania (preto sa dá ľahšie upraviť na adaptívnu verziu) a nevyžaduje celý počet bitov na kódovanie každého znaku (preto je v situáciach ako $p_a = \frac{1}{10}$, $p_b = \frac{9}{10}$ výhodnejšie). Spoločnou črtou aritmetického a Huffmanovho kódovania je rovnaký model zdroja – pravdepodobnosti výskytov znakov zdrojovej abecedy (nezávislé). Keďže pri kódovaní nevyužívajú žiadne kontextové informácie (pozičné závislosti znakov), zvyknú sa označovať ako „zero-order coders“. Do tejto skupiny patrí aj Shannonov-Fanov kód.

4.4.1 Kompresia (kódovanie)

Pri kompresii najskôr určíme pravdepodobnosti výskytu jednotlivých znakov zdrojovej abecedy. Nech $\{c_1, c_2, \dots c_n\}$ je zdrojová abeceda a nech $p_1, p_2, \dots p_n$ sú príslušné pravdepodobnosti. Proporčne, podľa pravdepodobností rozdelíme interval $\langle 0, 1)$ na n častí:

$$\begin{aligned} I_1 &= \langle 0, p_1) \\ I_2 &= \langle p_1, p_1 + p_2) \\ I_3 &= \langle p_1 + p_2, p_1 + p_2 + p_3) \\ &\vdots \\ I_n &= \langle p_1 + p_2 + \cdots + p_{n-1}, 1). \end{aligned}$$

Označme $p'_k = \sum_{i=1}^{k} p_i$, pričom $p'_0 = 0$. Zároveň symbolom $|\langle d, h)|$ označíme dĺžku intervalu, t.j. hodnotu $h - d$. Algoritmus na začiatku vychádza z intervalu $I = \langle 0, 1)$. Po prečítaní prvého znaku sa interval zúži na príslušnú časť, podľa tohto znaku. Teda ak je znak na vstupe c_k, zúžime interval na I_k. Čítaním ďalších znakov naďalej interval zužujeme:

$$I = \langle d, h) \xrightarrow{c_k} \langle d + p'_{k-1}|I|, d + p'_k|I|). \tag{4.2}$$

Výstupom je ľubovoľné číslo z výsledného intervalu (najlepšie to, ktoré má najkratší zápis). Hlavná myšlienka spočíva v tom, že znaky, ktoré majú vysokú pravdepodobnosť, zužujú interval najmenej. Čím je interval menší, tým viac bitov potrebujeme na zápis niektorého z čísel, ktoré doň patria (očakávaný počet potrebných bitov je $\log_2 \frac{1}{|I|}$).

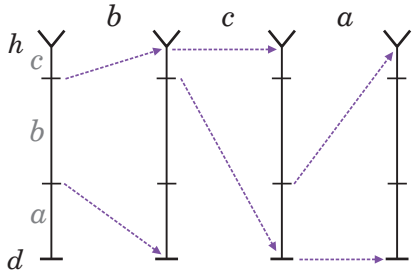
1. $I \leftarrow \langle 0, 1)$
2. pokiaľ nie sme na konci vstupu
   (a) načítame ďalší znak – c_k
   (b) upravíme interval I podľa (4.2)
3. dáme na výstup ľubovoľné číslo z intervalu I

Príklad: Nech vstupným textom je reťazec `aababbcbababcb`. Potom pravdepodobnosti výskytu jednotlivých znakov zdrojovej abecedy $\{a, b, c\}$ sú $p_a = \frac{5}{14}$, $p_b = \frac{7}{14}$ a $c = \frac{2}{14}$. Nasledujúca tabuľka ukazuje zmenu intervalu I počas spracúvania vstupu. Dolné a horné hranice sú počítané na 14 desatinných miest.

c	I
	$\langle 0\ ,\ 1)$
a	$\langle 0,00000000000000\ ,\ 0,35714285714286)$
a	$\langle 0,00000000000000\ ,\ 0,12755102040816)$
b	$\langle 0,04555393586006\ ,\ 0,10932944606414)$
a	$\langle 0,04555393586006\ ,\ 0,06833090379009)$
b	$\langle 0,05368856726364\ ,\ 0,06507705122865)$
b	$\langle 0,05775588296543\ ,\ 0,06345012494794)$
c	$\langle 0,06263666180758\ ,\ 0,06345012494794)$
b	$\langle 0,06292718435771\ ,\ 0,06333391592789)$
a	$\langle 0,06292718435771\ ,\ 0,06307244563277)$
b	$\langle 0,06297906338452\ ,\ 0,06305169402205)$
a	$\langle 0,06297906338452\ ,\ 0,06300500289792)$
b	$\langle 0,06298832749645\ ,\ 0,06300129725315)$
c	$\langle 0,06299944443076\ ,\ 0,06300129725315)$
b	$\langle 0,06300010615304\ ,\ 0,06300103256424)$

Potom napríklad číslo $0,063000679016\ldots$ je z výsledného intervalu a jeho binárny rozvoj je $0,00010000001000001101$. Teda výsledkom kódovania je uvedený dvadsaťbitový reťazec (bez 0 pred desatinnou čiarkou). Len pre porovnanie, Huffmanov kód potrebuje 21 bitov ($a = 10$, $b = 0$, $c = 11$).

Spracovanie vstupu bca ilustruje aj nasledujúci obrázok:



4.4.2 Dekompresia (dekódovanie)

Pri dekódovaní postupujeme analogicky, ako pri kódovaní. Na začiatku nastavíme interval $I = \langle 0, 1)$. Na vstupe máme číslo $x \in I$. Pre zistenie prvého znaku je potrebné určiť, v ktorom z potenciálnych n intervalov $I_1, \ldots, I_n$ sa x nachádza. Príslušný interval (povedzme I_k) určuje znak (c_k) a zároveň umožní zúžiť I ($I \mapsto I_k$).

Podobne postupujeme ďalej. Pre aktuálny interval I určujeme k také, že x je prvkom intervalu, ktorý toto k „vyrobí“ z intervalu I podľa (4.2). Na výstup dáme c_k a zúžime I.

Otázkou je, ako zistiť, že sme už dekódovali posledný znak. Možné sú dve riešenia:

1. Pridáme do abecedy špeciálny znak „EOF“ (koniec súboru) a pri dekódovaní postupujeme až dovtedy, kým tento znak nedekódujeme.
2. Spolu s výsledným číslom dáme pri kódovaní na výstup aj dĺžku kódovaného reťazca. Teda dekóder skončí po vypísaní daného počtu znakov.

4.4.3 Implementačné poznámky

Aritmetické kódovanie môžeme efektívne realizovať pomocou celočíselnej aritmetiky. Interval $\langle 0, 1)$ nahradíme intervalom celých čísel, napríklad $\langle 0x0000, 0xffff)$ (vyjadrené hexadecimálne, teda $\langle 0, 65535)$. Základné pozorovanie, ktoré umožní ostať počas celého výpočtu len v tomto intervale: ak sa začiatočné čísla (napr. bity) dolnej a hornej hranice aktuálneho intervalu zhodujú, už ostanú rovnaké. Dôvod je prostý: interval sa počas kódovania zužuje, preto zhody na začiatku sa už nezmenia. To znamená, že ak takúto zhodu zaznamenáme, môžme ju z oboch hraníc intervalu odstrániť (a dať na výstup). Napríklad, ak po úprave intervalu dostaneme $h = 0x6807$ a $d = 0x4af1$, tieto sa zhodujú na prvých dvoch bitoch (01), ktoré dáme na výstup a následne upravíme hranice – $h = 0xa01f$, $d = 0x2bc4$. Hornú hranicu dopĺňame jednotkami a dolnú nulami. Prirodzene, dekóder musí pri práci s intervalmi aplikovať rovnaký postup.

Problém môže nastať, ak pri zužovaní intervalu dostávame (binárne) $h = 1000zzz$ a $d = 0111www$. V takomto prípade nielen nevieme čo dať na výstup (kam sa nakoniec „preklopí“ najvyšší bit), ale aj strácame presnosť (dĺžku intervalu). Riešenie spočíva v tom, že v týchto situáciach odstránime úvodné nulové bity z h a úvodné jednotkové z d:

$h = 1zzz$ a $d = 0www$. Popritom si zapamätáme počet takto odstránených bitov a keď sa najbližšie zhodnú najvyššie bity hraníc, budeme vedieť, koľko a akých bitov dať na výstup.

4.4.4 Poznámky

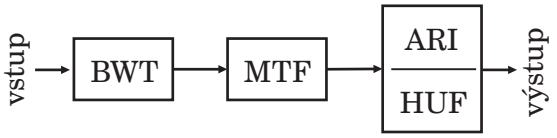
Aritmetické kódovanie, podobne ako Huffmanovo sa zvyčajne nepoužíva samostatne, ale vystupuje ako súčasť zložitejších kompresných algoritmov (najčastejšie ako záverečná fáza). Môžeme ho kombinovať so slovníkovými metódami (napr. LZARI je kombinácia LZSS a aritmetického kódovania), v štatistických metódach vyšších rádov (napr. PPM) aj v iných metódach (pozri napr. časť 4.5).

Problémom aritmetického kódovania je rýchlosť – ktorá nie je príliš veľká. Kompresný pomer je zvyčajne o čosi lepší ako pri Huffmanovom kódovaní (ale nie o veľa). Jednoduché je modifikovať aritmetické kódovanie na adaptívnu verziu. Jednoducho začneme s rovnomerne distribuovanými pravdepodobnosťami a každý načítaný znak zo vstupu najskôr spracujeme (zúžime interval), a potom príslušne upravíme pravdepodobnosti (teda zväčšíme početnosť tohto znaku). Samozrejme, aj Huffmanovo kódovanie je možné upraviť na adaptívne, avšak nie tak priamočiaro.

Adaptívne verzie aritmetického alebo Huffmanovho kódovania majú výhodu v tom, že nie je potrebné prenášať frekvenčnú tabuľku znakov a pri kódovaní nemusíme čítať vstup dvakrát (najskôr na zistenie frekvenčnej tabuľky a potom na samotné kódovanie).

4.5 BWT

BWT (Burrows-Wheeler Transformation) v podstate nie je algoritmus na kompresiu dát. Je to invertovateľná transformácia, ktorá reťazec znakov transformuje na iný reťazec znakov. Výstupný reťazec je potom vhodnejší na kompresiu ako pôvodný reťazec. Metóda kompresie dát využíva BWT ako úvodnú transformáciu, nasledovanú napríklad MTF (pozri časť 4.5.3) a štatistickým kóderom nultého rádu tak, ako je to zobrazené na obrázku. Prirodzene, možné sú aj ďalšie modifikácie.



4.5.1 Kódovanie

Transformácia pracuje nad blokom dát (reťazcom znakov) dĺžky n. Vytvoríme z reťazca n reťazcov dĺžky n tak, že vstupný reťazec rotujeme. Získané reťazce utriedime. Výstupom transformácie je reťazec pozostávajúci z posledných znakov v reťazcoch (teda ak

prejdeme v poradí utriedenia po reťazcoch a vypíšeme ich posledné znaky) a z pozície pôvodného vstupného reťazca medzi utriedenými reťazcami.

Hlavná myšlienka BWT spočíva v tom, že rovnaké kontexty (podreťazce vstupu) sú zvyčajne uvádzané rovnakými znakmi (je to podobná úvaha ako pri znakoch za kontextami). Rovnaké kontexty dáme k sebe triedením. Znaky, ktoré sú pred týmito kontextami sú na konci reťazcov. Teda na konci reťazcov môžeme očakávať častý výskyt rovnakých znakov vedľa seba.

Príklad: Nech vstupným textom je reťazec `aababbcbababcb`. Potom utriedenie jednotlivých rotácií dopadne takto (`i` označuje pozíciu začínajúceho znaku v pôvodnom reťazci):

i	reťazec
0	`aababbcbababcb`
1	`ababbcbababcba`
8	`ababcbaababbcb`
3	`abbcbababcbaab`
10	`abcbaababbcbab`
13	`baababbcbababc`
7	`bababcbaababbc`
2	`babbcbababcbaa`
9	`babcbaababbcba`
4	`bbcbababcbaaba`
11	`bcbaababbcbaba`
5	`bcbababcbaabab`
12	`cbaababbcbabab`
6	`cbababcbaababb`

Výstupom z BWT je v tomto prípade reťazec `babbbccaaaabbb` a pozícia, na ktorej sa nechádza pôvodný reťazec, teda `0`.

4.5.2 Dekódovanie

Pri dekódovaní máme k dispozícii reťazec zložený z posledných znakov utriedených reťazcov a index, kde treba hľadať pôvodný reťazec. Modelujme dekódovanie na tabuľke, akú sme použili v príklade kódovania. Teda poznáme posledný stĺpec znakov. Poznáme aj prvý stĺpec, stačí znaky len utriediť.

Pozrime sa na posledný znak v prvom riadku (inými slovami prvý znak v poslednom stĺpci). Nech je to x. Vieme, že toto x je ten istý znak, ktorý sa ako prvé x vyskytne v prvom stĺpci. Dôvod je ten, že za ním v reťazci nasleduje lexikograficky najmenší reťazec (inak by nebol v prvom riadku) a najmenší reťazec spomedzi ostatných začínajúcich x musí nasledovať aj za znakom, ktorý je prvým výskytom x v prvom stĺpci (inak by to nebol prvý výskyt). Túto pozíciu x v prvom stĺpci si označme ako obsadenú.

Zoberieme druhý znak v poslednom stĺpci, nech je to y. Opäť hľadáme v prvom stĺpci prvý neobsadený výskyt znaku y. Postupujeme takto ďalej, až kým neurčíme pre všetky

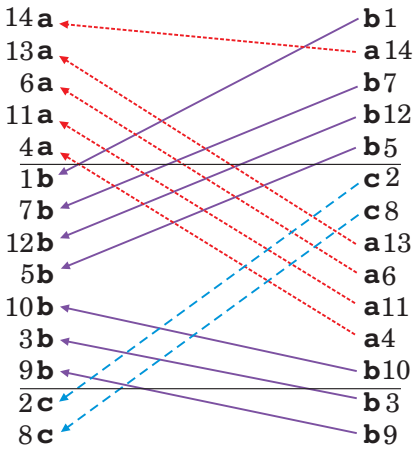
znaky z posledného stĺpca, ktorým znakom z prvého stĺpca zodpovedajú.

Teraz rekonštruujeme reťazec v prvom riadku. Keďže vieme, že posledný znak v riadku je v reťazci pred prvým znakom v riadku, dokážeme spätne prejsť a odzadu rekonštruovať požadovaný reťazec. Potom ho stačí len zrotovať, utriediť a vybrať výstupný reťazec zo správnej pozície.

Drobný problém v prezentovanej rekonštrukcii by mohol nastať, ak sú prvý znak a posledný v prvom riadku zhodné. Potom ale celý reťazec obsahuje len tento znak a môžeme sa podľa toho zariadiť.

Poznamenajme, že v praktickej implementácii BWT sa dekódovanie dá robiť na jeden prechod v lineárnom čase $O(n)$.

Príklad: Ilustrujme dekódovanie na výstupe príkladu kódovania, teda máme na vstupe reťazec `babbbccaaaabbb` a pozíciu `0`. Rekonštruujeme prvý stĺpec a potom dekódovanie prebieha naznačeným spôsobom podľa šípiek (šípky označujú zodpovedajúcim si znakom). Čísla pri znakoch hovoria o poradí znakov pri spätnej rekonštrukcii.



4.5.3 MTF

MTF (Move to front) je heuristika, ktorou sa snažíme pozičnú blízkosť rovnakých znakov pretransformovať do štatistickej významnosti znakov. MTF nekomprimuje text, ale vytvára predpoklady na úspešnú aplikáciu štatistických kóderov nultého rádu tým, že sa snaží znižovať entropiu.

MTF má pole, v ktorom sú usporiadané znaky. Po prečítaní znaku dá na výstup index (pozíciu) tohto znaku v poli a zároveň znak presunie v poli na začiatok. Takto pokračuje, až kým nevyčerpá celý vstup.

Príklad: Demonštrujme si MTF na príklade výstupu z BWT, teda na reťazci `aababbcbababcb`. Stĺpec „pole“ ukazuje poradie prvkov v poli *po* spracovaní príslušného znaku.

	pole	výstup
	abc	
b	bac	1
a	abc	1
b	bac	1
b	bac	0
b	bac	0
c	cba	2
c	cba	0

pokr.

	pole	výstup
a	acb	2
a	acb	0
a	acb	0
a	acb	0
b	bac	2
b	bac	0
b	bac	0

Hoci náš príklad nie je ideálnym na demonštráciu výhod MTF, porovnajme entropie pôvodného a nového reťazca:

$$H\left(\frac{5}{14}, \frac{7}{14}, \frac{2}{14}\right) \approx 0,4371$$
$$H\left(\frac{8}{14}, \frac{3}{14}, \frac{3}{14}\right) \approx 0,4318$$

Teda MTF sa snaží posúvať aktuálne spracúvané znaky na začiatok poľa a zabezpečiť častý výskyt nízkych indexov vo výstupe. Keď vo vstupe prejdeme na iný blok rovnakých znakov, až na prvý index opäť dostávame na výstup nízke hodnoty. Výsledkom je zníženie entropie a môže nasledovať úspešná aplikácia štatistického kódovania.

Dekódovanie MTF prebieha podobne ako kódovanie. Začneme s utriedeným poľom znakov. Prečítame index zo vstupu. Príslušný znak z poľa dáme na výstup. Zároveň presunieme znak na začiatok poľa a načítame ďalší index zo vstupu. Toto opakujeme, až kým neprečítame celý vstup.

4.5.4 Poznámky

Iná možnosť pri spracovaní výstupu BWT (namiesto MTF, prípadne navyše k MTF) je použiť RLE. RLE (Run Length Encoding) je jednoduchý spôsob kódovania, keď namiesto reťazca rovnakých znakov dávame na výstup len jeden znak a dĺžku reťazca.

Časovo najnáročnejšou operáciou v BWT je triedenie pri kódovaní. Dá sa napríklad použiť kombinácia radix-sortu (napr. na prvé dva znaky) s následným quicksortom (na utriedenie vnútri skupín). Prirodzene, pri kódovaní nie je ani potrebné vytvárať ďalšie reťazce – stačí správne indexovať do pôvodného reťazca.

Príkladom praktického použitia BWT je program Bzip2, ktorý je kombináciou BWT, MTF a Huffmanovho kódovania.

Kapitola 5

Kódovanie zvuku a obrazu

V tejto kapitole budeme hovoriť o efektívnych metódach kódovania obrazovej informácie a zvuku (hudby).

Kapitola 6

Kolmogorovská zložitosť a hranice kompresie údajov

Pozrieme sa, ako to vyzerá so stlačiteľnosťou informácie vo všeobecnosti a ukážeme, že väčšina údajov je nestlačiteľná.

Časť II

Samoopravné kódy

Kapitola 7

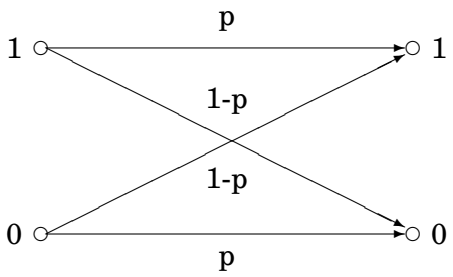
Základné princípy samoopravných kódov

Pri konštrukcii nerovnomerných kódov v predchádzajúcej kapitole sme predpokladali, že prenosový kanál realizuje identickú transformáciu; t.j. že sa správa pri prenose nemení. Tento optimistický predpoklad nebýva v reálnom živote naplnený. Preto sa budeme zaoberať takým zápisom informácie, ktorý umožní kontrolovať zmeny, ku ktorým došlo v priebehu prenosu informácie.[1] Najprv zavedieme model prenosového kanála, ktorý nám umožní matematicky popísať výskyt chýb pri prenose správ. Potom ukážeme, ako možno pridaním doplnkovej informácie (zväčšením redundancie správ) zvýšiť jej odolnosť voči chybám. Na geometrickom modeli kódu ukážeme, aké podmienky musí spĺňať kód, ktorý má odhaľovať/opravovať istý počet chýb a uvedieme základný vzťah medzi redundanciou a opravnou schopnosťou kódu. Potom budeme konštruovať jednoduché rovnomerné (blokové) kódy, ktoré budú schopné odhaľovať chyby (t.j. príjemca bude schopný zistiť, či v prijatom slove vznikli určité chyby alebo nie) alebo ich dokonca opravovať (príjemca bude pri dekódovaní schopný rekonštruovať pôvodne odvysielané kódové slovo) a popíšeme efektívne metódy kódovania a dekódovanie informácie pomocou takýchto kódov.

7.1 Binárny symetrický kanál bez pamäte

Na rozdiel od nerovnomerných kódov nebudeme skúmať zdroj informácie, ale prenosový kanál. Na popis prenosového kanála, na ktorý pôsobí šum spôsobujúci chyby v prenášaných správach zavedieme model, ktorý budeme nazývať q-nárnym symetrickým prenosovým kanálom bez pamäte. Špeciálnym a najčastejšie používaným q-nárnym symetrickým prenosovým kanálom bez pamäte je binárny $(q = 2)$ symetrický kanál bez pamäte, ktorý je zobrazený na obrázku 7.1. Upresníme teraz predstavu o tom, aké chyby môžu vznikať pri prenose správ q-nárnym symetrickým prenosovým kanálom bez pamäte. Budeme predpokladať, že pri prenose správ

[1] Pripomíname, že z hľadiska kódovania nie je principiálny rozdiel, či ide o prenos informácie v čase alebo v priestore. Aj preto sa v tejto kapitole budeme zaoberať kódovaním informácie pre prenášanie v priestore, ale riešenia, ktoré navrhneme budú rovnako dobré aj pre ochranu informácie prenášanú v čase.



Obr. 7.1: Binárny symetrický kanál bez pamäte

- dochádza k zámene jedného prenášaného symbolu kódovej abecedy na iný symbol kódovej abecedy,

- žiaden symbol nie je odolnejší voči chybe ako iný symbol; symbol sa prenáša správne s pravdepodobnosťou p a transformuje sa pri prenose na ktorýkoľvek iný symbol s pravdepodobnosťou $\frac{1-p}{q-1}$;

- výsledok prenosu jedného symbolu neovplyvňuje to, či bude nasledujúci symbol prenesený správne alebo nie.

Kódy opravujúce chyby predstavujú rozličné algebraické štruktúry ako vektorové priestory, okruhy polynómov, ideály a podobne. Aby mohli kódové slová bez problémov tvoriť takéto algebraické štruktúry, budeme predpokladať, že kódová abeceda je podmnožinou prirodzených čísel; $\Sigma = \{0, \ldots, q-1\}$. Aj keď teoretické konštrukcie budeme robiť pre všeobecný prípad, v ďalšom výklade sa budeme najčastejšie zaoberať binárnymi kódmi, ktoré sa v súčasnosti najčastejšie používajú.

V čom je podstata kódov odhaľujúcich, resp. opravujúcich chyby? Ak by sme na kódovanie správ používali úplné kódy, pri prenose správ by sa jedno kódové slovo mohlo v dôsledku šumu nahradiť iným kódovým slovom a príjemca by mal problém určiť, či prijal odvysielané kódové slovo, alebo došlo k chybe pri prenose. Preto nie je možné pri komunikácii prostredníctvom kanála so šumom používať úplné kódy. Podstata kódov odhaľujúcich a opravujúcich chyby je v tom, že množina kódových slov tvorí len podmnožinu všetkých možných slov a tak, keď dôjde počas prenosu správy ku chybe, prijaté slovo s veľkou pravdepodobnosťou nie je kódovým slovom. Zdôrazňujeme slová *s veľkou pravdepodobnosťou*, pretože nie je vylúčené, že počas prenosu vznikne chyba, ktorá prenášané kódové slovo transformuje na iné kódové slovo. Pri konštrukcii samoopravných kódov sa snažíme minimalizovať pravdepodobnosť takejto možnosti. Vychádzame z toho, že pre (binárny) prenosový kanál platí $p >> 1 - p$ [2]; t.j. je pravdepodobnejšie, že pri prenose kódového slova vznikne menej chýb. Ilustrujeme to na príklade.

Príklad. Uvažujme binárny symetrický kanál bez pamäte s parametrami $p = 0.99$, $1 - p = 0.01$, binárny blokový kód dĺžky 15 opravujúci tri chyby. V nasledujúcej tabuľke sú uvedené pravdepodobnosti chýb

[2] V podstate však stačí, aby $p \neq 1 - p$.

počet chýb	pravdepodobnosť	
0	$(0.99)^{15}$	0.860058354641289
1	$15 \cdot (0.99)^{14} \cdot (0.01)$	0.130311871915347
2	$\binom{15}{2} \cdot (0.99)^{13} \cdot (0.01)^2$	0.009213970741489
3	$\binom{15}{3} \cdot (0.99)^{12} \cdot (0.01)^3$	0.000403305116631
> 3	$\sum_{j>3} \binom{15}{j} \cdot (0.99)^{15-j} \cdot (0.01)^j$	0.000012497585244

Pravdepodobnosť toho, že v prenášanom slove vzniknú 4 a viac chýb je síce nenulová 0.000012497585244 ale podstatne menšia ako to, že v slove budú najviac 3 chyby, ktoré sa pri dekódovaní dajú opraviť.

7.2 Geometrická interpretácia samoopravného kódu

Skôr ako pristúpime k popisu a konštrukcii samoopravných kódov, využijeme geometrickú interpretáciu kódu a vysvetlíme princíp kódov opravujúcich a odhaľujúcich chyby. Predpokladáme, že máme zostrojiť kód dĺžky n opravujúci t chýb. (Na začiatku kvôli zjednodušeniu popíšeme konštrukciu binárneho kódu opravujúceho 1 chybu a potom konštrukciu zovšeobecníme.) Zavedieme najprv dva dôležité pojmy, ktoré budeme pri koštrukcii samoopravného kódu potrebovať.

Definícia 7.2.1. *Nech sú* $\mathbf{u}, \mathbf{v}$ *dva vektory vektorového priestoru* V*; nech* $\mathbf{u} = (u_1, \dots u_n)$*;* $\mathbf{v} = (v_1, \dots v_n)$*. Hammingovou váhou vektora* $\mathbf{u}$ *nazveme prirodzené číslo* $\mathbf{wt}(u)$*, definované nasledovne:*

$$\mathbf{wt}(u) = \sum_{i}^{n} (u_i \neq 0)$$

Hammingovou vzdialenosťou vektorov $\mathbf{u}, \mathbf{v}$ *nazveme prirodzené číslo* $\mathbf{d}(\mathbf{u}, \mathbf{v})$*;*

$$\mathbf{d}(\mathbf{u}, \mathbf{v}) = \mathbf{wt}(u - v) = \sum_{j=1}^{n} (a_j \neq b_j).$$

Hammingova váha vektora je počet jeho nenulových zložiek a Hammingova vzdialenosť dvoch vektorov udáva, v koľkých zložkách sa tieto dva vektory odlišujú. Vráťme sa teraz ku konštrukcii binárneho samoopravného kódu opravujúceho 1 chybu[3]. Zostrojíme ho tak, že budeme postupne vyberať kódové slová. Ako prvé kódové slovo $\mathbf{v_0}$ môžeme vybrať ľubovoľný binárny vektor z množiny $\{0, 1\}^n$. Bez ujmy na všeobecnosti môžeme vybrať $\mathbf{v_0} = (0, \dots, 0)$; t.j. nulový vektor. Vyberieme teraz druhé kódové slovo $\mathbf{v_1}$. Predpokladajme, že $\mathbf{d}(\mathbf{v_0}, \mathbf{v_1}) = 1$ a položme $\mathbf{v_1} = (1, 0, \dots, 0)$. Potom však existuje chyba (ktorú budeme reprezentovať binárnym vektorom) váhy 1 ktorá transformuje kódové slovo $\mathbf{v_1}$ na kódové slovo $\mathbf{v_0}$:

[3]Vzhľadom na typ chýb, ktorými sa budeme zaoberať, budeme pojmy „t chýb“ a „chyba váhy t“ používať ako synonymá.

$$\begin{array}{ll} \mathbf{v_0} & 000\ldots0 \\ \mathbf{v_1} & 100\ldots0 \\ \mathbf{e_1} & 100\ldots0 \\ \mathbf{v_1}\oplus\mathbf{e_1} & 000\ldots0 \quad = \mathbf{v_0} \end{array}$$

To znamená, že ak kód opravuje jednu chybu, tak potom $\mathbf{d}(\mathbf{v_0},\mathbf{v_1}) > 1$. Nech $\mathbf{d}(\mathbf{v_0},\mathbf{v_1}) = 2$ a vyberme ako druhé kódové slovo napríklad $\mathbf{v_1} = (1,1,0,\ldots,0)$. Žiadna chyba váhy 1 nemôže transformovať slovo $\mathbf{v_1}$ na slovo $\mathbf{v_0}$. Čo sa však stane, ak sme prijali slovo $0100\ldots0$? Existujú dve rovnako pravdepodobné možnosti (a množstvo menej pravdepodobných iných):

$$\begin{array}{ll} \mathbf{v_0} & 000\ldots0 \\ \mathbf{v_1} & 110\ldots0 \\ \mathbf{e_1} & 100\ldots0 \\ \mathbf{e_2} & 010\ldots0 \\ \mathbf{v_0}\oplus\mathbf{e_1} = & 100\ldots0 \quad = \mathbf{v_1}\oplus\mathbf{e_2}. \end{array}$$

Ak sme teda prijali slovo $0100\ldots0$, je zrejmé, že to nie je kódové slovo a odhalili sme chybu, ale nevieme ju opraviť a určiť odvysielané kódové slovo. Ak stačí, aby kód odhaľoval chyby váhy 1, vektor $\mathbf{v_1} = (110\ldots0)$ môže byť kódovým slovom. Ak požadujeme, aby kód opravoval chyby váhy $t \geq 1$, vektor $\mathbf{v_1} = (110\ldots0)$ a žiaden vektor váhy 2 nemôže byť kódovým slovom. Nech teda $\mathbf{d}(\mathbf{v_0},\mathbf{v_1}) = 3$ a $\mathbf{v_1} = (1,1,1,0,\ldots,0)$. Chybou váhy 1 sa slovo $\mathbf{v_1}$ transformuje v najhoršom prípade na slovo váhy 2, ale chybou váhy 1 sa zo slova $\mathbf{v_0}$ stane vektor váhy 1:

$$\begin{array}{lll} \mathbf{v_0} & 0000\ldots0 & \\ \mathbf{v_1} & 1110\ldots0 & \\ \mathbf{e_1} & 1000\ldots0 & \\ \mathbf{e_2} & 0100\ldots0 & \\ \mathbf{e_3} & 0010\ldots0 & \\ \mathbf{v_0}\oplus\mathbf{e_1} = & 1000\ldots0 & \mathbf{wt}(v_0\oplus e_1) = 1 \\ \mathbf{v_0}\oplus\mathbf{e_2} = & 0100\ldots0 & \mathbf{wt}(v_0\oplus e_1) = 1 \\ \mathbf{v_0}\oplus\mathbf{e_3} = & 0010\ldots0 & \mathbf{wt}(v_0\oplus e_1) = 1 \\ \mathbf{v_1}\oplus\mathbf{e_1} = & 0110\ldots0 & \mathbf{wt}(v_1\oplus e_1) = 2 \\ \mathbf{v_1}\oplus\mathbf{e_2} = & 1010\ldots0 & \mathbf{wt}(v_1\oplus e_2) = 2 \\ \mathbf{v_1}\oplus\mathbf{e_3} = & 1100\ldots0 & \mathbf{wt}(v_1\oplus e_3) = 2. \end{array}$$

Ak sme prijali slovo $1000\ldots0$, dekódujeme ho na základe toho, že

$$P(1000\ldots0|\mathbf{v_0}) > P(1000\ldots0|\mathbf{v_1}),$$

ako $\mathbf{v_0}$. (Ak použijeme hodnoty $n = 15, p = 0.99$ z predchádzajúceho príkladu, tak

$$P(1000\ldots0|\mathbf{v_0}) = 0.00868745812768978 > 0.000087752102299 8968 = P(1000\ldots0|\mathbf{v_1}),$$

a teda pravdepodobnosť toho, že bolo odvysielané slovo $\mathbf{v_0}$ je podstatne väčšia.) Zovšeobecníme teraz našu konštrukciu na prípad, keď má kód opravovať chyby váhy $t > 1$. Ukázalo sa, že rozhodujúcim parametrom, od ktorého závisí opravná schopnosť kódu je

minimálna vzdialenosť kódových slov. Zavedieme pre tento pojem špeciálne označenie: minimálnou vzdialenosťou kódu $\mathcal{C}$ nazveme prirodzené číslo

$$d^* = \min_{\mathbf{u},\mathbf{v}\in\mathcal{C}} \mathbf{d}(\mathbf{u},\mathbf{v}).$$

Ak by bola minimálna vzdialenosť kódu $\mathcal{C}$ $d^* \leq t$ tak potom chybou váhy menšej alebo rovnej t by sa mohlo transformovať jedno kódové slovo na iné kódové slovo. Ak $d^* = t+1$, kód $\mathcal{C}$ dokáže odhaľovať chyby váhy t. Na to, aby kód $\mathcal{C}$ opravoval chyby váhy t musí byť $d^* \geq 2t+1$.

Popri samoopravnej schopnosti (danej minimálnou vzdialenosťou kódu) je zaujímavou kvantitatívnou charakteristikou kódu, ktorá vyjadruje jeho efektívnosť, počet kódových slov. Extrémnym prípadom je kód dĺžky $2n+1$ ktorý má dve kódové slová (napr. $0\ldots0$ a $1\ldots1$). Tento kód má síce maximálnu opravnú schopnosť (je schopný opravovať n chýb), ale na prenos jedného bitu správy potrebuje $2n+1$ kódových symbolov. Pri konštrukcii samoopravných kódov sa snažíme o kompromis medzi opravnou schopnosťou a mohutnosťou kódu. Koľko kódových slov môže vlastne obsahovať samoopravný kód dĺžky n opravujúci t chýb? Pozrieme sa najprv na binárny prípad. Množina binárnych vektorov (neskôr ukážeme, že sa jedná o vektorový priestor), z ktorej vyberáme kódové slová, má 2^n prvkov. Označíme symbolom $B_r(\mathbf{v})$ množinu vektorov;

$$B_r(\mathbf{v}) = \{\mathbf{u} | \mathbf{u} \in \{0,1\}^n \& \ \mathbf{d}(\mathbf{u},\mathbf{v}) \leq r\},$$

ktorú budeme nazývať guľou s polomerom r a stredom $\mathbf{v}$. Je zrejmé, že pre $0 \leq r \leq t$ platí

$$\forall \mathbf{u},\mathbf{v} \in \mathcal{C};\ (\mathbf{u} \neq \mathbf{v}) \Rightarrow B_r(\mathbf{u}) \bigcap B_r(\mathbf{v}) = \emptyset$$

a mohutnosť $B_r(\mathbf{v})$ je

$$|B_r(\mathbf{v})| = \sum_{j=0}^{r} \binom{n}{j}.$$

Ak by kód $\mathcal{C}$ mal maximálny počet kódových slov (pre dĺžku kódu n a opravnú schopnosť t), potom by množina vektorov $\{0,1\}^n$ musela byť pokrytá disjunktnými guľami polomeru t so stredami v kódových slovách. Mohutnosť kódu $\mathcal{C}$ by v takomto prípade bola

$$|\mathcal{C}| = \frac{2^n}{\sum_{j=0}^{t}\binom{n}{j}}.$$

Je zrejmé, že je len málo takých hodnôt n,t pre ktoré by bol podiel $\frac{2^n}{|S(\mathbf{v},r)|}$ celočíselný. Ak by sme sa však aj uspokojili s kódom menšej mohutnosti, zostáva otázkou, ako ho zostrojiť. Úplné preberanie neprichádza do úvahy, nakoľko jeho zložitosť je odvodená od čísla

$$\binom{2^n}{\lfloor \frac{2^n}{\sum_{j=0}^{t}\binom{n}{j}} \rfloor},$$

ktoré je už pre relatívne malé hodnoty n,t veľké (pozri nasledujúcu tabuľku). V tejto kapitole sa budeme zaoberať metódami systematického vytvárania samoopravných kódov.

n	t	mohutnosť kódu	počet možných kódov
7	1	16	93343021201262177400
7	2	4	10668000
7	3	2	8128
15	5	6	1718574240691455027134464
15	4	16	84137321239748052363790529051765801371652428817494731388928
15	3	56	$0.9837970552 \times 10^{178}$
15	2	270	$0.7332302377 \times 10^{678}$
15	1	2048	$0.1094851418 \times 10^{3326}$

Poznámka. Aká bude mohutnosť samoopravných kódov nad inou ako binárnou abecedou? Mohutnosť gule s polomerom t vo vektorovom priestore $\{0, \ldots, q-1\}$ je

$$\sum_{j=0}^{t} \binom{n}{j} (q-1)^j.$$

Ak $q > 2$ nestačí len vybrať j zložiek vektora ($\binom{n}{j}$), ktoré treba zmeniť, ale je potrebné aj určiť ktorým z ostatných $q-1$ symbolov sa má pôvodný symbol nahradiť. Pre mohutnosť q-kódu V dĺžky n, opravujúceho t chýb platí

$$|V| \leq \frac{q^n}{\sum_{j=0}^{t} \binom{n}{j}(q-1)^j}.$$

Vráťme sa ešte ku geometrickému modelu samoopravných kódov, aby sme zaviedli niekoľko dôležitých pojmov, ktoré majú názornú geometrickú interpretáciu. Predstavme si, že je daný q-nárny samoopravný kód V dĺžky n opravujúci t chýb. Kódu V zodpovedá množina bodov-vektorov vektorového priestoru $GF(q)^n$. Vytvoríme gule polomeru 1 so stredmi v kódových slovách („kódové“ gule). Postupne budeme zväčšovať polomery „kódových“ gulí: $r = 2, 3, \ldots, t-1$. Prvá dôležitá hodnota polomeru je t. To je maximálna hodnota, pri ktorej sú sféry so stredmi v kódových slovách ešte disjunktné. Hodnota t sa nazýva aj *hranicou sférického uloženia kódu (sphere packing bound)*. Pri dekódovaní sa všetky vektory z gule $B_t(\mathbf{u})$ zobrazia na vektor $\mathbf{u}$. Vo väčšine kódov gule polomeru t so stredmi v kódových slovách nepokryjú všetky vektory vektorového priestoru $GF(q)^n$. To znamená, že budú existovať slová-vektory, ktoré nepatria do žiadnej gule a pri dekódovaní sa nezobrazia na žiadne kódové slovo. Ak by sme ďalej zväčšovali polomer gulí ($r = t+1, \ldots, T$), po konečnom počte krokov dospejeme do štádia, keď každý vektor vektorového priestoru $GF(q)^n$ patrí aspoň do jednej gule $B_T(\mathbf{u})$. Minimálna hodnota polomeru, pri ktorej sa dosahuje, že

$$GF(q)^n = \bigcup_{\mathbf{u} \in V} B_T(\mathbf{u})$$

sa nazýva *hranicou pokrytia kódu* V. Teraz môžeme formalizovať aj intuitívne predstavy o efektívnosti kódovania.

Definícia 7.2.2. *Kód* V *sa nazýva dokonalý, ak sa hranica sférického uloženia rovná hranici pokrytia kódu* V.

Ináč povedané, kód V je dokonalý, ak každý vektor vektorového priestoru $GF(q)^n$ sa nachádza vo vzdialenosti nanajvýš t od práve jedného kódového slova. Ako sa ukáže neskôr, dokonalých kódov je málo.

Skôr ako sa budeme zaoberať systematicky metódami konštrukcie rozličných samoopravných kódov, uvedieme niekoľko jednoduchších príkladov kódov opravujúcich alebo odhaľujúcich chyby a ilustrujeme na nich už zavedené, resp. zavedieme niektoré nové pojmy. Odteraz sa až do odvolania budeme opäť zaoberať binárnymi kódmi.

7.3 Jednoduché kódy odhaľujúce/opravujúce chyby

7.3.1 Testovanie parity

Nech je daná množina binárnych vektorov dĺžky n. Pridáme ku každému vektoru $(n+1)$-vý bit tak aby počet jednotkových bitov vo vektore (dĺžky $n+1$) bol párny. Kódové slová budú potom vyzerať nasledovne (doplnený bit je oddelený medzerou):

$$\begin{array}{ll} 01000011100001010 & 0 \\ 01011010010101011 & 1 \\ \multicolumn{2}{c}{\ldots} \end{array}$$

Doplnený bit sa nazýva *paritným bitom.* Ak v kódovom slove vznikne pri prenose chyba nepárnej váhy $(1,3,5,\ldots)$ počet jednotkových bitov v prijatom slove bude nepárny a príjemca bude vedieť, že nastala chyba (aj keď nedokáže určiť, kde.) Ak by však pri prenose nastala chyba párnej váhy $(2,4,\ldots)$, v prijatom slove bude párny počet jednotkových bitov a príjemca bude prijaté slovo považovať za kódové slovo. Ak sa vrátime k predchádzajúcemu príkladu ($p = 0.99$, $n = 15$), tak pravdepodobnosť neodhalenej chyby je 0.009226196681.

7.3.2 Obdĺžnikové kódy.

Uvažujme opäť binárne zapísanú informáciu, ktorú chceme upraviť do formy umožňujúcej opraviť aspoň jednu chybu (chybu váhy 1). Zapíšeme informáciu do obdĺžnikovej matice typu $m \times n$ a pridáme k nej jeden kontrolný riadok a jeden kontrolný stĺpec.

$$\begin{array}{l|l} 0110101010 & 1 \\ 1110000111 & 0 \\ 1010101010 & 1 \\ \hline 0010000111 & 0 \end{array}$$

Na i-tom mieste kontrolného stĺpca sa bude nachádzať paritný bit i-teho riadku ($a_{i,10} = a_{i,0} \oplus \cdots \oplus a_{i,9}$), na j-tom mieste kontrolného riadku sa bude nachádzať paritný bit j-teho

stĺpca ($a_{3,j} = a_{0,j} \oplus \cdots \oplus a_{2,j}$). Predpokladajme, že nastala chyba váhy 1 napríklad na mieste $(0,4)$. Príjemca vyčísli kontrolné sumy pre riadky aj stĺpce prijatej matice a zistí pozíciu chyby:

0110<u>0</u>01010	1	1
1110000111	0	0
1010101010	1	0
0010000111	0	0
0000100000	0	

Všimneme si, že ak vznikne chyba váhy 1 v kontrolnom riadku alebo v kontrolnom stĺpci, pozícia chyby sa určí úplne rovnako, ako v prípade chyby v „informačnom“ symbole:

0110101010	1	0
1110000111	0	0
1010101010	1	0
001000011<u>0</u>	0	1
0000000001	0	

Obdĺžnikový kód je schopný opravovať chyby váhy 1. Čo sa stane, ak v kódovom slove vznikne chyba väčšj váhy? Predpokladajme, že vznikla chyba váhy 2:

<u>0</u>110101010	1	1
1110000111	0	0
1010101010	1	0
001000011<u>0</u>	0	1
1000000001	0	

Vyčíslením kontrolných súm príjemca zistí, že vznikla chyba väčšej váhy. Ak by aj uhádol, že ide o chybu váhy 2, nevie či chyby vznikli v symboloch $a_{0,0}, a_{3,9}$ alebo $a_{3,0}, a_{0,9}$. Ak by chyba váhy 2 vznikla v tom istom riadku (stĺpci), na kontrolnej sume príslušného riadku (stĺpca) by sa to neprejavilo, a príjemca by vedel akurát povedať, že v niektorých stĺcoch (riadkoch) vznikla chyba väčšej váhy.

<u>1</u>110101011<u>1</u>	1	0
1110000111	0	0
1010101010	1	0
0010000111	0	0
1000000001	0	

Samoopravné kódy sa zakladajú na tom, že

- nie každé možné slovo je kódovým slovom;
- kódové slová sú „dosť ďaleko od seba“.

Minimálna vzdialenosť kódu vyjadrená pomocou Hammingovej vzdialenosti kódových slov nám umožnila precizovať význam slov „dosť ďaleko od seba“. Pomocou obdĺžnikových kódov ilustrujeme pojem „redundancie (nadbytočnosti)“, ktorý upresňuje prvú požiadavku kladenú na samoopravné kódy. V kódovom slove obdĺžnikového kódu rozlišujeme dva druhy symbolov: *informačné* (pomocou nich sa zapisuje informácia, ktorú

má kódové slovo preniesť) a *kontrolné symboly* (zaznamenávajúce štruktúru kódového slova.) Dĺžka kódového slova sa zvykne označovať symbolom n, počet informačných symbolov k a počet kontrolných symbolov je potom $n-k$. Samoopravný kód, ktorý má dĺžku n a počet informačných symbolov k sa označuje aj ako (n,k)-kód. Počet kontrolných symbolov sa nazýva *absolútnou redundanciou* kódu. Kódy s rozličnými dĺžkami môžu mať rozličné počty kontrolných symbolov. Aby ich bolo možné porovnávať z hľadiska redundancie, zavádzame pojem *relatívnej redundancie kódu*, definovanej ako podiel počtu kontrolných symbolov k celkovej dĺžke kódového slova; $\frac{n-k}{n} = 1 - \frac{k}{n}$. Určíme absolútnu a relatívnu nadbytočnosť obdĺžnikových kódov. Predpokladajme kvôli jednoduchosti, že kódové slovo obdĺžnikového kódu má tvar štvorcovej matice[4] typu $m \times m$. Táto matica obsahuje štvorcovú podmaticu $(m-1)\times(m-1)$ informačných symbolov a $2m-1$ kontrolných symbolov. Relatívna nadbytočnosť štvorcového kódu je $\frac{2m-1}{m^2} = \frac{2}{m} - \frac{1}{m^2}$. Pre veľké m je relatívna nadbytočnosť štvorcového kódu zanedbateľná.

V prípade obdĺžnikového kódu bolo možné rozlíšiť informačné a kontrolné symboly. Existujú samoopravné kódy, pre ktoré takéto rozdelenie symbolov kódového slova neexistuje. Aby bolo možné vyjadriť redundanciu aj pre tieto kódy, zovšeobecníme pojem redundancie nasledujúcim spôsobom.

Definícia 7.3.1. *Nech V je kód dĺžky n nad abecedou $\{0,\ldots,q-1\}$. (Relatívnou) redundanciou kódu V je*

$$R(V) = \frac{\log_q |V|}{n}.$$

Redundancia kódu úzko súvisí s ďalším dôležitým pojmom, pomocou ktorého sa vyjadruje efektívnosť kódu; s prenosovou rýchlosťou. Prenosová rýchlosť kódu je číslo z intervalu $<0,1>$, ktoré je definované ako

$$\frac{\text{počet prenesených informačných symbolov}}{\text{celkový počet prenesených symbolov}} = 1 - R.$$

V ďalšom sa budeme zaoberať kódmi, ktoré majú vysoké prenosové rýchlosti a zároveň dobré opravné schopnosti. Začneme zaujímavým kódom opravujúcim jednu chybu.

7.4 Hammingov kód

Hammingove kódy sú binárne (n,k)-kódy, s parametrami $n = 2^m - 1$, $m \geq 3$, $m \in N$, $k = 2^m - 1 - m$ opravujúce chyby váhy 1. Princíp vytvárania Hammingových kódov, kódovanie a dekódovanie ilustrujeme na Hammingovom $(15,11)$-kóde.

Predpokladajme, že sme už vytvorili kódové slovo $\mathbf{v} = (v_1,\ldots,v_{15})$. Z jednotlivých komponentov kódového slova vytvoríme 4 kontrolné sumy s_0, s_1, s_2, s_3, pomocou ktorých budeme schopní rozlišovať 16 rozličných udalostí: pri prenose nenastala žiadna chyba, nastala chyba váhy 1 v $1.,\ldots,15.$ komponente kódového slova. Zavedieme dva potrebné pojmy a potom vytvoríme kontrolné sumy. Symbolom $\sigma(i,n)$ budeme označovať n-bitový

[4] v takomto prípade hovoríme o štvorcovom kóde

vektor, ktorý je binárnou reprezentáciou čísla i. Nech sú $\mathbf{u} = (u_1, \ldots, u_n), \mathbf{v} = (v_1, \ldots, v_n)$ dva binárne vektory, symbolom $u\&v$ budeme označovať vektor $\mathbf{u}\&\mathbf{v} = (u_1v_1, \ldots, u_nv_n)$. Pre $j = 0, 1, 2, 3$ platí:

$$s_j = \bigoplus_{\sigma(i,4)\&\sigma(2^j,4)=\sigma(2^j,4)} v_i,$$

t.j.

$$\begin{aligned}
s_0 &= v_1 \oplus v_3 \oplus v_5 \oplus v_7 \oplus v_9 \oplus v_{11} \oplus v_{13} \oplus v_{15} \\
s_1 &= v_2 \oplus v_3 \oplus v_6 \oplus v_7 \oplus v_{10} \oplus v_{11} \oplus v_{14} \oplus v_{15} \\
s_2 &= v_4 \oplus v_5 \oplus v_6 \oplus v_7 \oplus v_{12} \oplus v_{13} \oplus v_{14} \oplus v_{15} \\
s_3 &= v_8 \oplus v_9 \oplus v_{10} \oplus v_{11} \oplus v_{12} \oplus v_{13} \oplus v_{14} \oplus v_{15}.
\end{aligned}$$

Všimnime si, že komponent v_i sa vyskytuje práve vo $\mathbf{wt}(\sigma(i, 4))$ kontrolných sumách. Keďže existujú práve štyri binárne vektory dĺžky 4 s Hammingovou váhou 1 reprezentujúce čísla $1, 2, 4, 8$, každý z komponentov v_1, v_2, v_4, v_8 vystupuje v jednej kontrolnej sume. To znamená, že ak zvolíme hodnoty komponentov $v_3, v_5, v_6, v_7, v_9, v_{10}, v_{11}, v_{12}, v_{13}, v_{14}, v_{15}$ kódového slova ľubovoľne, vhodnou voľbou komponentov v_1, v_2, v_4, v_8 dosiahneme, že kontrolné sumy budú pre kódové slovo nulové: $s_0 = s_1 = s_2 = s_3 = 0$. Stačí položiť:

$$\begin{aligned}
v_1 &= v_3 \oplus v_5 \oplus v_7 \oplus v_9 \oplus v_{11} \oplus v_{13} \oplus v_{15} \\
v_2 &= v_3 \oplus v_6 \oplus v_7 \oplus v_{10} \oplus v_{11} \oplus v_{14} \oplus v_{15} \\
v_4 &= v_5 \oplus v_6 \oplus v_7 \oplus v_{12} \oplus v_{13} \oplus v_{14} \oplus v_{15} \\
v_8 &= v_9 \oplus v_{10} \oplus v_{11} \oplus v_{12} \oplus v_{13} \oplus v_{14} \oplus v_{15}.
\end{aligned}$$

Kódovanie správ pomocou Hammingovho $(15, 11)$**-kódu** prebieha tak, že sa správa najprv rozdelí na bloky dĺžky 11 a tie sa doplnia 4 kontrolnými symbolmi na kódové slovo:

1	1	1	1	0	0	0	0	1	1	1					informačný vektor
		1		1	1	1		0	0	0	0	1	1	1	informačný vektor
1	1		0				1								kontrolný vektor
1	1	1	0	1	1	1	1	0	0	0	0	1	1	1	kódové slovo

Dekódovanie Hammingovho $(15, 11)$**-kódu.** Predpokladajme, že pri prenose kódového slova vznikla chyba váhy 1 v i-tom komponente kódového slova; t.j. bolo prijaté slovo

$$v_0, \ldots, v_{i-1}, v_i \oplus 1, v_{i+1}, \ldots, v_{15}.$$

Chyba spôsobí, že všetky kontrolné sumy, ktoré obsahujú komponent v_i nadobudnú hodnotu 1. To sú však práve tie sumy s_j, pre ktoré $\sigma(i, 4)\&\sigma(2^j, 4) = \sigma(2^j, 4)$; t.j. binárny vektor $s = (s_3, s_2, s_1, s_0)$ predstavuje číslo $\sigma(i, 4)$. Vektor hodnôt jednotlivých kontrolných súm sa nazýva *syndróm chyby*. V našom prípade syndróm chyby predstavuje pozíciu, na ktorej chyba váhy 1 v kódovom slove vznikla, resp. nulová hodnota syndrómu chyby znamená, že bolo prijaté kódové slovo.

Príklad. Predpokladajme, že chyba vznikla v 13. komponente kódového slova. Potom bolo prijaté slovo:

$$v_1, \ldots, v_{12}, v_{13} \oplus 1, v_{14}, v_{15}.$$

Kontrolné sumy nadobúdajú hodnoty:

$$\begin{aligned}
s_0 &= v_1 \oplus v_3 \oplus v_5 \oplus v_7 \oplus v_9 \oplus v_{11} \oplus (v_{13} \oplus 1) \oplus v_{15} = 1\\
s_1 &= v_2 \oplus v_3 \oplus v_6 \oplus v_7 \oplus v_{10} \oplus v_{11} \oplus v_{14} \oplus v_{15} = 0\\
s_2 &= v_4 \oplus v_5 \oplus v_6 \oplus v_7 \oplus v_{12} \oplus (v_{13} \oplus 1) \oplus v_{14} \oplus v_{15} = 1\\
s_3 &= v_8 \oplus v_9 \oplus v_{10} \oplus v_{11} \oplus v_{12} \oplus (v_{13} \oplus 1) \oplus v_{14} \oplus v_{15} = 1
\end{aligned}$$

Hammingov kód nie je schopný opravovať chyby váhy ≥ 2. Pri dekódovaní sa takéto chyby buď vôbec neodhalia alebo sa interpretujú ako chyby váhy 1:

1	1	1	0	1	1	1	1	0	0	0	0	1	1	1	kódové slovo
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	chybový vektor
0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	prijaté slovo
0	0	0	0												syndróm chyby
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	predpokladaná chyba
0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	dekódované slovo
1	1	1	0	1	1	1	1	0	0	0	0	1	1	1	kódové slovo
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	chybový vektor
0	0	1	0	1	1	1	1	0	0	0	0	1	1	1	prijaté slovo
0	0	1	1												syndróm chyby
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	predpokladaná chyba
0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	dekódované slovo

Porovnáme ešte redundanciu Hammingových R_H a obdĺžnikových kódov R_O.

n	$a \times b$	$(n-k)_H$	$(n-k)_O$	R_H	R_O
$7^{(*)}$	2×4	3	5	0.4285	0.6250
15	3×5	4	7	0.2666	0.4666
$31^{(*)}$	4×8	5	10	0.1612	0.3125
63	7×9	6	15	0.0952	0.2380
$127^{(*)}$	8×16	7	23	0.0551	0.1796
255	15×17	8	31	0.0313	0.1215
511	16×32	9	72	0.0176	0.1409
1023	31×33	10	63	0.0097	0.0615

V prípadoch označených hviezdičkou neexistujú obdĺžnikové kódy potrebnej dĺžky (n je prvočíslo), a preto sme Hammingove kódy dĺžky n porovnávali s obdĺžnikovými kódmi dĺžky $n+1$. Aj pre $n = 511$ má obdĺžnikový kód dĺžky 512 rozmerov 16×32 menšiu redundanciu (0.0917) ako obdĺžnikový kód dĺžky 511 rozmerov 7×73.

Poznámka. Pre Hammingove kódy platí ($n = 2^m - 1,\ m \geq 3$)

$$\left[\binom{n}{0} + \binom{n}{1}\right] \mid 2^n,$$

a teda Hammingove kódy sú dokonalé binárne kódy.

Kapitola 8

Lineárne kódy

Predchádzajúce konštrukcie samoopravných kódov (obdĺžnikové kódy, Hammingove kódy) nám umožnili zostrojiť samoopravné kódy opravujúce chyby váhy 1. Ak je však pravdepodobnosť chyby pri prenose znaku dostatočne vysoká, budeme na zaistenie spoľahlivého prenosu správ takýmto prenosovým kanálom potrebovať samoopravné kódy s vyššou opravnou schopnosťou. Vychádzajúc z geometrickej predstavy o samoopravných kódoch by sme teoreticky mohli skonštruovať potrebný samoopravný kód, ale je otázne jednak to, či by sa táto konštrukcia dala spraviť v rozumnom čase a či by pre takto zostrojený kód existovali efektívne metódy kódovania a dekódovania. Schodnejšou cestou pre konštrukciu samoopravného kódu je nájsť vhodnú algebraickú štruktúru a jej prvky použiť ako kódové slová. Asi prvé čo nám napadne, je zobrať konečnú grupu a ako kód použiť nejakú jej vhodnú podgrupu. Kódy, ktorých slová s nejakou binárnou operáciou tvoria grupu, skutočne existujú a nazývajú sa *grupové kódy*.

Príklad. Nech je $(G, +)$ konečná grupa s (napríklad) aditívnou operáciou. Množina $(G^n, \oplus)$ je grupa, ktorej prvkami sú usporiadané n-tice prvkov grupy G a operácia $\oplus$ je odvodená z aditívnej operácie grupy G nasledovne: nech $\mathbf{u} = (u_1, \ldots, u_n$ a $\mathbf{v} = (v_1, \ldots, v_n)$ sú prvky G^n, potom $\mathbf{u} \oplus \mathbf{v} = (u_1 + v_1 \ldots u_n + v_n)$. Je zrejmé, že operácia $\oplus$ je asociatívna, množina G^n je uzavretá vzhľadom na operáciu $\oplus$, neutrálnym prvkom v G^n vzhľadom na opráciu $\oplus$ je vektor $(0, \ldots, 0)$, kde 0 je neutrálny prvok grupy G a napokon,k ľubovoľnému prvku $(u_1, \ldots, u_n) \in G^n$ existuje opačný prvok $(-u_1, \ldots, -u_n) \in G^n$, kde $-u_i$ je opačný prvok k prvku $u_i,\ i = 1, \ldots, n$.[1]

Aby sme dosiahli požadovanú efektívnosť kódovania a dekódovania, budeme na konštrukciu samoopravných kódov používať o niečo zložitejšie štruktúry, ako sú grupy. Predpokladáme, že je dané konečné pole $GF(q)$, kde q je mocnina nejakého prvočísla p. (Najčastejšie budeme pracovať s $q = p = 2$.) Množina $GF(q)^n$ n-tíc (vektorov) nad poľom $GF(q)$ s aditívnou operáciou "$+$" (sčítanie po zložkách) a multiplikatívnou operáciou "$\cdot$" (násobenie zložiek vektora prvkom poľa $GF(q)$) tvorí vektorový priestor.[2] Lineárnym kódom nad abecedou $GF(q)$ je ľubovoľný vektorový (lineárny) podpriestor vektorového

[1] Tam, kde to nepovedie k nedorozumeniu, budeme v dalšom aditívnu operáciu nad vektormi označovať symbolom "+"

[2] Podrobnejšie informácie o vektorových priestoroch, konečných poliach a ďalších algebraických štruktúrach nájde čitateľ v **??**.

priestoru $GF(q)^n$. Ak je dimenzia vektorového podpriestoru C rovná k, lineárny kód C sa nazýva lineárnym (n,k)-kódom.

Príklad. Nech $q = 2, n = 8$. Pozrieme sa najprv na dva extrémne prípady. Ak $k = 8$, kód C má $2^8 = 256$ kódových slov. Tento kód pozostáva zo všetkých možných binárnych vektorov/slov dĺžky 8, má prenosovú rýchlosť 1 ale jeho opravná schopnosť je nulová. Druhým extrémnym prípadom je lineárny $(8,0)$ kód, ktorý má dimenziu 0, jediné kódové slovo (napríklad $\mathbf{v_0} = (00000000)$), ale je na prenos informácie prakticky bezcenný, lebo nedokáže preniesť jediný bit informácie[3]. Prakticky použiteľný kód s najmenším počtom kódových slov je lineárny $(8,1)$-kód, s mohutnosťou 2. Tento kód obsahuje dve kódové slová: nulové slovo $\mathbf{v_0} = (00000000)$ a nenulové slovo $\mathbf{v_1}$. Slovo $\mathbf{v_1}$ môžeme vybrať ľubovoľne, pretože $\mathbf{v_1} + \mathbf{v_1} = (00000000)$ a tak konštruovať kódy s rozličnými opravnými schopnosťami. Položíme $\mathbf{v_1} = (11111111)$ a dostávame kód s maximálnou vzdialenosťou $d = 8$ rozpoznávajúci chyby váhy ≤ 7 a opravujúci chyby váhy ≤ 3. Prenosová rýchlosť tohto kódu je $1/8$ a redundancia $7/8$.

8.1 Základné vlastnosti lineárnych kódov

Lineárny kód je teda ľubovoľná neprázdna množina vektorov (n-tíc) C taká, že pre ľubovoľné $\mathbf{v_1}, \ldots, \mathbf{v_m} \in C$, $a_1, \ldots, a_m \in GF(q)$ patrí aj lineárna kombinácia $a_1\mathbf{v_1} + \cdots + a_m\mathbf{v_m}$ do množiny C. To znamená, že pre ľubovoľné kódové slovo $\mathbf{u}$ a prvok $a \in GF(q)$ patrí do kódu C aj slovo $a\mathbf{u}$ a pre ľubovoľné dve kódové slová $\mathbf{u}, \mathbf{v}$ je potom aj $\mathbf{u} + \mathbf{v}$ a $\mathbf{u} - \mathbf{v}$ kódové slovo kódu C. Keďže pre ľubovoľné $\mathbf{u} \in C$ platí $\mathbf{u} - \mathbf{u} = \mathbf{0}$, nulové slovo patrí vždy do kódu C. Vzhľadom na tieto skutočnosti sa štúdium viacerých vlastností lineárnych kódov redukuje na skúmanie vzťahov medzi nulovým kódovým slovom a ostatnými kódovým slovami lineárneho kódu C.

Veta 8.1.1. *Nech je C lineárny kód. Potom pre minimálnu vzdialenosť d^* kódu C platí nasledujúci vzťah*

$$d^* = \min_{\mathbf{u},\mathbf{v} \in C} d(\mathbf{u},\mathbf{v}) = \min_{\mathbf{u} \neq \mathbf{0} \in C} \mathbf{wt}(\mathbf{u}).$$

Dôkaz. Nech sú $\mathbf{u}, \mathbf{v}$ dve slová, ktorých vzdialenosť sa rovná minimálnej vzdialenosti kódu C: $d(\mathbf{u},\mathbf{v}) = d^*$. Slovo $\mathbf{u} - \mathbf{v}$ je tiež kódové slovo a pre jeho váhu platí $\mathbf{wt}(\mathbf{u} - \mathbf{v}) = d(\mathbf{u},\mathbf{v}) = d^*$. Na druhej strane, ak by v kóde C existovalo nenulové kódové slovo $\mathbf{x}$ s váhou $\mathbf{wt}(\mathbf{x}) < d^*$, potom by aj $d(\mathbf{x},\mathbf{0}) < d^*$ čo je v spore s definíciou minimálnej vzdialenosti. □

Ak si dáme do súvislosti geometrickú interpretáciu samoopravných kódov s tvrdením vety, tak vidíme, že na zostrojenie samoopravného kódu opravujúceho chyby váhy t stačí zostrojiť lineárny kód s minimálnou váhou $w^* \geq 2t + 1$.

Budeme pokračovať v skúmaní lineárnych kódov. Pripomenieme, že *skalárny súčin dvoch vektorov* $\mathbf{u} = (u_1, \ldots, u_n)$ a $\mathbf{v} = (v_1, \ldots, v_n)$ je definovaný ako

$$\langle \mathbf{u}, \mathbf{v} \rangle = u_1v_1 + \cdots + u_nv_n.$$

[3] Aj takto kód sa však dá použiť v špeciálnych prípadoch, napríklad pri testovaní prenosového kanála.

Nech C je lineárny podpriestor vektorového priestoru $GF(q)^n$. Dá sa ľahko overiť, že množina $C^\perp$, definovaná nasledovne

$$C^\perp = \{\mathbf{u} \in GF(q)^n; \quad \forall\, \mathbf{v} \in C \quad \langle \mathbf{u}, \mathbf{v} \rangle = 0\}$$

tvorí lineárny podpriestor vektorového priestoru $GF(q)^n$. (Podpriestor $C^\perp$ sa nazýva *ortogonálny doplnok podpriestoru* C.) Kedže $C^\perp$ je lineárny podpriestor vektorového priestoru $GF(q)^n$, predstavuje podľa definície lineárneho kódu taktiež lineárny kód, ktorý budeme nazývať *duálnym kódom kódu* C.

Poznámka. To, že je $C^\perp$ ortogonálnym doplnkom lineárneho podpriestoru C neznamená, že sú tieto podpriestory disjunktné. Zrejme $\mathbf{0} \in C^\perp \bigcap C$ a existujú dokonca lineárne kódy, pre ktoré platí $C^\perp = C$ (samoduálne lineárne kódy.)

Vďaka tomu, že lineárny kód C predstavuje lineárny podpriestor vektorového priestoru $GF(q)^n$, možno ho popísať efektívnejšie, ako tie blokové kódy, ktoré nemali žiadnu rozumnú štruktúru a bolo ich potrebné popísať vymenovaním všetkých kódových slov. Lineárny podpriestor je jednoznačne zadaný pomocou množiny vektorov, ktorá ho generuje. Spomedzi všetkých množín vektorov, generujúcich daný lineárny podpriestor (lineárny kód) C vyberieme množinu s minimálnym počtom prvkov[4], bázu a vektory-prvky bázy zapíšeme ako riadky matice G. Matica G sa nazýva *generujúcou maticou lineárneho kódu* C, pretože ľubovoľný vektor-kódové slovo kódu C možno zapísať pomocou lineárnej kombinácie vektorov-riadkov matice G. Ak je C lineárnym podpriestorom dimenzie k vektorového priestoru dimenzie n, tak jeho generujúca matica G má k (lineárne nezávislých) riadkov a n stĺpcov. Pripomíname, že samotný kód C má potom q^k kódových slov.

Generujúca matica umožňuje efektívne vytváranie kódových slov. Ľubovoľný vektor $\mathbf{i} \in GF(q)^k$; $\mathbf{i} = (i_1, \ldots, i_k)$ môžeme chápať ako k-ticu informačných symbolov (informačný vektor) a transformovať ho na kódové slovo nasledujúcim spôsobom

$$\mathbf{u} = \mathbf{i}G,$$

kde G je generujúca matica lineárneho (n, k) kódu. Pripomíname, že existuje viacero spôsobov výberu generujúcej matice G lineárneho kódu a tak sa informačnému vektoru $\mathbf{i}$ v závislosti od výberu G vo všeobecnosti priradia rozličné slová. V časti **??** sa budeme podrobnejšie zaoberať vplyvom výberu generujúcej matice na vlastnosti lineárneho kódu.

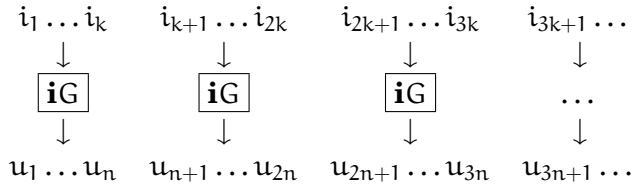
Ako vyzerá kódovanie správy[5] pomocou lineárneho (n, k) kódu? Postupnosť znakov správy sa rozdelí na bloky dĺžky k a tie sa postupne vynásobia generujúcou maticou G a tak sa transformujú na postupnosť kódových slov, obr. 8.1

Ilustrujeme kódovanie správy pomocou lineárneho kódu na nasledujúcom jednoduchom príklade.

Príklad 8.1. *Hammingove kódy opravujúce chyby váhy 1 sú lineárne kódy. Kvôli zjednodušeniu výpočtov zoberieme kratší Hammingov kód, ako sme skonštruovali v časti 7.4;*

[4] pripomíname, že takýchto množín je viac a tak vyberieme ľubovoľnú z nich.

[5] predpokladáme, že správa je zapísaná ako postupnosť znakov - prvkov $GF(q)$



Obr. 8.1: Kódovanie správy pomocou lineárneho kódu s generujúcou maticou G

Hammingov $(7,4)$ *kód* C. *Kód* C *je binárny kód dĺžky* 7 *s minimálnou vzdialenosťou* $d^* = 3$*; kódové slovo má* 4 *informačné a* 3 *kontrolné symboly. Nech sú* i_1, i_2, i_3, i_4 *informačné symboly kódového slova* $\mathbf{u} = (u_1, \dots, u_7)$. *Položíme* $u_3 = i_1, u_5 = i_2, u_6 = i_3, u_7 = i_4$. *Hodnoty kontrolných symbolov* u_1, u_2, u_4 *vypočítame pomocou troch kontrolných súm (znak "+" označuje súčet modulo 2):*

$$\begin{aligned} u_1 + u_3 + u_5 + u_7 &= 0 \qquad & u_1 &= i_1 + i_2 + i_4 \\ u_2 + u_3 + u_6 + u_7 &= 0 \qquad & u_2 &= i_1 + i_3 + i_4 \\ u_4 + u_5 + u_6 + u_7 &= 0 \qquad & u_4 &= i_2 + i_3 + i_4 \end{aligned}$$

Nulový vektor spĺňa vyššie uvedené vzťahy a teda nulové slovo je kódovým slovom Hammingovho $(7,4)$ *kódu. Nech sú* $\mathbf{u}, \mathbf{v}$ *dve kódové slová Hammingovho* $(7,4)$ *kódu; t.j.*

$$\mathbf{u} = (u_3 + u_5 + u_7, u_3 + u_6 + u_7, u_3, u_5 + u_6 + u_7, u_5, u_6, u_7),$$

$$\mathbf{v} = (v_3 + v_5 + v_7, v_3 + v_6 + v_7, v_3, v_5 + v_6 + v_7, v_5, v_6, v_7).$$

Potom súčet vektorov

$$\begin{aligned} \mathbf{u} + \mathbf{v} \;=\; & (u_3 + u_5 + u_7 + v_3 + v_5 + v_7, u_3 + u_6 + u_7 + v_3 + v_6 + v_7, u_3 + v_3, \\ & u_5 + u_6 + u_7 + v_5 + v_6 + v_7, u_5 + v_5, u_6 + v_6, u_7 + v_7) = \\ & (u_3 + v_3 + u_5 + v_5 + u_7 + v_7, u_3 + v_3 + u_6 + v_6 + u_7 + v_7, u_3 + v_3, \\ & u_5 + v_5 + u_6 + v_6 + u_7 + v_7, u_5 + v_5, u_6 + v_6, u_7 + v_7) \end{aligned}$$

tiež spĺňa kontrolné sumy a teda patrí do kódu C. *Tým sme dokázali, že ľubovoľná lineárna kombinácia vektorov-slov kódu* C *je kódovým slovom (pripomíname, že* $GF(2)^7$ *s operáciami modulárneho sčítania po zložkách je vektorový priestor a koeficienty v lineárnej kombinácii sú prvky poľa* $GF(2)$*, t.j. prvky množiny* $\{0, 1\}$*, a teda aj to, že Hammingov kód je lineárny kód. (Hammingov (7,4) kód je uvedený v nasledujúcej tabuľke).*

0000000	1110000	1001100	0101010
1101001	0111100	1011010	0011001
1100110	0100101	1000011	0010110
1010101	0110011	0001111	1111111

Generujúca matica Hammingovho (7,4)-kódu vyzerá nasledovne:

$$G = \begin{bmatrix} 1110000 \\ 1001100 \\ 0101010 \\ 1101001 \end{bmatrix}$$

Vytvoríme kódové slovo pre informačný vektor $\mathbf{i} = (1111)$*:*

$$(1111) \begin{bmatrix} 1110000 \\ 1001100 \\ 0101010 \\ 1101001 \end{bmatrix} = (1111111).$$

Pri dekódovaní správ zakódovaných pomocou lineárneho kódu C možno výhodne použiť generujúcu maticu duálneho kódu $C^{\perp}$, ktorú označíme symbolom H. Ak je C lineárny (n, k)-kód, $C^{\perp}$ je lineárny $(n, n-k)$-kód a generujúca matica kódu $C^{\perp}$ má $n-k$ riadkov a n stĺpcov, pričom riadky matice H tvoria vektory bázy lineárneho podpriestoru $C^{\perp}$. Keďže C je ortogonálny doplnok lineárneho podpriestoru $C^{\perp}$, každý vektor (kódové slovo) $\mathbf{u} \in C$ je ortogonálny na ľubovoľný vektor $\mathbf{v} \in C^{\perp}$ a špeciálne, na ľubovoľný vektor-riadok matice H. To znamená, že u je kódové slovo práve vtedy, ak

$$\mathbf{u}H^{\top} = \mathbf{0}$$

kde $\mathbf{0}$ je v tomto prípade nulový vektor dĺžky $n - k$ Keďže matica H umožňuje overiť, či je nejaké slovo kódovým slovom kódu C, nazýva sa *kontrolnou maticou kódu* C. Pripomíname ešte, že generujúca matica G kódu C je kontrolnou maticou jeho duálneho kódu $C^{\perp}$.

Príklad 8.2. *Kontrolná matica Hammingovho* $(7, 4)$ *kódu z príkladu 8.1 má tvar*

$$H = \begin{bmatrix} 1010101 \\ 0110011 \\ 0001111 \end{bmatrix}$$

Na základe kontrolnej matice je možné určiť minimálnu vzdialenosť príslušného lineárneho kódu.

Veta 8.1.2. *Lineárny kód* C *nad poľom* $GF(q)$ *obsahuje nenulové kódové slovo váhy menšej alebo rovnej* w *práve vtedy, ak jeho kontrolná matica* H *obsahuje* w *lineárne závislých stĺpcov.*

Dôkaz Označme vektory-stĺpce kontrolnej matice H symbolmi $\mathbf{h_1}, \ldots, \mathbf{h_n}$;

$$H = [\mathbf{h_1}^{\top}, \ldots, \mathbf{h_n}^{\top}].$$

Nech v množine vektorov $\{\mathbf{h_1}, \ldots, \mathbf{h_n}\}$ existuje w lineárne závislých vektorov $\mathbf{h_{i_1}}, \ldots, \mathbf{h_{i_w}}$; t.j. existujú také konštanty $a_{i_1}, \ldots, a_{i_w} \in GF(q)$, že

$$a_{i_1}\mathbf{h_{i_1}} + \ldots, a_{i_w}\mathbf{h_{i_w}} = \mathbf{0}$$

Medzi konštantami $a_{i_1}, \ldots, a_{i_w} \in GF(q)$ je aspoň jedna nenulová, a teda vektor $\mathbf{a}$ ktorého komponenty na pozíciách $i_1, \ldots, i_w$ nadobúdajú v poradí hodnoty $a_{i_1}, \ldots, a_{i_w}$ a ostatné komponenty sú nulové, predstavuje nenulové kódové slovo váhy $\leq w$, nakoľko

$$\mathbf{a}H^\top = a_{i_1}\mathbf{h_{i_1}} + \ldots, a_{i_w}\mathbf{h_{i_w}} = \mathbf{0}.$$

Na druhej strane, nech v kóde C existuje kódové slovo $\mathbf{a}$ váhy w s nenulovými komponentami $a_{i_1}, \ldots, a_{i_w}$. Keďže $\mathbf{a}$ je kódové slovo, platí preň

$$\mathbf{a}H^\top = a_{i_1}\mathbf{h_{i_1}} + \ldots, a_{i_w}\mathbf{h_{i_w}} = \mathbf{0};$$

t.j. vektory-stĺpce kontrolnej matice $\mathbf{h_{i_1}}, \ldots, \mathbf{h_{i_w}}$ sú lineárne závislé. □

Dôsledok. Lineárny kód C s kontrolnou maticou H má minimálnu vzdialenosť w práve vtedy, ak je v matici H ľubovoľných $w-1$ stĺpcov lineárne nezávislých a v H existuje w lineárne závislých stĺpcov.

To znamená, že ak chceme zostrojiť lineárny (n,k)-kód C opravujúci chyby váhy aspoň t, musíme skonštruovať maticu H typu $(n-k)\times n$, v ktorej je ľubovoľných $2t$ stĺpcov lineárne nezávislých a použiť ju ako kontrolnú maticu kódu C. Keďže medzi vektormi dĺžky $n-k$ môže byť nanajvýš $n-k$ lineárne nezávislých vektorov, v matici H môžu byť nezávislé nanajvýš všetky $(n-k)$-tice stĺpcov, ale ľubovoľná $(n-k+1)$-tica stĺpcov matice H je lineárne závislá. Tým sme dokázali nasledujúcu vetu.

Veta 8.1.3 (Singeltonova hranica)**.** *Pre minimálnu vzdialenosť (minimálnu váhu) lineárneho* (n,k)*-kódu platí nasledujúca nerovnosť*

$$d^* \leq 1 + n - k.$$

Ľubovoľný lineárny kód s minimálnou vzdialenosťou, ktorá spĺňa rovnosť

$$d^* = 1 + n - k$$

sa nazýva *lineárnym kódom s maximálnou vzdialenosťou*. Zo Singeltonovej hranice vyplýva, že na to, aby kód bol schopný opravovať chyby váhy t musí v kódovom slove byť aspoň $2t$ kontrolných symbolov; t.j aspoň dva kontrolné symboly na chybu v jednom komponente kódového slova. Väčšina samoopravných kódov má viac kontrolných symbolov.

Aj keď je ľubovoľná $n \times k$ matica, ktorej riadky tvoria bázu lineárneho podpriestoru dimenzie k generujúcou maticou lineárneho (n,k)-kódu, kvôli zjednodušeniu výpočtov upravíme generujúcu maticu na nasledujúci štandardný tvar; $G = [I_k \vdots P]$, kde I_k je jednotková matica rádu k a P je matica typu $k \times (n-k)$. (Výhodou štandardného tvaru generujúcej matice je o.i. aj to že sa z nej dá jednoducho odvodiť kontrolná matica.) Ukážeme, že pre ľubovoľný lineárny (n,k)-kód generovaný generujúcou maticou G existuje lineárny (n,k)-kód generovaný generujúcou maticou v štandardnom tvare s rovnakými parametrami.

Nech je G generujúca matica lineárneho (n,k)-kódu C. Riadky matice G tvoria bázu lineárneho podpriestoru C. Z lineárnej algebry je známe (pozri napr. [4]), že ak vektory bázy transformujeme pomocou nasledujúcich transformácií

- vektor nahradíme jeho nenulovým násobkom;
- k vektoru bázy pripočítame ľubovoľnú lineárnu kombináciu ostatných vektorov,

výsledná množina vektorov bude tvoriť bázu pôvodného lineárneho priestoru. To znamená, že ľubovoľná matica G', ktorú dostaneme z generujúcej matice G pomocou vyššie uvedených elementárnych operácií nad riadkami, je generujúcou maticou pôvodného kódu C.

Ďalšou transformáciou generujúcej matice je výmena jej stĺpcov. Nech je $G = [\mathbf{g_1 g_2} \dots \mathbf{g_n}]$ generujúca matica kódu C. Predpokladajme kvôli jednoduchosti, že v generujúcej matici G vymeníme prvý a druhý stĺpec, t.j. dostávame maticu $G'' = [\mathbf{g_2 g_1} \dots \mathbf{g_n}]$. Je zrejmé, že riadky matice G'' sú lineárne nezávislé, a teda matica G'' je generujúcou maticou lineárneho (n, k)-kódu, ktorý označíme symbolom C''. Nech je $\mathbf{i} = (i_1 \dots i_k)$ ľubovoľný informačný vektor. Pre vektor $\mathbf{i}$ zostrojíme kódové slová tak v kóde C ako aj v C'':

$$\mathbf{i}G = (\langle \mathbf{i}, \mathbf{g_1} \rangle, \langle \mathbf{i}, \mathbf{g_2} \rangle, \dots, \langle \mathbf{i}, \mathbf{g_n} \rangle) = \mathbf{u} = (u_1, u_2, \dots, u_n);$$

$$\mathbf{i}G'' = (\langle \mathbf{i}, \mathbf{g_2} \rangle, \langle \mathbf{i}, \mathbf{g_1} \rangle, \dots, \langle \mathbf{i}, \mathbf{g_n} \rangle) = \mathbf{u}'' = (u_2, u_1, \dots, u_n).$$

Z vyššie uvedeného vyplýva, že výmena i-teho a j-teho stĺpca generujúcej matice kódu C zodpovedá s výmenou i-teho a j-teho komponentu v kódových slovách kódu C. Nech je π ľubovoľná permutácia množiny $1 \dots n$ a nech sú $\mathbf{u}, \mathbf{v}$ ľubovoľné kódové slová kódu C, resp. $\mathbf{u}'', \mathbf{v}''$ im prislúchajúce kódové slová kódu C'', ktorý dostaneme permutáciou komponentov kódových slov permutáciou π potom

$$d(\mathbf{u}, \mathbf{v}) = \sum_{1 \le i \le n} (u_i \neq v_i) = \sum_{\pi(i), 1 \le i \le n} (u_{\pi(i)} \neq v_{\pi(i)}) = d(\mathbf{u}'', \mathbf{v}'')$$

To znamená, že kódy, ktoré dostaneme z kódu C pomocou elementárnych operácií nad riadkami a permutácii stĺpcov generujúcej matice G majú rovnaké parametre (počet kódových slov, minimálnu vzdialenosť) ako kód C a preto ich budeme nazývať *ekvivalentnými kódmi*. Pomocou vyššie popísaných transformácií nad riadkami a stĺpcami generujúcej matice možno ľubovoľnú generujúcu maticu transformovať na tvar

$$G = [I_k \vdots P],$$

kde I_k je jednotková matica rádu k a P je matica typu $k \times (n - k)$. Generujúca matica v tvare $G = [I_k \vdots P]$ sa nazýva *generujúcou maticou v systematickom tvare.* Ak $G = [I_k \vdots P]$, príslušná kontrolná matica má tvar

$$H = [-P^\top \vdots I_{n-k}],$$

nakoľko $GH^\top = -P + P = 0$. Ak má kód generujúcu maticu v systematickom tvare, tak v jeho kódových slovách nasledujú kontrolné symboly až za informačnými. Takýto kód sa niekedy nazýva *systematickým kódom* [2, 1].

Poznámka. Jacobus van Lint [15] definuje systematický kód odlišne: kód C je systematický v k komponentoch, ak $|C| = q^k$ a pre ľubovoľný výber hodnôt v daných k komponentoch existuje práve jedno kódové slovo. Keďže lineárny systematický (n, k)-kód podľa našej definície je systematickým kódom aj podľa van Linta (ale nie naopak), budeme sa pridržiavať zavedenej definície systematického kódu.

Veta 8.1.4. *Ku každému lineárnemu kódu existuje ekvivalentný systematický lineárny kód.*

Dôkaz. Nech je C ľubovoľný lineárny (n, k)-kód s generujúcou maticou G. Potom generujúcu maticu G možno transformovať na maticu G'' v systematickom tvare, ktorá je generujúcou maticou kódu ekvivalentnému kódu C. Keďže G'' je systematickom tvare, kód ktorý generuje je systematický. □

Poznámka. Z predchádzajúcej vety vyplýva, že ak to bude potrebné, môžeme bez ujmy na všeobecnosti predpokladať, že lineárny kód je systematický.

8.2 Dekódovanie lineárnych kódov

Nech je daný lineárny kód C a nech $\mathbf{u} \in C$ je odvysielané kódové slovo. Predpokladajme, že pri prenose slova $\mathbf{u}$ vznikla chyba $\mathbf{e}$ v jej dôsledku bolo prijaté slovo $\mathbf{w} = \mathbf{u} + \mathbf{e}$. Ako tabuľku dekódovania budeme na dekódovanie lineárnych kódov budeme používať tzv. *maticu štandardného rozkladu*, ktorú vytvoríme tak, že faktorizujeme aditívnu grupu $(GF(q)^n, +)$ podľa C a za reprezentantov jednotlivých tried rozkladu vyberieme vektory minimálnej váhy, ktoré sa v daných triedach rozkladu nachádzajú. Rozklad potom zapíšeme v podobe matice, kde v prvom stĺpci sú uvedení reprezentanti tried rozkladu a v prvom riadku kódové slová kódu C:

$\mathbf{v_0}$	$\mathbf{v_1}$	$\mathbf{v_2}$	$\dots$	$\mathbf{v_{q^k-1}}$
$\mathbf{e_1}$	$\mathbf{v_1} + \mathbf{e_1}$	$\mathbf{v_2} + \mathbf{e_1}$	$\dots$	$\mathbf{v_{q^k-1}} + \mathbf{e_1}$
$\mathbf{e_2}$	$\mathbf{v_1} + \mathbf{e_2}$	$\mathbf{v_2} + \mathbf{e_2}$	$\dots$	$\mathbf{v_{q^k-1}} + \mathbf{e_2}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\mathbf{e_{q^{n-k}-1}}$	$\mathbf{v_1} + \mathbf{e_{q^{n-k}-1}}$	$\mathbf{v_2} + \mathbf{e_{q^{n-k}-1}}$	$\dots$	$\mathbf{v_{q^k-1}} + \mathbf{e_{q^{n-k}-1}}$

Dekódovanie pomocou tabuľky dekódovania vyzerá nasledovne: nájdeme v tabuľke prijaté slovo $\mathbf{w}$. Ak sa $\mathbf{w}$ nachádza v stĺpci j dekódujeme ho ako kódové slovo $\mathbf{v_j}$; t.j. ako kódové slovo, ktoré sa nachádza v prvom riadku a j-tom stĺpci tabuľky dekódovania. Je zrejmé, že dekódovanie pomocou tabuľky dekódovania naráža na niekoľko problémov. Prvým je možnosť nesprávneho dekódovania prijatého slova. Ak pri prenose kódového slova $\mathbf{u}$ vznikne chyba $\mathbf{v} + \mathbf{e}$, kde $\mathbf{v}$ je kódové slovo, tak sa prijaté slovo $\mathbf{w} = \mathbf{u} + (\mathbf{v} + \mathbf{e})$ dekóduje nesprávne na kódové slovo $\mathbf{u} + \mathbf{v}$. Kritériá správneho dekódovania lineárneho[6] kódu pomocou tabuľky dekódovania uvádza nasledujúca veta.

[6] tvrdenie vety platí pre blokový kód nad $GF(q)$

Veta 8.2.1. *Ak sa matica štandardného rozkladu používa ako matica dekódovania lineárneho kódu, tak sa prijatý wektor* $\mathbf{w}$ *dekóduje na odvysielaný vektor (kódové slovo)* $\mathbf{u}$ *práve vtedy, ak chyba* $\mathbf{w}-\mathbf{u}$*, ktorá vznikla pri prenose je reprezentantom niektorej triedy rozkladu.*

Dôkaz. Nech je $\mathbf{e_i}=\mathbf{w}-\mathbf{u}$ reprezentantom niektorej triedy rozkladu. Potom prijatý vektor $\mathbf{w}$ patrí do triedy $[\mathbf{e_i}]$ a v tabuľke dekódovania sa nachádza v stĺpci určenom vektorom $\mathbf{u}=\mathbf{w}-\mathbf{e_i}$, a teda bude dekódovaný správne.
Nech na druhej strane $\mathbf{e_i}=\mathbf{w}-\mathbf{u}$ nie je reprezentantom niektorej triedy rozkladu; t.j. prijaté slovo $\mathbf{w}$ patrí do triedy $[\mathbf{e_j}]$, $i \neq j$. Potom sa však $\mathbf{w}$ nachádza v stĺpci zodpovedajúcom kódovému slovu $\mathbf{w}-\mathbf{e_j}$, a teda sa dekóduje nesprávne. □

Z vety 8.2.1 vyplýva, že nie je možné vylúčiť možnosť nesprávneho dekódovania lineárneho kódu. Predpokladajme, že prenosový kanál je q-nárny symetrický kanál[7] a potom vhodnou voľbou reprezentantov tried rozkladu môžeme minimalizovať pravdepodobnosť nesprávneho dekódovania. Využijeme nasledujúci dôsledok predchádzajúcej vety.

Dôsledok vety 8.2.1 (Lineárny) kód opravuje chyby váhy $\leq t$ práve vtedy, ak sú reprezentantami tried rozkladu všetky vektory váhy t a menšej.

Kód, definujúci rozklad, v ktorom sú reprezentantmi tried rozkladu všetky vektory váhy t a menšej, sa nazýva *dokonalým kódom*. Ak sa dekódovanie prijatého slova robí na základe maximálnej pravdepodobnosti (t.j. prijaté slovo sa dekóduje na kódové slovo, ktoré bolo s najvyššou pravdepodobnosťou odvysielané), tak pravdepodobnosť nesprávneho dekódovania dokonalého kódu je minimálna. Problém je v tom, že dokonalý lineárny (n,k)-kód nad $GF(q)$ opravujúci chyby váhy $\leq t$, musí spĺňať nasledujúcu podmienku:

$$\sum_{j=0}^{t}\binom{n}{j}(q-1)^j = q^{n-k}$$

a kódov, ktorých parametre túto podmienku spĺňajú takmer niet (ako sme už spomínali, jedinými známymi dokonalými kódmi sú Hammingove kódy a Golayov kód). Preto sa podmienka o reprezentantoch tried rozkladu mierne oslabuje a zavádza sa pojem *kvázidokonalého kódu* ako kódu definujúceho rozklad, v ktorom sú reprezentantmi tried rozkladu všetky vektory váhy t a menšej a niekoľko vektorov váhy $t+1$. Ilustrujeme zavedené pojmy na nasledujúcom príklade.

Príklad 8.3. *[2] Lineárny (5,2)-kód* C *s generujúcou maticou*

$$G = \begin{bmatrix} 10111 \\ 01101 \end{bmatrix}$$

a kontrolnou maticou

$$H = \begin{bmatrix} 11100 \\ 10010 \\ 11001 \end{bmatrix}$$

[7] Pripomíname, že pri prenose slova q-nárnym symetrickým kanálom je pravdepodobnosť toho, že vznikne chyba menšej váhy väčšia, ako pravdepodobnosť toho, že vznikne chyba väčšej váhy.

má minimálnu vzdialenosť 3 a umožňuje opravovať chyby váhy 1. Matica štandardného rozdelenia kódu C *vyzerať nasledovne:*

00000	10111	01101	11010
00001	10110	01100	11011
00010	10101	01111	11000
00100	10011	01001	11110
01000	11111	00101	10010
10000	00111	11101	01010
00011	10100	01110	11001
00110	10001	01011	11100

Kód C *je kvázidokonalý kód, pretože okrem nulového vektora a piatich vektorov váhy* 1 *sú reprezentantami tried rozkladu aj dva vektory váhy* 2*. Kód opravuje všetky chyby váhy* 1 *a dve z desiatich možných chýb váhy* 2*. Pravdepodobnosť nesprávneho dekódovania kódového slova preneseného binárnym symetrickým kanálom (s pravdepodobnosťou správneho prenosu znaku* $p = 0.99$*) je* 0.0007860898*.*

Matica štandardného rozkladu môže byť pre praktické používanie príliš veľká. Využijeme teraz to, že dekódujeme lineárny kód a na jeho dekódovanie zostrojíme podstatne menšiu tabuľku dekódovania. Predpokladajme znova, že bolo odvysielané kódové slovo $\mathbf{u}$ a že pri prenose nastala chyba $\mathbf{e}$, v dôsledku ktorej bolo prijaté slovo $\mathbf{w} = \mathbf{u} + \mathbf{e}$. Vynásobíme prijaté slovo kontrolnou maticou a dostávame

$$(\mathbf{u} + \mathbf{e})H^\top = \mathbf{u}H^\top + \mathbf{e}H^\top = \mathbf{0} + \mathbf{e}H^\top = \mathbf{s},$$

kde $\mathbf{s}$ je vektor dĺžky $n - k$, nazvaný *syndrómom chyby.* Ako sme videli, syndróm chyby nezávisí od odvysielaného kódového slova, ale len od samotného chybového vektora e. Pozrime sa teraz na triedu rozkladu s reprezentantom $\mathbf{e_i}$:

$$[\mathbf{e_i}] = \mathbf{v_0} + \mathbf{e_i}, \ldots, \mathbf{v_{q^k-1}} + \mathbf{e_i}.$$

Pre ľubovoľný vektor $\mathbf{w} = \mathbf{v_j} + \mathbf{e_i}$ z triedy $[\mathbf{e_i}]$ platí

$$(\mathbf{w})H^\top = \mathbf{v_j}H^\top + \mathbf{e_i}H^\top = \mathbf{0} + \mathbf{e_i}H^\top = \mathbf{s_i};$$

t.j. všetky vektory z triedy $[\mathbf{e_i}]$ majú rovnaký syndróm, s_i. Ak budeme používať metódu dekódovania na základe maximálnej pravdepodobnosti (prijaté slovo w dekódovať na to kódové slovo, ktoré bolo odvysielané s najväčšou pravdepodobnosťou) tak:

1. vypočítame syndróm $(\mathbf{w})H^\top = \mathbf{s_i}$,
2. v tabuľke dekódovania nájdeme reprezentanta triedy rozkladu, ktorej zodpovedá syndróm $\mathbf{s_i}$; $\mathbf{e_i}$,
3. vypočítame kódové slovo: $\mathbf{w} - \mathbf{e_i}$.

Tabuľka dekódovania, ktorú sme použili v druhom kroku má q^{n-k} riadkov a dva stĺpce; v prvom sú uvedené syndrómy chýb a v druhom im prislúchajúci reprezentanti tried rozkladu.

Príklad 8.4. *[2] Lineárny (5,2)-kód* C *z predchádzajúceho príkladu má tabuľku syndrómov*

predstaviteľ triedy rozkladu	***syndróm***
00000	000
00001	001
00010	010
00100	100
01000	101
10000	111
00011	011
00110	110

8.3 Reedove-Mullerove kódy

V tejto časti uvedieme podrobnejšie jeden špeciálny prípad lineárnych kódov, Reedove-Mullerove kódy, ktoré majú jednoduchý popis a jednoduché dekódovanie. Reedove-Mullerove kódy sú charakterizované dvoma základnými parametrami - rádom r a hodnotou $m; 0 \leq r < m$ určujúcou dĺžku kódového slova. Existujú Reedove-Mullerove kódy s rozličnými dĺžkami kódových slov a rôznymi opravnými schopnosťmi. Reedov-Mullerov kód s parametrami r, m budeme označovať symbolom $\mathcal{R}(r, m)$. V nasledujúcej tabuľke sú uvedené základné parametre kódu $\mathcal{R}(r, m)$.

$n = 2^m$	dĺžka kódu (kódového slova)
$k = \sum_{0 \leq j \leq r} \binom{m}{j}$	počet informačných symbolov
$n - k = \sum_{r < j \leq m} \binom{m}{j}$	počet kontrolných symbolov
$d = 2^{m-r}$	minimálna váha/vzdialenosť kódu

Tabuľka 8.1: Základné parametre Reedových-Mullerových kódov

Keďže Reedove-Mullerove kódy sú lineárne kódy, možno ich zadať pomocou generujúcej matice. Generujúca matica pre Reedov-Mullerov kód $\mathcal{R}(r, m)$ má zvláštny tvar:

$$G = \begin{bmatrix} G_0 \\ G_1 \\ \vdots \\ G_r \end{bmatrix}$$

Aby sme mohli popísať konštrukciu generujúcej matice G kódu $\mathcal{R}(r, m)$, zavedieme operáciu súčinu vektorov. Nech sú $u = (a_1, \ldots, a_n)$ a $v = (b_1, \ldots, b_n)$ dva vektory vektorového priestoru V. Súčinom (pozor, nejedná sa ani o vektorový ani o skalárny súčin vektorov) vektorov u, v nazveme vektor $uv = (a_1 b_1, \ldots, a_n b_n)$. (Ide o súčin vektorov po

zložkách; v binárnom prípade môžeme pomocou konvencie jazyka C zapísať súčin vektorov u, v nasledovne $uv = u\&v$.) Podmatice $G_0, \ldots, G_r$ generujúcej matice G sú definované nasledovne:

1. G_0 je jednotkový vektor dĺžky 2^m;
2. G_1 je matica typu $m \times 2^m$, ktorej stĺpcami sú všetky možné binárne vektory dĺžky m;
3. G_l, $1 \leq l \leq r$ je binárna matica typu $\binom{m}{l} \times 2^m$; riadkami podmatice G_l sú všetky vektory, ktoré sú výsledkom súčinu l vektorov z matice G_1.

Ilustrujeme konštrukcie generujúcej matice Reedových-Mullerových kódov na príklade $\mathcal{R}(3,4)$.

Príklad 8.5.

$$G = \begin{bmatrix} G_0 = & \begin{bmatrix} 1111111111111111 \end{bmatrix} \\ G_1 = & \begin{bmatrix} 0000000011111111 \\ 0000111100001111 \\ 0011001100110011 \\ 0101010101010101 \end{bmatrix} \\ G_2 = & \begin{bmatrix} 0000000000001111 \\ 0000000000110011 \\ 0000000001010101 \\ 0000001100000011 \\ 0000010100000101 \\ 0001000100010001 \end{bmatrix} \\ G_3 = & \begin{bmatrix} 0000000000000011 \\ 0000000000000101 \\ 0000000000010001 \\ 0000000100000001 \end{bmatrix} \end{bmatrix}$$

Generujúca matica kódu $\mathcal{R}(3,4)$ je matica typu $(15,16)$. To znamená, že kód $\mathcal{R}(3,4)$ má 15 informačných a 1 kontrolný symbol. (Kód $\mathcal{R}(3,4)$ je triviálny kód s testom parity, schopným odhaľovať chyby nepárnej váhy.) Neskôr zostrojíme aj netriviálny Reedov-Mullerov kód a na ňom ilustrujeme metódy kódovania a dekódovanie informácie. Skonštruujeme $\mathcal{R}(2,4)$, Reedov-Mullerov kód rádu 2 dĺžky 16 s generujúcou maticou typu $(11,16)$:

$$G = \begin{bmatrix} G_0 \\ G_1 \\ G_2 \end{bmatrix}$$

Z konštrukcie generujúcej matice Reedovho-Mullerovho kódu vyplýva, že $\mathcal{R}(r-1,m)$ je možné zostrojiť z $\mathcal{R}(r,m)$ tak, že sa z generujúcej matice G kódu $\mathcal{R}(r,m)$ vynechá podmatica G_r. To ale znamená, že $\mathcal{R}(r-1,m) \subset \mathcal{R}(r,m)$, a teda minimálna vzdialenosť d^* kódu $\mathcal{R}(r,m)$ nemôže byť väčšia ako je minimálna vzdialenosť kódu $\mathcal{R}(r-1,m)$. Ukážeme, že

$$d^* = 2^{m-r}.$$

Každý riadok podmatice G_s generujúcej matice G kódu $\mathcal{R}(r,m)$ má váhu 2^{m-s}, $0 \leq s \leq r < m$. Keďže aj riadky generujúcej matice predstavujú kódové slová kódu $\mathcal{R}(r,m)$,

$$d^* \leq 2^{m-r}.$$

Ukážeme, že kód $\mathcal{R}(r,m)$ opravuje chyby váhy $2^{m-r-1}-1$ a teda jeho minimálna vzdialenosť nie je menšia ako $2^{m-r}-1$. Vzhľadom na to, že kód $\mathcal{R}(r,m)$ obsahuje len slová párnej váhy, z vyššie uvedeného potom vyplýva, že $d^* = 2^{m-r}$. Keďže Reedove-Mullerove kódy sú lineárne kódy, mohli by sme pre kód $\mathcal{R}(r,m)$ zostrojiť kontrolnú maticu a na dekódovanie prijatých slov použiť klasickú metódu dekódovania lineárnych kódov. Reed navrhol zvláštnu metódu dekódovania Reedových-Mullerových kódov, ktorá umožňuje rekonštruovať informačné symboly na základe prijatého slova priamo, bez toho, aby bolo potrebné vypočítať syndróm chyby a určovať vektor chýb. Popíšeme teraz Reedovu metódu dekódovania.

Nech je daný informačný vektor $\mathbf{i} = (i_0, \ldots, i_{k-1})$. Vzhľadom na štruktúru generujúcej matice kódu $\mathcal{R}(r,m)$, rozdelíme aj informačný vektor na bloky veľkosťou zodpovedajúce podmaticiam G_j generujúcej matice: $\mathbf{i} = (\mathbf{i_0}, \mathbf{i_1}, \ldots, \mathbf{i_r})$. Pripomíname, že blok $\mathbf{i_l}$ bude mať dĺžku $\binom{m}{l}$. Kódové slovo $\mathbf{u}$ zodpovedajúce informačnému vektoru $\mathbf{i}$ zostrojíme tak, že vynásobíme informačný vektor generujúcou maticou kódu:

$$\mathbf{u} = \mathbf{i}G = (\mathbf{i_0}, \mathbf{i_1}, \ldots, \mathbf{i_r}) \times \begin{bmatrix} G_0 \\ G_1 \\ \vdots \\ G_r \end{bmatrix}$$

Predpokladajme, že pri prenose kódového slova došlo ku chybám, v dôsledku ktorých bolo prijaté slovo $\mathbf{v} = \mathbf{u} + \mathbf{e}$, $\mathbf{wt}(e) < 2^{m-r-1}$. Podstata Reedovho algoritmu spočíva v tom, že sa pomocou kontrolných súm, ktorých argumentami sú symboly prijatého slova $\mathbf{v}$ určia informačné symboly bloku $\mathbf{i_r}$ a potom sa vypočíta slovo

$$\mathbf{v^{(1)}} = \mathbf{v} - \mathbf{i_r}G_r = (\mathbf{i_0}, \ldots, \mathbf{i_{r-1}}) \times \begin{bmatrix} G_0 \\ G_1 \\ \vdots \\ G_{r-1} \end{bmatrix} + \mathbf{e},$$

ktoré je „pokazeným“ kódovým slovom kódu $\mathcal{R}(r-1,m)$. Podobným spôsobom postupne určíme hodnoty informačných symbolov z blokov $\mathbf{i_{r-1}}, \ldots, \mathbf{i_1}$. Hodnotu posledného informačného bitu, $i_0 = \mathbf{i_0}$ určíme zo slova

$$\mathbf{v^{(r)}} = \mathbf{v^{(r-1)}} - \mathbf{i_1}G_1 = \mathbf{i_0} \times G_0 + \mathbf{e}.$$

Ak $i_0 = 0$, $\mathbf{v^{(r)}} = \mathbf{e}$, v opačnom prípade $\mathbf{v^{(r)}} = \overline{\mathbf{e}}$. To znamená, že ak $\mathbf{wt}(v^{(r)}) < 2^{m-1}$, $i_0 = 0$; v opačnom prípade $i_0 = 1$. Otvorenou otázkou zostáva, ako zostaviť kontrolné sumy na výpočet informačných symbolov $i_1, \ldots, i_{k-1}$. Riešenie tohto problému ilustrujeme na avizovanom príklade dekódovania kódu $\mathcal{R}(2,4)$.

Príklad 8.6. *Nech je $\mathcal{C}$ Reedov-Mullerov kód rádu 2, dĺžky 16. Generujúca matica kódu $\mathcal{C}$, doplnená kvôli názornosti o riadok obsahujúci kódové slovo a stĺpec obsahujúci informačný vektor, je uvedená v tabuľke 8.2.*

	v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}	v_{15}
i_0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
i_1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
i_2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
i_3	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
i_4	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
i_5	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
i_6	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1
i_7	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1
i_8	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1
i_9	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1
i_{10}	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1

Tabuľka 8.2: Generujúca matica kódu $\mathcal{R}(2,4)$

Nech je

$$\mathbf{i} = (1\ 0011\ 100001)$$

informačný vektor, potom kódové slovo prislúchajúce vektoru $\mathbf{i}$ *je*

$$\mathbf{u} = (1000\ 1000\ 1000\ 0111).$$

Predpokladajme, že pri prenose vznikla chyba $\mathbf{e}$ *váhy 1;*

$$\mathbf{e} = (0001\ 0000\ 0000\ 0000),$$

a bolo prijaté slovo

$$\mathbf{v} = \mathbf{u} + \mathbf{e} = (1001\ 1000\ 1000\ 0111).$$

Pri konštrukcii kontrolných súm, budeme vychádzať zo vzťahu $\mathbf{u} = \mathbf{i}G$*; t.j. zo vzťahov medzi informačnými symbolmi a symbolmi kódového slova* $\mathbf{u}$*:*

$$\begin{aligned} u_0 &= i_0 \\ u_1 &= i_0 + i_4 \\ u_2 &= i_0 + i_3 \\ u_3 &= i_0 + i_3 + i_4 + i_{10} \\ &\dots \\ u_{15} &= i_0 + i_1 + \dots + i_{10} \end{aligned}$$

Vyjadríme neznáme hodnoty informačných symbolov $i_5, \dots, i_{10}$ *pomocou známych symbolov prijatého slova* $\mathbf{v}$*:*

$$\begin{aligned} i_{10} &= v_0 + v_1 + v_2 + v_3 \\ i_{10} &= v_4 + v_5 + v_6 + v_7 \\ i_{10} &= v_8 + v_9 + v_{10} + v_{11} \\ i_{10} &= v_{12} + v_{13} + v_{14} + v_{15} \end{aligned}$$

Všimneme si, že každý symbol prijatého slova vystupuje práve v jednej kontrolnej sume. To znamená, že ak pri prenose vznikla chyba váhy 1, chyba ovplyvní výsledok práve jednej kontrolnej sumy. Hodnotu i_{10} *určíme „hlasovaním"; v našom prípade sú hodnoty kontrolných súm* $0,1,1,1$ *a teda* $i_{10} = 1$. *Podobne zostavíme kontrolné sumy pre informačné symboly* i_9, i_8, i_7, i_6, i_5 *a určíme hodnoty uvedených informačných symbolov*

$$
\begin{array}{rcl|l}
i_9 &=& v_0+v_1+v_4+v_5 = 0 & \\
i_9 &=& v_2+v_3+v_6+v_7 = 1 & i_9 = 0 \\
i_9 &=& v_8+v_9+v_{12}+v_{13} = 0 & \\
i_9 &=& v_{10}+v_{11}+v_{14}+v_{15} = 0 & \\
\hline
i_8 &=& v_0+v_2+v_4+v_6 = 0 & \\
i_8 &=& v_1+v_3+v_5+v_7 = 1 & i_8 = 0 \\
i_8 &=& v_8+v_{10}+v_{12}+v_{14} = 0 & \\
i_8 &=& v_9+v_{11}+v_{13}+v_{15} = 0 & \\
\hline
i_7 &=& v_0+v_1+v_8+v_9 = 0 & \\
i_7 &=& v_2+v_3+v_{10}+v_{11} = 1 & i_7 = 0 \\
i_7 &=& v_4+v_5+v_{12}+v_{13} = 0 & \\
i_7 &=& v_6+v_7+v_{14}+v_{15} = 0 & \\
\hline
i_6 &=& v_0+v_2+v_8+v_{10} = 0 & \\
i_6 &=& v_1+v_3+v_9+v_{11} = 1 & i_6 = 0 \\
i_6 &=& v_4+v_6+v_{10}+v_{12} = 0 & \\
i_6 &=& v_5+v_7+v_{13}+v_{15} = 0 & \\
\hline
i_5 &=& v_0+v_4+v_8+v_{12} = 1 & \\
i_5 &=& v_1+v_5+v_9+v_{13} = 1 & i_5 = 1 \\
i_5 &=& v_2+v_6+v_{10}+v_{14} = 1 & \\
i_5 &=& v_3+v_7+v_{11}+v_{15} = 0 &
\end{array}
$$

Blok $\mathbf{i_2} = (100001)$ *informačných symbolov vynásobíme podmaticou* G_2 *generujúcej matice* G *a výsledok odpočítame od prijatého slova* $\mathbf{v}$*:*

$$
\begin{array}{rcl}
\mathbf{v} & = & 1001\ 1000\ 1000\ 0111 \\
\mathbf{i_2}G_2 & = & 0001\ 0001\ 0001\ 1110 \\
\mathbf{v^{(1)}} = \mathbf{v} - \mathbf{i_2}G_2 & = & 1000\ 1001\ 1001\ 1001
\end{array}
$$

Pripomíname, že slovo $\mathbf{v^{(1)}}$ *je kódové slovo kódu* $\mathcal{R}(1,4)$ *„pokazené" chybou* $\mathbf{e}$. *Zostavíme*

kontrolné sumy na výpočet informačných symbolov $i_1, \ldots, i_4$*:*

$i_4 = 1$					
$v_0 + v_1$	$=$	1	$v_2 + v_3$	$=$	0
$v_4 + v_5$	$=$	1	$v_6 + v_7$	$=$	1
$v_8 + v_9$	$=$	1	$v_{10} + v_{11}$	$=$	1
$v_{12} + v_{13}$	$=$	1	$v_{14} + v_{15}$	$=$	1
$i_3 = 1$					
$v_0 + v_2$	$=$	1	$v_1 + v_3$	$=$	0
$v_4 + v_6$	$=$	1	$v_5 + v_7$	$=$	1
$v_8 + v_{10}$	$=$	1	$v_9 + v_{11}$	$=$	1
$v_{12} + v_{14}$	$=$	1	$v_{13} + v_{15}$	$=$	1
$i_2 = 0$					
$v_0 + v_4$	$=$	0	$v_1 + v_5$	$=$	0
$v_2 + v_6$	$=$	0	$v_3 + v_7$	$=$	1
$v_8 + v_{12}$	$=$	0	$v_9 + v_{13}$	$=$	0
$v_{10} + v_{14}$	$=$	0	$v_{11} + v_{15}$	$=$	0
$i_1 = 0$					
$v_0 + v_8$	$=$	0	$v_1 + v_9$	$=$	0
$v_2 + v_{10}$	$=$	0	$v_3 + v_{11}$	$=$	1
$v_4 + v_{12}$	$=$	0	$v_5 + v_{13}$	$=$	0
$v_6 + v_{14}$	$=$	0	$v_7 + v_{15}$	$=$	0

Vynásobíme blok $\mathbf{i_1} = (0011)$ *informačných symbolov podmaticou* G_1 *generujúcej matice kódu* $\mathcal{C}$ *a výsledok odčítame od vektora* $\mathbf{v^{(1)}}$*:*

$$\begin{aligned} \mathbf{v^{(1)}} &= 1000\ 1001\ 1001\ 1001 \\ \mathbf{i_1}G_1 &= 0110\ 0110\ 0110\ 0110 \\ \mathbf{v^{(2)}} = \mathbf{v^{(1)}} - \mathbf{i_1}G_1 &= 1110\ 1111\ 1111\ 1111 \end{aligned}$$

Napokon určíme i_0*. Váha* $\mathbf{wt}(v^{(2)}) = 15 > 8$*, a to znamená, že* $i_0 = 1$*.*

Kapitola 9

Cyklické kódy

Lineárne kódy, ktoré sme študovali v predchádzajúcej časti, boli príkladom samoopravných kódov, ktoré sa dali efektívne konštruovať a pre ktoré existovali efektívne metódy kódovania a (aspoň principiálne efektívne metódy) dekódovania. Pri štúdiu lineárnych kódov sme zaviedli základné parametre samoopravných kódov (opravná schopnosť, minimálna vzdialenosť, prenosová rýchlosť a i.) a určili vzťahy medzi nimi. Z praktického hľadiska je však najmä dekódovanie dlhších lineárnych kódov priestorovo náročné. Preto je potrebné hľadať iné triedy samoopravných kódov, ktoré by zachovávali dobré vlastnosti lineárnych kódov a vyznačovali sa aj výpočtovo efektívnymi metódami dekódovania. Cyklické kódy, ktoré sú podtriedou lineárnych kódov, vďaka silnejšej algebraickej štruktúre čiastočne spĺňajú uvedené požiadavky.

Definícia 9.0.1. *Lineárny kód $\mathcal{C}$ nazveme* cyklickým kódom*, ak pre ľubovoľné kódové slovo $\mathbf{u} = (u_0, u_1, \ldots, u_{n-1}) \in \mathcal{C}$ platí $\mathbf{u}' = (u_{n-1}, u_0, u_1, \ldots, u_{n-2}) \in \mathcal{C}$.*

Názov *cyklický kód* vyplýva z toho, že operácia na slovách

$$(u_0, u_1, \ldots, u_{n-1}) \to (u_{n-1}, u_0, u_1, \ldots, u_{n-2})$$

predstavuje cyklický posun kódového slova. Na cyklické kódy sa môžeme teda dívať ako na lineárne podpriestory vektorového priestoru $GF(q)^n$ spĺňajúce dodatočnú podmienku na uzavretosť vzhľadom na cyklickú posun kódových slov. Z hľadiska konštrukcie, kódovania ale najmä dekódovania bude efektívnejšia polynomická reprezentácia cyklických kódov; t.j. reprezentácia kódových slov z $\mathcal{C} \subset GF(q)^n$ pomocou polynómov z faktorového okruhu $GF(q)[x]/x^n - 1$.

Príklad. Uvažujme Hammingov $(15, 11)$ kód, ktorý sme zaviedli v časti 7.4 s kontrolnou maticou

$$H = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Permutáciou stĺpcov kontrolnej matice H dostaneme kontrolnú maticu H' kódu, ktorý

je ekvivalentný pôvodnému Hammingovmu $(15, 11)$ kódu:

$$H' = H = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Stĺpce kontrolnej matice H' môžeme chápať dvojako: buď ako vektory dĺžky 4 nad poľom $GF(2)$, alebo ako prvky poľa $GF(2^4)$. Nech je α primitívny prvok poľa $GF(2^4)$, potom stĺpce kontrolnej matice H' môžeme vyjadriť pomocou mocnín prvku α nasledovne:

$$H' = \left[\alpha^0, \alpha^1, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6, \alpha^7, \alpha^8, \alpha^9, \alpha^{10}, \alpha^{11}, \alpha^{12}, \alpha^{13}, \alpha^{14}\right].$$

Ako bude vyzerať dekódovanie? Nech $\mathbf{u} = (u_0, u_1, \ldots, u_{14}) \in \mathcal{C}$ je kódové slovo. Potom

$$\mathbf{u}H'^{\top} = 0.$$

Posledný vzťah môžeme rozpísať nasledovne:

$$\begin{aligned} \mathbf{u}H'^{\top} &= u_0\alpha^0 + u_1\alpha^1 + u_2\alpha^2 + u_3\alpha^3 + u_4\alpha^4 + u_5\alpha^5 + u_6\alpha^6 + u_7\alpha^7 + u + 8\alpha^8 + u_9\alpha^9 + \\ &+ u_{10}\alpha^{10} + u_{11}\alpha^{11} + u_{12}\alpha^{12}\alpha^{13} + u_{14}\alpha^{14} = 0. \end{aligned}$$

Zavedieme teraz prirodzenú korešpondenciu medzi kódovými slovami Hammingovho $(15, 11)$ kódu a polynómami z okruhu polynómov $GF(2)[x]/x^{15} - 1$:

$$(u_0, u_1, \ldots, u_{14}) \leftrightarrow u_0 + u_1x + u_2x^2 + \cdots + u_{14}x^{14},$$

resp. vo všeobecnom prípade vektor $\mathbf{v} = v_0, \ldots, v_{n-1}$ nad poľom $GF(q)$ budeme reprezentovať polynómom $v_0 + v_1x + \cdots + v_{n-1}x^{n-1}$ z okruhu polynómov $GF(q)[x]/x^n - 1$. Vráťme sa ku príkladu. Násobenie vektora $\mathbf{v}$ kontrolnou maticou H' predstavuje vyčíslenie hodnoty polynómu $v(x)$ v prvku $\alpha \in GF(2^4)$. Je zrejmé, že prijaté slovo $\mathbf{v}$ je kódovým slovom Hammingovho $(15, 11)$ kódu práve vtedy, ak $v(\alpha) = 0$. Polynómy zodpovedajúce kódovým slovám, budeme nazývať *kódovými polynómami*.

V predchádzajúcom prípade sme od kontrolnej matice nad poľom $GF(q)$ prešli k takej reprezentácii kontrolnej matice, v ktorej celému vektoru-stĺpcu zodpovedal jeden prvok nejakého rozšírenia pôvodného poľa (prvok poľa $GF(q^m)$). Toto však nie je jediná možnosť, ako vyjadriť kontrolnú maticu ako maticu nad rozšírením pôvodného poľa. Predpokladajme, že je daná kontrolná matica H typu $(n - k) \times n$ nad poľom $GF(q)$ a číslo $(n - k)$ je deliteľné m; t.j. $(n - k) = mr$. Vektor-stĺpec dĺžky $(n - k)$ môžeme rozbiť na r blokov dĺžky m a každý blok reprezentovať prvkom poľa $GF(q^m)$:

Kontrolná matica H sa potom transformuje na nasledujúcu maticu:

$$H = \begin{bmatrix} \beta_{11} & \beta_{12} & \ldots & \beta_{1n} \\ \beta_{21} & \beta_{22} & \ldots & \beta_{2n} \\ \vdots & \vdots & \ldots & \vdots \\ \beta_{r1} & \beta_{r2} & \ldots & \beta_{rn} \end{bmatrix};$$

kde $\beta_{ij} \in GF(q^m), i = 1\ldots, r; j = 1, \ldots, n$. Namiesto pôvodnej kontrolnej matice typu $(n-k) \times n$ nad poľom $GF(q)$ dostávame kontrolnú maticu typu $r \times n$ nad poľom $GF(q^m)$; kde $r = (n-k)/m$. Aby sme pri dekódovaní mohli nahradiť násobenie prijatého slova u kontrolnou maticou dosadzovaním prvkov poľa $GF(q^m)$ do polynómu $u(x)$, budeme kontrolnú maticu H zapisovať v tvare

$$H = \begin{bmatrix} \gamma_1^0 & \gamma_1^1 & \cdots & \gamma_1^{n-1} \\ \gamma_2^0 & \gamma_2^1 & \cdots & \gamma_2^{n-1} \\ \vdots & \vdots & \cdots & \vdots \\ \gamma_r^0 & \gamma_r^1 & \cdots & \gamma_r^{n-1} \end{bmatrix};$$

kde $\gamma_1, \ldots, \gamma_r \in GF(q^m)$. Parametre n, q, m nie sú celkom nezávislé. Na začiatok budeme predpokladať, že $n = q^m - 1$, neskôr ukážeme, aké ďalšie hodnoty môže dĺžka kódu nadobúdať.

Dekódovanie prijatého slova $\mathbf{u} = (u_0, u_1, \ldots, u_{n-1})$ možno realizovať dvojako: prijaté slovo vynásobíme kontrolnou maticou H v ktorej prvky γ_i^j nahradíme príslušnými vektormi dĺžky m nad poľom $GF(q)$ a vypočítame syndróm s;

$$s = \mathbf{u}H^\top.$$

V druhom prípade násobeniu prijatého vektora (slova) $\mathbf{u}$ kontrolnou maticou H nad poľom $GF(q^m)$ zodpovedá dosadzovanie prvkov $\gamma_1, \ldots, \gamma_r \in GF(q^m)$ do polynómu $u(x)$:

$$\mathbf{u}H^\top = \begin{array}{rcl} u_0\gamma_1^0 + u_1\gamma_1^1 + \cdots + u_{n-1}\gamma_1^{n-1} & = & u(\gamma_1), \\ u_0\gamma_2^0 + u_1\gamma_2^1 + \cdots + u_{n-1}\gamma_2^{n-1} & = & u(\gamma_2), \\ \vdots & & \vdots \\ u_0\gamma_r^0 + u_1\gamma_r^1 + \cdots + u_{n-1}\gamma_r^{n-1} & = & u(\gamma_2 r). \end{array}$$

Podmienka $\mathbf{u}H^\top = 0$ je ekvivalentná tomu, že prvky $\gamma_1, \ldots, \gamma_r \in GF(q^m)$ sú korene polynómu $u(x)$. Ilustrujeme uvedenú konštrukciu na príklade.

Príklad 9.1. *Nech je α primitívny prvok poľa $GF(2^4)$, nech $n = 15$. Položíme $\gamma_1 = \alpha, \gamma_2 = \alpha^3$ a zostrojíme kontrolnú maticu H $(15, 7)$-kódu:*

$$H = \begin{bmatrix} \alpha^0 & \alpha^1 & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 & \alpha^7 & \alpha^8 & \alpha^9 & \alpha^{10} & \alpha^{11} & \alpha^{12} & \alpha^{13} & \alpha^{14} \\ \alpha^0 & \alpha^3 & \alpha^6 & \alpha^9 & \alpha^{12} & \alpha^{15} & \alpha^{18} & \alpha^{21} & \alpha^{24} & \alpha^{27} & \alpha^{30} & \alpha^{33} & \alpha^{36} & \alpha^{39} & \alpha^{42} \end{bmatrix}$$

Prvky poľa $GF(2^4)$ môžeme reprezentovať pomocou binárnych vektorov dĺžky 4 (pozri tabuľku 15.4). Kontrolná matica bude potom binárnou maticou typy 8×15:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Pri štúdiu Bose-Chandhury-Hocquenghemových (BCH) kódov v nasledujúcej podkapitole ukážeme, že kód z predchádzajúceho príkladu je BCH kód $(15, 7)$ opravujúci chyby váhy 2.

9.1 Polynomický popis cyklických kódov

V tejto časti popíšeme najprv algebraickú štruktúru cyklických kódov a potom ukážeme, ako na základe týchto poznatkov možno konštruovať cyklické kódy. Budeme uvažovať cyklický kód $\mathcal{C}$ dĺžky n nad poľom $GF(q)$. Pripomenieme, že cyklický kód $\mathcal{C}$ je lineárnym podpriestorom vektorového priestoru dimenzie n nad poľom $GF(q)$ a že $\mathcal{C}$ je uzavretý na cyklický posun svojich prvkov. Vektorový priestor $GF(q)^n$ možno prirodzeným spôsobom zobraziť na faktorový okruh polynómov $GF(q)[x]/x^n - 1$:

$$\forall \mathbf{u} \in GF(q)^n;\ \mathbf{u} = (u_0, u_1, \ldots, u_{n-1}) \leftrightarrow u_0 + u_1 x + u_2 x^2 + \cdots + u_{n-1} x^{n-1}$$

Dá sa ľahko overiť, že faktorový okruh polynómov $GF(q)[x]/x^n - 1$ má vlastnosti vektorového priestoru. Naviac, vo faktorovom okruhu $GF(q)[x]/x^n - 1$ je definované (modulárne) násobenie polynómov:

$$\forall a(x), b(x) \in GF(q)[x]/x^n - 1 : a(x) \cdot b(x) = a(x) \cdot b(x) \pmod{x^n - 1}.$$

Zostáva nájsť vyjadrenie cyklického posunu pomocou operácií nad polynómami v okruhu $GF(q)[x]/x^n - 1$. Cyklický posun vektora $\mathbf{u}$ zodpovedá súčinu polynómov $x \cdot u(x)$. Skutočne,

$$x \cdot u(x) \quad = \quad u_{n-1} x^n + u_{n-2} x^{n-1} + \cdots + u_1 x^2 + u_0 x \pmod{x^n - 1},$$

pričom

$$\begin{array}{rlllll} & u_{n-1}x^n & +u_{n-2}x^{n-1}+ \quad \ldots \quad +u_1x^2 + u_0x & & : \ (x^n - 1) = u_{n-1} \\ - & u_{n-1}x^n & & +u_{n-1} & & \\ \hline & & u_{n-2}x^{n-1}+ \quad \ldots \quad +u_1x^2 + u_0x + u_{n-1} & & & \end{array}$$

Cyklický kód môžeme teraz charakterizovať nasledujúcim spôsobom:

Veta 9.1.1. *Nech* $GF(q)[x]/x^n - 1$ *je faktorový okruh polynómov nad poľom* $GF(q)$. *Podmnožina polynómov* $\mathcal{C} \subset GF(q)[x]/x^n - 1$ *tvorí cyklický kód práve vtedy, ak*

1. $\mathcal{C}$ *je aditívna podgrupa okruhu* $GF(q)[x]/x^n - 1$,
2. *ak* $u(x) \in \mathcal{C}$ *a* $a(x) \in GF(q)[x]/x^n - 1$, *tak*

$$a(x) \cdot u(x) \pmod{x^n - 1} \in \mathcal{C}.$$

Dôkaz. Cyklický kód je zároveň lineárnym kódom. To znamená, že $\mathcal{C}$ tvorí aditívnu podgrupu okruhu $GF(q)[x]/x^n - 1$. Z lineárnosti kódu $\mathcal{C}$ vyplýva, že aj súčin ľubovoľného prvku $a \in GF(q)$ a polynómu $u(x) \in \mathcal{C}$ patrí do $\mathcal{C}$. Z cyklickosti $\mathcal{C}$ vyplýva, že pre $u(x) \in \mathcal{C}$ aj $x^k \cdot u(x) \in \mathcal{C}$. To znamená, že pre ľubovoľný polynóm $a(x) \in GF(q)[x]/x^n - 1; a(x) = a_0 + a_1x + \cdots + a_{n-1}x^{n-1}$ patria aj polynómy $a_0 \cdot u(x), a_1x \cdot u(x), \ldots, a_{n-1}x^{n-1} \cdot u(x)$ do $\mathcal{C}$ a z toho, že $\mathcal{C}$ je aditívna grupa vyplýva, že aj

$$a_0 \cdot u(x) + a_1x \cdot u(x) + \cdots + a_{n-1}x^{n-1} \cdot u(x) = a(x) \cdot u(x) \in \mathcal{C}.$$

Opačne, nech $\mathcal{C}$ spĺňa uvedené dve podmienky. Z prvej vyplýva, že $\mathcal{C}$ je uzavretá na sčítanie a z druhej, že $\mathcal{C}$ je uzavretá na násobenie skalárom (ostatné vlastnosti násobenia a sčítania sa do $\mathcal{C}$ prenášajú z okruhu $GF(q)[x]/x^n - 1$). To znamená, že $\mathcal{C}$ tvorí lineárny podpriestor (lineárny kód). Z druhej podmienky naviac vyplýva, že $\mathcal{C}$ je uzavretá na cyklický posun; t.j. $\mathcal{C}$ je cyklický kód. □

Poznámka. Množina $\mathcal{C}$ tvorí ideál okruhu $GF(q)[x]/x^n - 1$. Neskôr dokážeme, že $\mathcal{C}$ je hlavný ideál.

Preskúmame teraz množinu $\mathcal{C}$ podrobnejšie. Budeme v nej hľadať polynóm $g(x)$, ktorý je nenulový, normovaný a má spomedzi všetkých (nenulových) prvkov-polynómov množiny $\mathcal{C}$ minimálny stupeň. Keďže $\mathcal{C}$ je neprázdna množina, nenulový normovaný polynóm $g(x)$ minimálneho stupňa existuje. Ukážeme, že je daný jednoznačne. Predpokladajme opak, t.j. že v $\mathcal{C}$ existujú dva rôzne polynómy $g_1(x)$ a $g_2(x)$ požadovaných vlastností. Potom aj ich rozdiel, polynóm $g_1(x) - g_2(x)$ patrí do $\mathcal{C}$. Ale polynómy $g_1(x)$ a $g_2(x)$ majú ten istý stupeň a sú normované. To znamená, že ich rozdiel je nenulový polynóm nižšieho stupňa, z ktorého po vydelení koeficientom pri najvyššej mocnine možno vytvoriť normovaný nenulový polynóm nižšieho stupňa ako je stupeň polynómov $g_1(x)$ a $g_2(x)$. Spor. Polynóm $g(x)$ je teda daný jednoznačne a z dôvodu, ktorý zakrátko uvedieme, nazýva sa generujúcim polynómom kódu $\mathcal{C}$ Preskúmame vzťah polynómu $g(x)$ a ostatných prvkov (polynómov) kódu $\mathcal{C}$. Nech je $u(x)$ nenulový kódový polynóm kódu $\mathcal{C}$. Potom existujú polynómy $q(x)$ a $r(x)$ také, že

$$u(x) = q(x)g(x) + r(x) \mod x^n - 1,$$

pričom $\deg(r(x) < \deg(g(x))$. Keďže $g(x) \in \mathcal{C}$, tak potom podľa vety 9.1.1 aj $q(x)g(x) \in \mathcal{C}$. To však znamená, že aj

$$u(x) - q(x)g(x) \in \mathcal{C},$$

nakoľko ide o rozdiel dvoch kódových slov a $\mathcal{C}$ je cyklický a teda aj lineárny kód. Ale

$$u(x) - q(x)g(x) = r(x)$$

a $\deg(r(x) < \deg(g(x))$. To by znamenalo, že v kóde $\mathcal{C}$ existuje polynóm nižšieho stupňa ako je stupeň generujúceho polynómu. To však je možné len v prípade, keď $r(x) = 0$. Potom však pre ľubovoľný kódový polynóm $u(x)$ z $\mathcal{C}$ platí

$$u(x) = q(x)g(x).$$

Tým sme dokázali nasledujúcu vetu

Veta 9.1.2. *Nech* $\mathcal{C} \subseteq GF(q)[x]/x^n - 1$ *je cyklický kód dĺžky* n. *Potom v kóde* $\mathcal{C}$ *existuje jediný nenulový normovaný polynóm* $g(x)$ *stupňa* $n - k$ *taký, že*

$$\mathcal{C} = \{a(x)g(x); \quad a(x) \in GF(q)[x],\ \deg(g(x)) < k\}.$$

Keďže cyklický kód je jednoznačne zadaný svojím generujúcim polynómom, ponúkajú sa prirodzené otázky, či sa tento vzťah nedá využiť pri konstrukcii, kódovaní i dekódovaní cyklických kódov a ak áno, ako. Odpoveď na tieto otázky budeme hľadať v nasledujúcej časti tejto kapitoly. Začneme základnou otázkou: aké vlastnosti musí mať polynóm, aby bol generujúcim polynómom cyklického kódu? Odpoveď na táto otázku dáva nasledujúca veta.

Veta 9.1.3. *Cyklický kód* $\mathcal{C}$ *dĺžky* n *s generujúcim polynómom* $g(x)$ *existuje práve vtedy, ak* $g(x)|x^n - 1$.

Dôkaz. Predpokladajme, že $\mathcal{C} \subseteq GF(q)[x]/x^n - 1$ je cyklický kód dĺžky n s generujúcim polynómom $g(x)$. Platí

$$x^n - 1 = q(x)g(x) + r(x),$$

kde $q(x), r(x) \in GF(q)[x]/x^n - 1$ a $\deg(r(x)) < \deg(g(x))$. Keďže platí

$$x^n - 1 = 0 \mod x^n - 1$$

a $\deg(r(x)) < n$, tak potom

$$0 = (q(x)g(x) + r(x)) \mod x^n - 1 = (q(x)g(x)) \mod (x^n - 1) + r(x).$$

Kód $\mathcal{C}$ je cyklický a $g(x)$ je jeho generujúci polynóm, preto aj $(q(x)g(x)) \mod x^n - 1$ je kódový polynóm kódu $\mathcal{C}$. Keďže 0 je tiež kódový polynóm, potom aj $r(x) = 0 - q(x)g(x)$ musí byť kódovým polynómom kódu $\mathcal{C}$. To je však možné len v prípade, keď $r(x) = 0$, a teda

$$x^n - 1 = q(x)g(x).$$

Nech opačne $g(x) \in GF(q)[x]$, $g(x)|x^n - 1$, $\deg(g(x)) = n - k$. Množina polynómov $\mathcal{C} = \{a(x)g(x); \quad a(x) \in GF(q)[x],\ \deg(g(x)) < k\}$ je podľa vety 9.1.1 cyklickým kódom. □

To, že polynóm $g(x)$ delí polynóm $x^n - 1$ znamená, že existuje polynóm, označíme ho ako $h(x)$, taký, že $x^m - 1 = g(x)h(x)$. Ak je $g(x)$ generujúcim polynómom cyklického kódu $\mathcal{C}$, tak potom sa polynóm $h(x)$ nazýva *kontrolným polynómom cyklického kódu* $\mathcal{C}$. Podobne ako generujúci polynóm, zohráva aj kontrolný polynóm v teórii lineárnych kódov dôležitú úlohu.

Pozrime sa teraz na polynóm $x^n - 1$ nad poľom $GF(q)$. Vyjadríme $x^n - 1$ ako súčin ireducibilných polynómov nad poľom $GF(q)$:

$$x^n - 1 = f_1(x) \cdot f_2(x) \cdot \cdots \cdot f_l(x).$$

Generujúci polynóm $g(x)$ cyklického kódu dĺžky n nad poľom $GF(q)$ sa dá potom vyjadriť ako súčin vybraných ireducibilných polynómov z rozkladu $x^n - 1$:

$$g(x) = f_{i_1}(x) \cdot f_{i_2}(x) \ldots f_{i_j}(x).$$

Keďže polynóm $x^n - 1$ má l ireducibilných faktorov nad poľom $GF(q)$, existuje 2^l rozličných generujúcich polynómov a práve toľko cyklických kódov dĺžky n nad poľom $GF(q)$. Cyklický kód však tvoria kódové polynómy, ktoré sú násobkami generujúceho polynómu; $u(x) = a(x)g(x)$. Kódový polynóm má stupeň nanajvýš $n - 1$. Ak bude mať generujúci polynóm stupeň $n - k$, tak potom kód bude obsahovať 2^k kódových slov. Z toho vyplýva, že niektoré polynómy nebudú generovať použiteľné kódy. Ilustrujeme vytváranie cyklických kódov pomocou generujúcich polynómov na príklade [1].

Príklad. Uvedieme všetky binárne cyklické kódy dĺžky 7. Polynóm $x^7 - 1$[1] má tri ireducibilné faktory[2]

$$x^7 - 1 = (x + 1)(x^3 + x + 1)(x^3 + x^2 + 1).$$

To znamená, že teoreticky existuje 8 binárnych cyklických kódov dĺžky 7.

generujúci polynóm	kontrolný polynóm	kód
1	$x^7 - 1$	$\mathcal{C}_1$
$x + 1$	$(x^3 + x + 1)(x^3 + x^2 + 1)$	$\mathcal{C}_2$
$(x^3 + x + 1)$	$(x + 1)(x^3 + x^2 + 1)$	$\mathcal{C}_3$
$(x^3 + x^2 + 1)$	$(x + 1)(x^3 + x + 1)$	$\mathcal{C}_4$
$(x + 1)(x^3 + x + 1)$	$(x^3 + x^2 + 1)$	$\mathcal{C}_5$
$(x + 1)(x^3 + x^2 + 1)$	$(x^3 + x + 1)$	$\mathcal{C}_6$
$(x^3 + x + 1)(x^3 + x^2 + 1)$	$(x + 1)$	$\mathcal{C}_7$
$x^7 - 1$	1	$\mathcal{C}_8$

- kód $\mathcal{C}_1$ pozostáva zo všetkých binárnych vektorov dĺžky 7, jeho opravná schopnosť je nulová.
- kód $\mathcal{C}_2$ je kód s testom na paritu a má 64 kódových slov dĺžky 7 (párnej váhy)
- kódy $\mathcal{C}_3$ a $\mathcal{C}_4$ sú Hammingove (7,4) kód opravujúce jednu chybu. Majú 16 kódových slov
- kódy $\mathcal{C}_5$ a $\mathcal{C}_6$ sú duálne kódy k Hamingovému (7,4) kódu, majú 8 kódových slov
- kód $\mathcal{C}_7$ má dve kódové slová $0000000, 1111111$, opravuje 3 chyby a nazýva sa *kód s opakovaním (repetition code)*.
- kód $\mathcal{C}_8$ má jediné kódové slovo 0000000 a je prakticky nepoužiteľný.

9.2 Maticový popis cyklických kódov

Cyklické kódy sú podmnožinou lineárnych kódov. Lineárne kódy je možné zadať pomocou generujúcej, resp. kontrolnej matice. Jeden typ kontrolnej matice pre cyklické kódy

[1] pripomíname, že nad poľom $GF(2)$ $x^7 - 1 = x^7 + 1$

[2] o ireducibilite polynómov $(x^3 + x + 1), (x^3 + x^2 + 1)$ sa presvedčíme ľahko; ak by totiž boli reducibilné, museli by mať aspoň jeden faktor stupňa 1, a teda ako koreň prvok z poľa $GF(2)$.

sme už uviedli v úvode tejto kapitoly. Medzitým sme zistili, že cyklické kódy je možné popísať pomocou generujúcich a kontrolných polynómov. Ukážeme teraz, aký je vzťah medzi generujúcim plonómom agenerujúcou maticou a kontrolným polynómom a kontrolnou maticou cyklického kódu. Uvažujme cyklický kód $\mathcal{C}$ dĺžky n nad poľom $GF(q)$ s generujúcim polynómom $g(x)$ stupňa $n-k$ a kontrolným polynómom $h(x)$. Keďže cyklický kód $\mathcal{C}$ pozostáva zo všetkých násobkov generujúceho polynómu $g(x)$, kódovými polynómami budú aj polynómy $g(x), x\cdot g(x), \ldots, x^{k-1}\cdot g(x)$. Ukážeme, že polynómy $g(x), x\cdot g(x), \ldots, x^{k-1}\cdot g(x)$, resp. vektory ich koeficientov sú lineárne nezávislé. Predpokladajme opak, potom by musela existovať k-tica prvkov $a_0, \ldots, a_{k-1} \in GF(q)$, s aspoň jedným nenulovým prvkom taká, že (v polynomickom vyjadrení)

$$a_0 \cdot g(x) + a_1 \cdot x \cdot g(x) + \cdots + a_{k-1} \cdot x^{k-1} \cdot g(x) = 0 \tag{9.1}$$

Zo vzťahu 9.1 však vyplýva, že

$$(a_0 + a_1 \cdot x + \cdots + a_{k-1} \cdot x^{k-1}) \cdot g(x) = 0,$$

resp.

$$(a_0 + a_1 \cdot x + \cdots + a_{k-1} \cdot x^{k-1}) = 0,$$

čo je však v spore s nenulovosťou vektora prvkov $a_0, \ldots, a_{k-1}$. To znamená, že vektory koeficientov polynómov $g(x), x\cdot g(x), \ldots, x^{k-1}\cdot g(x)$ sú nezávislé a keďže cyklický kód $\mathcal{C}$ je zároveň lineárny kód dimenzie k, vektory koeficientov polynómov $g(x), x\cdot g(x), \ldots, x^{k-1}\cdot g(x)$ tvoria jeho bázu. Táto báza zapísaná v podobe matice predstavuje generujúcu maticu cyklického kódu $\mathcal{C}$:

$$G = \begin{bmatrix} 0 & \ldots & 0 & 0 & g_{n-k} & g_{n-k-1} & \cdots & g_2 & g_1 & g_0 \\ 0 & \ldots & 0 & g_{n-k} & g_{n-k-1} & g_{n-k-2} & \cdots & g_1 & g_0 & 0 \\ 0 & \ldots & g_{n-k} & g_{n-k-1} & g_{n-k-2} & g_{n-k-3} & \cdots & g_0 & 0 & 0 \\ \vdots & & & & & & & & & \vdots \\ g_{n-k} & \ldots & & & & & \ldots & 0 & 0 & 0 \end{bmatrix} \tag{9.2}$$

Teraz zostrojíme kontrolnú maticu cyklického kódu $\mathcal{C}$ pomocou jeho kontrolného polynómu $h(x)$. Keďže $h(x)|x^n - 1$ existuje lineárny kód $\mathcal{C}'$ s generujúcim polynómom $h(x)$. Generujúcu maticu H' kódu $\mathcal{C}'$ možno taktiež vyjadriť v tvare 9.2. Kód $\mathcal{C}'$ však nie je duálnym kódom kódu $\mathcal{C}$, pretože $G \cdot H'^{\top} \neq 0$, a teda matica H' nie je kontrolnou maticou kódu $\mathcal{C}$. Ukážeme, že kontrolná matica cyklického kódu $\mathcal{C}$ odvodená z jeho kontrolného polynómu $h(x)$ existuje a má tvar

$$H = \begin{bmatrix} 0 & \ldots & 0 & 0 & h_k & h_{k-1} & \ldots & h_2 & h_1 & h_0 \\ 0 & \ldots & 0 & h_k & h_{k-1} & h_{k-2} & \ldots & h_1 & h_0 & 0 \\ 0 & \ldots & h_k & h_{k-1} & h_{k-2} & h_{k-3} & \ldots & h_0 & 0 & 0 \\ \vdots & & & & & & & & & \vdots \\ h_k & \ldots & & & & & \ldots & 0 & 0 & 0 \end{bmatrix} \tag{9.3}$$

Potrebujeme dokázať, že $G \cdot H^\top = 0$. Využijeme skutočnosť, že $h(x) \cdot g(x) = 0 \mod x^n - 1$. To znamená, že

$$\begin{array}{lcl} g_0 h_0 & = & 1, \\ (g_0 h_1 + g_1 h_0) & = & 0, \\ (g_0 h_2 + g_1 h_1 + g_2 h_0) & = & 0, \\ \quad \cdots & & \\ (g_{n-k-1} h_k + g_{n-k} h_{k-1}) & = & 0, \\ g_{n-k} h_k & = & 1. \end{array} \tag{9.4}$$

Súčinom matíc $G \cdot H^\top$ je matica typu $(n-k) \times (n-k)$ ktorú možno zapísať v tvare

$$G \cdot H^\top = \begin{bmatrix} a_{n-1} & a_{n-2} & \dots & a_k \\ a_{n-2} & a_{n-3} & \dots & a_{k-1} \\ \vdots & & \vdots & \\ a_{n-k} & a_{n-k-1} & \dots & a_1 \end{bmatrix} \tag{9.5}$$

kde

$$a_m = \sum_{i=0}^{m} g_{m-i} h_i, \qquad 0 < m < n.$$

Zo vzťahov 9.4 vyplýva, že $a_m = 0$ pre $m = 1, \dots, n-1$ a teda matica 9.3 je kontrolnou maticou kódu $\mathcal{C}$. Ukážeme, že lineárny kód $\mathcal{C}^\perp$, duálny ku kódu $\mathcal{C}$ je zároveň cyklický kód s generujúcim polynómom $x^k \cdot h x^{-1}$. Stačí ukázať, že polynóm $x^k \cdot h x^{-1}$ delí polynóm $x^n - 1$. Vyjadríme polynóm $x^n - 1$ v podobe súčinu generujúceho a kontrolného polynómu $x^n - 1 = g(x) \cdot h(x)$, dosadíme $x := x^{-1}$ a výslednú rovnosť vynásobíme polynómom x^n. Dostávame rovnosť

$$x^n \cdot g(x^{-1}) \cdot h(x^{-1}) = x^{n-k} g(x^{-1}) \cdot c^k \cdot h(x^{-1}) = 1 - x^n.$$

Z poslednej rovnosti vyplýva, že polynómom $x^k \cdot h(x^{-1})$ delí polynóm $x^n - 1$.

Poznámka. Kód $\mathcal{C}'$ zadaný generujúcou maticou H' sa niekedy tiež nazýva duálnym kódom kódu $\mathcal{C}$. V skutočnosti sa však nejedná o duálny kód (pretože $G \cdot H'^\top \neq 0$) ale ide len o kód ekvivalentný s duálnym kódom $\mathcal{C}^\perp$.

9.3 Kódovanie pomocou cyklických kódov

Správu, ktorú potrebujeme zakódovať pomocou cyklického (n, k)-kódu $\mathcal{C}$ (s generujúcim polynómom $g(x)$ stupňa $n - k$ rozdelíme na postupnosť disjunktných blokov dĺžky k (informačné vektory). Každému informačnému vektoru $i = (i_0, \dots, i_{k-1})$ priradíme informačný polynóm $i(x) = i_0 + i_1 \cdot x + \cdots + i_{k-1} \cdot x^{k-1}$ stupňa nanajvýš $k - 1$.Keďže kódové slová (polynómy) cyklického kódu $\mathcal{C}$ sú násobkami generujúceho polynómu $g(x)$, stačí informačný polynóm vynásobiť generujúcim polynómom a dostaneme kódový polynóm kódu $\mathcal{C}$

$$u(x) = i(x) \cdot g(x).$$

Takýto spôsob kódovania je korektný, ale je nesystematický, pretože z kódového polynómu $u(x)$ sa nedá bezprostredne určiť informačný polynóm $i(x)$. Existuje aj systematický spôsob kódovania, ktorého podstata je nasledovná:

- Informačný polynóm $i(x)$ vynásobíme polynómom x^{n-k}.
- Vypočítame $x^{n-k} \cdot i(x) \mod g(x)$
- Od polynómu $x^{n-k} \cdot i(x)$ odčítame $x^{n-k} \cdot i(x) \mod g(x)$ a dostávame hľadané kódové slovo $x^{n-k} \cdot i(x) - x^{n-k} \cdot i(x) \mod g(x)$.

Keďže stupeň polynómu $x^{n-k} \cdot i(x) \mod g(x)$je nanajvýš $n-k-1$ a prvky informačného vektora tvoria v kódovom slove koeficienty pri mocninách $x^{n-k}, \ldots, x^{n-1}$, v kódovom slove sú jednoznačne oddelené informačné a „kontrolné“ symboly tak, ako sme požadovali, obr. 9.1.

$i(x)$	$-x^{n-k} \cdot i(x) \mod g(x)$

Obr. 9.1: Kódové slovo systematického cyklického kódu

9.4 Dekódovanie cyklických kódov

Keďže cyklické kódy sú podmožinou lineárnych kódov, možno na ich dekódovanie použiť tie isté metódy ako na dekódovanie lineárnych kódov. Pri dekódovaní lineárnych kódov s väčšou opravnou schopnosťou sme narážali na to, že si bolo potrebné pamätať rozsiahlu dekódovaciu tabuľku (obsahujúcu zoznam syndrómov chýb a im prislúchajúcich chybových vektorov). Využijeme algebraickú štruktúru cyklických kódova na zostrojenie efektívnejšieho algoritmu dekódovania. Predpokladajme, že informácia, ktorú spracovávame, je zapísaná vo forme polynómov. Do popisu spracovania zahrnieme aj kódovanie správ:

1. informačný vektor **i** transformujeme na informačný polynóm $i(x)$,
2. informačný polynóm (napríklad nesystematicky) transformujeme na kódový polynóm: $u(x) = i(x) \cdot g(x)$ s vektorom koeficientov **u**,
3. koeficienty kódového polynómu sa prenášajú prenosovým kanálom. Počas prenosu vznikne chyba **e**, ktorá transformuje prenášané kódové slovo na slovo $\mathbf{v} = \mathbf{u} + \mathbf{e}$. V polynomickom vyjadrení $v(x) = u(x) + e(x)$.
4. Príjemca interpretuje prijaté slovo **v** ako polynóm $v(x)$, vydelí prijatý polynóm generujúcim polynómom a vypočíta zvyšok po delení:

$$\begin{aligned} v(x) \mod g(x) &= (u(x) + e(x)) \mod g(x) = u(x) \mod g(x) + e(x) \mod g(x) = \\ &= e(x) \mod g(x) = s(x). \end{aligned}$$

Výsledkom delenia je polynóm $s(x)$, ktorý sa nazýva syndrómový polynóm. Je zrejmé, že $\deg(s(x)) < \deg(g(x))$. Keďže ľubovoľné kódové slovo $u(x)$ je násobkom generujúceho polynómu $g(x)$, $u(x) \mod g(x) = 0$ a teda syndrómový polynóm nezávisí od odvysielaného kódového polynómu, ale len od polynómu chýb.

Skôr ako budeme pokračovať vo výklade, uvedieme kvôli prehľadnosti zoznam polynómov, ktoré sa používajú pri popise, kódovaní a dekódovaní cyklického (n, k)-kódu.

názov	označenie	stupeň
generujúci polynóm	$g(x)$	$n-k$
kontrolný polynóm	$h(x)$	k
informačný polynóm	$i(x)$	$k-1$
kódový polynóm	$u(x)$	$n-1$
chybový polynóm	$e(x)$	$n-1$
prijatý polynóm	$v(x)$	$n-1$
syndrómový polynóm	$s(x)$	$n-k-1$

Tabuľka 9.1: Polynómy cyklických kódov

Aby bolo možné používať syndrómový polynóm $s(x)$ na určenie chyby $e(x)$, potrebujeme mať záruku, že syndróm chyby určuje chybu[3] jednoznačne; t.j. že neexistujú dve rôzne chyby váhy menšej alebo rovnej opravnej schopnosti kódu s tým istým syndrómom. Túto dôležitú vlastnosť cyklických kódov sformulujeme a dokážeme v nasledujúcej vete.

Veta 9.4.1. *Nech je* d *minimálna vzdialenosť cyklického kódu* $\mathcal{C}$*, potom každému polynómu chýb váhy menšej ako* $d/2$ *zodpovedá práve jeden syndrómový polynóm.*

Dôkaz. Každému chybovému polynómu $e(x)$ zodpovedá nejaký syndrómový polynóm $s(x) = e(x) \mod g(x)$. Predpokladajme, že existujú dva rozličné chybové polynómy váhy menšej ako $d/2$; $e_1(x) \neq e_2(x)$ s tým istým syndrómovým polynómom $s(x)$. To znamená, že

$$\begin{aligned} e_1(x) &= q_1(x) \cdot g(x) + s(x) \\ e_2(x) &= q_2(x) \cdot g(x) + s(x) \end{aligned}$$

ale potom je rozdiel chybových polynómov

$$e_1(x) - e_2(x) = (q_1(x) - q_2(x)) \cdot g(x) \tag{9.6}$$

násobkom generujúceho polynómu, a teda kódovým slovom. Ale polynóm $e_1(x) - e_2(x)$ má váhu $< d$, čo je v spore s minimálnou váhou kódu $\mathcal{C}$. To znamená, že $e_1(x) - e_2(x) = 0$, resp. $e_1(x) = e_2(x)$. □

Príklad. Uvažujeme $(7, 4)$ Hammingov cyklický kód s generujúcim polynómom $g(x) = x^3 + x + 1$. Tento kód má minimálnu vzdialenosť 3, a opravuje chyby váhy 1. Všetky

[3] podobne ako pri lineárnych kódoch, aj tu sa rozumie chyba váhy menšej alebo rovnej opravnej schopnosti kódu

chybové polynómy váhy nanajvýš 1 a im prislúchajúce syndrómové polynómy sú uvedené v tabuľke

chybový polynóm $e(x)$	syndrómový polynóm $s(x)$
0	0
1	1
x	x
x^2	x^2
x^3	$x+1$
x^4	x^2+x
x^5	x^2+x+1
x^6	x^2+1

Príklad spracovania (kódovanie, prenos a dekódovanie) správ pomocou Hammingovho kódu je uvedený v tabuľke

vektor/polynóm	označenie	vektor	polynóm	spôsob výpočtu
informačný	$i(x)$	1011	x^3+x+1	
kódový	$u(x)$	1000101	x^6+x^2+1	$u(x)=i(x)\cdot g(x)$
odvysielaný	$u(x)$	1000101	x^6+x^2+1	
chybový	$e(x)$	0010000	x^4	
prijatý	$v(x)$	1010101	$x^6++x^4+x^2+1$	$v(x)=u(x)+e(x)$
syndrómový	$s(x)$	110	x^2+x	$v(x) \mod g(x)$
chybový	$e(x)$	0010000	x^4	$s(x)\leftrightarrow e(x)$
opravený	$u(x)$	1000101	x^6+x^2+1	$u(x)=v(x)-e(x)$
informačný	$i(x)$	1011	x^3+x+1	$i(x)=u(x)\div g(x)$

Použitie polynomickej reprezentácie kódu umožnilo nahradiť maticové oprácie (násobenie vektorov generujúcou, resp. kontrolnou maticou) jednoduchšie realizovateľným násobením a resp. delením generujúcim polynómom, ale základný problém dekódovania lineárnych kódov—potrebu rozsiahlej dekódovacej tabuľky—nevyriešilo. Využijeme teraz silnejšiu algebraickú štruktúru cyklických kódov na návrh efektívnejšej metódy dekódovania.

Predpokladajme, že kódové slovo $u(x)$ bolo pri prenose modifikované chybou $e(x)$ a v dôsledku toho bolo prijaté slovo $v(x)=u(x)+e(x)$. Základná myšlienka dekódovania spočíva v tom, že pri cyklickom posune prijatého slova sa súčasne posúva kódové slovo aj chyba:

$$x\cdot v(x)=x\cdot u(x)+x\cdot e(x) \mod x^n-1.$$

To znamená, že po istom počte cyklických posunov prijatého slova dostaneme slovo, ktoré bude predstavovať kódové slovo (kódový polynóm), v ktorom bude chybou modifikovaný koeficient pri mocnine x^{n-1} (a možno aj niektoré iné koeficienty). Ak by sme túto chybu dokázali odhaliť a opraviť, potom by stačilo cyklicky posunúť opravené slovo o patričný počet miest, aby sme dostali pôvodne odvysielané kódové slovo $u(x)$. Čo potrebujeme na to, aby sme dokázali odhaliť chybu v koeficiente pri najvyššej mocnine prijatého polynómu? Stačilo by na to mať zoznam syndrómov chýb zodpovedajúcich všetkým (daným kódom opraviteľným) kódovým slovám $e(x),\ e_{n-1}\neq 0$. Potom bude stačiť posúvať prijaté

slovo, vyčísľovať v každom kroku syndróm chyby a porovnávať ho s redukovanou tabuľkou syndrómov. Na tejto myšlienke je postavený Meggitov algoritmus dekódovania cyklických kódov, ktorý bol publikovaný v roku 1960. Meggitov algoritmus naviac využíva tú skutočnosť, že na výpočet postupnosti syndrómov polynómov $v(x), x \cdot v(x), \ldots, x^{n-1} \cdot v(x)$ nie je potrebné počítať $((x^j \cdot v(x) \mod x^n - 1) \mod g(x)$, $j = 0, \ldots, n-1$, ale stačí počítať $(x^j \cdot s(x) \mod g(x)$. Toto zjednodušenie sa zakladá na tvrdení nasledujúcej vety.

Veta 9.4.2. *Nech je daný cyklický kód $\mathcal{C}$ s generujúcim polynómom $g(x)$ a nech je $v(x)$ polynóm prijatý po odvysielaní (nejakého) kódového slova kódu $\mathcal{C}$. Nech je $s(x)$ syndrómový polynóm polynómu $v(x)$, potom má polynóm $x \cdot v(x) \mod x^n - 1$ (cyklický posun prijatého slova) syndrómový polynóm $x \cdot s(x) \mod g(x)$.*

Dôkaz Nech je $v(x) = v_0 + v_1 \cdot x + \cdots + v_{n-1} \cdot x^{n-1}$ prijatý polynóm. Platí

$$v(x) = q(x) \cdot g(x) + s(x).$$

Cyklický posun prijatého slova je (v polynomickom vyjadrení):

$$\begin{aligned}
& x \cdot v(x) \mod x^n - 1 = v_{n-1} + v_0 \cdot x + v_1 \cdot x^2 + \cdots + v_{n-2} \cdot x^{n-1} = \\
= \; & x \cdot v(x) - v_{n-1} \cdot (x^n - 1) = x \cdot v(x) - v_{n-1} \cdot g(x)h(x) = \\
= \; & x \cdot (q(x) \cdot g(x) + s(x)) - v_{n-1} \cdot g(x)h(x) = x \cdot s(x) + x \cdot q(x) \cdot g(x) - v_{n-1} \cdot g(x)h(x) = \\
= \; & x \cdot s(x) + g(x) \cdot (x \cdot q(x) - v_{n-1} \cdot h(x)) = x \cdot s(x) \mod g(x).
\end{aligned}$$

□

Uvedieme teraz Meggitov algoritmus dekódovania cyklických kódov. Kvôli jednoduchosti výkladu sa obmedzíme na binárne cyklické kódy; zovšeobecnenie Meggitovho algoritmu na q-árne cyklické kódy ponecháme na čitateľa. Predpokladáme, že

- $\mathcal{C}$ je cyklický (n,k)-kód nad poľom $GF(2)$ s generujúcim polynómom $g(x)$
- bol odvysielaný kódový polynóm $u(x)$,
- pri prenose vznikla (kódom $\mathcal{C}$ korigovateľná) chyba $e(x)$, v dôsledku čoho bol prijatý polynóm $v(x)$.

Meggittov algoritmus dekódovania binárnych cyklických kódov [1]

1. Vytvor dekódovaciu tabuľku $\mathcal{T}$ obsahujúcu všetky syndrómy, zodpovedajúce predstaviteľom tried rozkladu faktorového okruhu $GF(2)[x]/x^n - 1$, podľa cyklického kódu $\mathcal{C}$; chybovým polynómom stupňa $n - 1$. (To sú chybové polynómy, v ktorých je koeficient pri najvyššej mocnine x nenulový.)
2. Vydeľ prijatý polynóm $v(x)$ generujúcim polynómom $g(x)$ a vypočítaj syndrómový polynóm: $s(x) = v(x) \mod g(x)$.

3. Porovnaj syndróm $s(x)$ so syndrómami v dekódovacej tabuľke $\mathcal{T}$; ak sa $s(x)$ nachádza v dekódovacej tabuľke $\mathcal{T}$, oprav (aktuálne) najvyšší koeficient prijatého polynómu $v(x)$:
$$v(x) \leftarrow v(x) + x^{n-1}.$$

4. Vykonaj cyklický posun prijatého polynómu:
$$v(x) \leftarrow x \cdot v(x) \mod x^n - 1.$$
Ak sa krok 4 vykonal $n-1$-krát, vykonaj posledný cyklický posun polynómu $v(x)$ a skonči. (Dekódované slovo predstavujú koeficienty polynómu $v(x)$.) V opačnom prípade pokračuj krokom 2.

Poznámka 1. Meggittov algoritmus možno zefektívniť využitím výsledku vety 9.4.2 a namiesto cyklického posunu prijatého slova a výpočtu syndrómu na základe posunutého slova len „prepočítavame“ syndróm podľa vzťahu $s(x) \leftarrow x \cdot s(x) \mod g(x)$. Problém je v tom, že v tých krokoch, keď opravujeme chybu v prijatom slove, mení sa samotné prijaté slovo a treba vypočítať nový syndróm. Označme kvôli jednoduchosti polynóm $x^{n-1} \mod g(x)$ symbolom $\sigma(x)$, potom syndróm chyby prijatého slova po korekcii najvyššieho bitu bude
$$(v(x) + x^{n-1}) \mod g(x) = v(x) \mod g(x) + x^{n-1} \mod g(x) = s(x) + \sigma(x).$$

Meggittov algoritmus sa potom dá zapísať nasledovne:

1. Vytvor dekódovaciu tabuľku $\mathcal{T}$ obsahujúcu všetky syndrómy, zodpovedajúce predstaviteľom tried rozkladu faktorového okruhu $GF(2)[x]/x^n - 1$, podľa cyklického kódu $\mathcal{C}$; chybovým polynómom stupňa $n-1$. (Predstavitelia tried rozkladu sú chybové polynómy, v ktorých je koeficient pri najvyššej mocnine x nenulový.) $\sigma(x) \leftarrow x^{n-1} \mod g(x)$, $s(x) \leftarrow v(x) \mod g(x)$, `PocetPosunov` $\leftarrow 0$.

2. Ak $s(x) = 0$ pokračuj krokom 6, ináč pokračuj krokom 3.

3. Zisti, či sa $s(x)$ nachádza v tabuľke $\mathcal{T}$. Ak áno pokračuj krokom 4, ak nie, pokračuj krokom 5.

4. $v(x) \leftarrow (v(x) + x^{n-1})$, $s(x) \leftarrow s(x) + \sigma(x)$, pokračuj krokom 5.

5. $v(x) \leftarrow x \cdot v(x) \mod x^n$, $s(x) \leftarrow x \cdot s(x) \mod g(x)$, $++$ `PocetPosunov` pokračuj krokom 2.

6. $v(x) \leftarrow x^{n-}$`PocetPosunov`$\cdot v(x) \mod x^n - 1$ (cyklicky posuň prijaté slovo o $n-$`PocetPosunov` miest.

Poznámka 2. Meggittov algoritmus sa dá veľmi efektívne realizovať pomocou dekodéra založeného na *posuvných registroch s lineárnou spätnou väzbou* (linear feedback shift register, LFSR), pomocou ktorých sa vykonávajú cyklické posuny, modulárne násobenie a delenie polynómov. Nie je problém v prípade potreby modifikovať najvyšší bit, ktorý LFSR obsahuje, ale konštrukcia LFSR by sa skomplikovala, ak by bolo potrebné zabezpečiť, aby bolo možné v každom kroku meniť ľubovoľné bity LFSR. Preto sa v 3. kroku koriguje len aktuálne najvyšší bit slova, ktoré register obsahuje, hoci syndrómový polynóm jednoznačne určuje aj prípadné ďalšie chyby. Na druhej strane, keďže sa v 3. kroku opravuje len najvyšší bit spracovávaného slova, dekódovacia tabuľka nemusí obsahovať chybové polynómy.

Ilustrujeme teraz Meggittov algoritmus na príklade [1].

Príklad. Uvažujme binárny cyklický (15,7)-kód $\mathcal{C}$ s generujúcim polynómom $g(x) = x^8 + x^7 + x^6 + x^4 + 1$, opravujúci chyby váhy 2. Dekódovacia tabuľka obsahuje 15 syndrómových polynómov. Kvôli názornosti v nej uvádzame aj syndrómom prislúchajúce chybové polynómy, ktoré však v ďalšom pri dekódovaní nebudeme používať.

chybový polynóm	syndrómový polynóm	syndróm
x^{14}	$x^7 + x^6 + x^5 + x^3$	1110 1000
$x^{14} + 1$	$x^7 + x^6 + x^5 + x^3 + 1$	1110 1001
$x^{14} + x$	$x^7 + x^6 + x^5 + x^3 + x$	1110 1010
$x^{14} + x^2$	$x^7 + x^6 + x^5 + x^3 + x^2$	1110 1100
$x^{14} + x^3$	$x^7 + x^6 + x^5$	1110 0000
$x^{14} + x^4$	$x^7 + x^6 + x^5 + x^4 + x^3$	1111 1000
$x^{14} + x^5$	$x^7 + x^6 + x^3$	1100 1000
$x^{14} + x^6$	$x^7 + x^5 + x^3$	1010 1000
$x^{14} + x^7$	$x^6 + x^5 + x^3$	0110 1000
$x^{14} + x^8$	$x^5 + x^4 + x^3 + 1$	0011 1001
$x^{14} + x^9$	$x^7 + x^4 + x^3 + x + 1$	1001 1011
$x^{14} + x^{10}$	$x^3 + x^2 + x$	0000 1110
$x^{14} + x^{11}$	$x^7 + x^6 + x^5 + x^4 + x^2 + 1$	1111 0101
$x^{14} + x^{12}$	$x^7 + x^6 + x^4 + x$	1101 0010
$x^{14} + x^{13}$	$x^7 + x^4 + x^3 + x^2$	1001 1100

Tabuľka 9.2: Dekódovacia tabuľka binárneho (15,7)-kódu

Predpokladáme, že bolo odvysielané slovo $u(x)$, počas prenosu vznikla chyba $e(x)$ a bolo prijaté slovo $v(x)$:

$$\begin{array}{lll} u(x) & 011\ 1011\ 1111\ 0001 & x^{13} + x^{12} + x^{11} + x^9 + x^8 + x^7 + x^6 + x^5 + x^4 + 1 \\ e(x) & 000\ 0010\ 0001\ 0000 & x^9 + x^4 \\ v(x) & 011\ 1001\ 1110\ 0001 & x^{13} + x^{12} + x^{11} + x^8 + x^7 + x^6 + x^5 + 1 \end{array}$$

V nasledujúcej tabuľke sú uvedené hodnoty koeficientov spracovávaného polynómu $v(x)$ a syndrómu $s(x)$ v jednotlivých krokoch algoritmu. Kvôli stručnosti uvádzame len

vektory koeficientov príslušných polynómov, pričom koeficienty pri najvyšších mocninách sú vľavo.

krok	$v(x)$				$s(x)$		$s(x) \in$ DT?
0.	011	10$\underline{0}$1	111$\underline{0}$	0001	0110	0011	–
1.	111	0$\underline{0}$11	11$\underline{0}$0	0010	1100	0110	–
2.	110	$\underline{0}$111	1$\underline{0}$00	0101	0101	1101	–
3.	10$\underline{0}$	1111	$\underline{0}$000	1011	1011	1010	–
4.	0$\underline{0}$1	111$\underline{0}$	0001	0111	1010	0101	–
5.	$\underline{0}$11	11$\underline{0}$0	0010	1110	1001	1011	✓
6.	111	1$\underline{0}$00	0101	1101	1110	0110	–
7.	111	$\underline{0}$000	1011	1011	0001	1101	–
8.	11$\underline{0}$	0001	0111	0111	0011	1010	–
9.	1$\underline{0}$0	0010	1110	1111	0111	0100	–
10.	$\underline{0}$00	0101	1101	1111	1110	1000	✓
11.	000	1011	1011	1111	0000	0000	–
	011	1011	1111	0001	0000	0000	$*$

Tabuľka 9.3: Dekódovanie (15,7)-kódu pomocou Meggittovej metódy

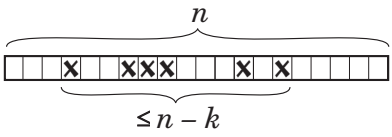
Poznámka. Vráťme sa ešte k predchádzajúcemu príkladu. Z porovnania zložitosti dekódovania uvedeného (15,7)-kódu pomocou štandardnej metódy dekódovania lineárnych kódov a Meggittovej metódy, jednoznačne vychádza lepšie Meggittova metóda. Štandardná tabuľka dekódovania obsahuje 256 dvojíc (syndróm, chyba), zatiaľ čo Meggittov dekóder pracuje s tabuľkou pozostávajúcou z 15 položiek (syndrómov). Meggittova metóda umožňuje opraviť všetky chyby váhy najviac 2; t.j. 121 rozličných chýb. Chyby váhy väčšej ako 2 Meggittov dekóder neopravuje. Ak sa na (15,7)-kódu pozeráme ako na lineárny kód, tak zostávajúcich 135 tried štandardného rozkladu vektorového priestoru $GF(2)^{15}$ podľa kódu má ako predstaviteľov vektory-chyby váhy väčšej ako 2. Keďže binárnych vektorov dĺžky 15 váhy 3 je 455, v ideálnom prípade (ak by predstaviteľmi ostávajúcich tried rozkladu boli chybové vektory váhy 3), by dekóder lineárneho kódu umožnil popri chybách váhy 0,1 a 2 opraviť asi 30% chýb váhy 3.

Cyklický posun prijatých slov kódovaných pomocou cyklických kódov možno využiť na ešte efektívnejšie dekódovanie, ako poskytuje Meggittov dekóder v prípade, ak sa chyby v prijatom slove nenachádzajú príliš ďaleko od seba. V takomto prípade je možné použiť metódu nazývanú *error trapping* , ktorú podrobnejšie popíšeme v nasledujúcej časti.

9.5 Error trapping dekódovanie

Error trapping dekódovanie („chytanie/lapanie“ chýb) je metóda dekódovania cyklických kódov. Vhodná je pri kódoch opravujúcich jednu, prípadne dve chyby a v situáciach, keď očakávame výskyt chýb na blízkych pozíciach v kódovom slove (tzv. burst chyby).

Budeme predpokladať, že $\mathcal{C}$ je cyklický (n, k) kód nad $GF(2)$ opravujúci t chýb. Teda kódové slová v $\mathcal{C}$ majú dĺžku n a k informačných symbolov. Nech $g(x)$ je generujúci polynóm kódu $\mathcal{C}$. Metóda *error trapping* funguje správne vtedy, ak je najviac t chýb rozmiestnených na najviac $n - k$ susedných pozíciach.



Nech $f(x) = a_0+a_1x+a_2x^2+\cdots+a_nx^n$ je polynóm nad $GF(2)$. Označme $\mathbf{wt}(f(x)) = \sum_i a_i$ jeho váhu. Nech w je slovo s najviac t chybami. Potom platia nasledujúce tvrdenia.

Lema 1. *Ak je váha syndrómu slova w najviac t, tak chybový polynóm je rovný syndrómu.*

Dôkaz. Označme $s(x)$ syndróm a $e(x)$ príslušný chybový polynóm slova w. Potom platí:

$$s(x) = e(x) \bmod g(x).$$

Inak povedané, $e(x) = q(x)g(x) + s(x)$. Z predpokladu o počte chýb vo w vyplýva, že $\mathbf{wt}(e(x)) \leq t$. Navyše vieme, že $\mathbf{wt}(s(x)) \leq t$ (predpoklad lemy). Teda váha $e(x) - s(x)$ je najviac $2t$. Kód $\mathcal{C}$ opravuje t chýb, teda minimálna vzdialenosť ľubovoľných dvoch kódových slov je aspoň $2t+1$. Keďže $e(x)-s(x)$ je kódové slovo (je to násobok generujúceho polynómu) s váhou najviac $2t$ a 0 je tiež kódové slovo, dostávame:

$$e(x) - s(x) = 0.$$

Odtiaľ bezprostredne vyplýva tvrdenie lemy. □

Lema 2. *Ak sú chyby na najviac $n - k$ susedných pozíciach, tak existuje cyklický posun w, ktorý má syndróm váhy najviac t.*

Dôkaz. Vezmime taký cyklický posun w, pre ktorý sú chyby „najviac vpravo“ (formálne, stupeň chybového polynómu pre príslušný cyklický posun je minimálny). Označme takýto posun w' a prislúchajúci chybový polynóm $e'(x)$. Podľa predpokladu lemy je $\deg(e'(x)) \leq n - k - 1$, kde symbolom deg označujeme stupeň polynómu. Pre syndróm $s'(x)$ máme:

$$s'(x) = e'(x) \bmod g'(x).$$

Keďže $g(x)$ je generujúcim polynómom kódu $\mathcal{C}$, $\deg(g(x)) = n - k$. Preto $s'(x) = e'(x)$. Po zohľadnení predpokladu o počte chýb vo w dostávame $\mathbf{wt}(s'(x))) \leq t$. □

Predchádzajúce dve lemy poskytujú teoretické zdôvodnenie pre error trapping dekódovanie cyklických kódov (samozrejme, pri splnení predpokladov o umiestnení chýb v dekódovanom slove w). Postupne skušame pre všetky rotácie w' slova w:

1. $s'(x) \leftarrow w'(x) \bmod g(x)$
2. ak $\mathbf{wt}(s'(x)) \leq t$ (podľa lemy 2):
   (a) oprav w': $w'(x) \leftarrow w'(x) + s'(x)$ (lema 1)
   (b) $w \leftarrow$ aplikuj „inverznú“ rotáciu na w'
   (c) koniec

Pod pojmom inverzná rotácia máme na mysli to, že ak sme w' dostali z w cyklickým posunom o p pozícií doľava, tak teraz opravené w' posunieme cyklicky o p pozícií doprava. Na konci dostaneme vo w opravené slovo.

Príklad. Uvažujme binárny cyklický kód $(15, 7)$ opravujúci 2 chyby, s generujúcim polynómom

$$g(x) = x^8 + x^7 + x^6 + x^4 + 1.$$

V tomto prípade bude error trapping metóda dekódovania úspešná pre tie dvojice chýb, ktoré sú od seba vzdialené (cyklicky) o najviac $15 - 7 = 8$ pozícií. A to sú všetky

Samozrejme, slová s jednou (alebo dokonca žiadnou) chybou dokáže error trapping dekódovať vždy.

Príklad. Uvažujme binárny cyklický kód $(15, 5)$ opravujúci 3 chyby, s generujúcim polynómom

$$g(x) = x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1.$$

Vzdialenosť chýb musí byť v tomto prípade najviac $15 - 5 = 10$. Pre chyby váhy menšej ako 3 je podmienka triviálne splnená. Demonštrujme si príklad dokódovania na slove $w = 100000100010110$. Poznamenajme, že pri výpočtoch budeme polynómy reprezentovať ako vektory ich koeficientov (pre prehľadnejší zápis). Postupne rotujeme w doľava, počítame syndróm a hľadáme taký, ktorý má váhu najviac 3:

```
posun: 0              posun: 1
100000100010110       000001000101101
10100110111                1000101101
  1001001100110
  10100110111
    11010111010
    10100110111
     1110001101

posun: 2              posun: 3
000010001011010       000100010110100
    10100110111          10100110111
      101101101            1011011010
```

```
posun: 4
001000101101000
  10100110111
    10110110100
    10100110111
       10000011
```

Váha syndrómu pre posun 4 je rovná 3. Takže môžeme opraviť slovo $w' = 001000101101000 \oplus 10000011 = 001000111101011$ a posunúť ho naspäť (o 4 pozície doprava). Teda opravené slovo w je:

$$101100100011110.$$

Urobme ešte skúšku správnosti, keď skúsime vydeliť toto slovo generujúcim polynómom (pre kódové slovo očakávame zvyšok 0):

```
101100100011110
10100110111
   101001101110
   10100110111
            0
```

V prípade binárneho cyklického $(15, 5)$ kódu môže nastať práve päť rozmiestnení 3 chýb, keď tieto neležia v úseku dĺžky 10. Jedno rozmiestnenie je nakreslené na nasledujúcom obrázku, ostatné sú jeho cyklickými posunmi.

Pravdepodobnosť, že takáto situácia nastane pri náhodnej voľbe práve 3 chýb je $5/\binom{15}{3} \sim 0,011$.

9.6 Golayov kód

V časti 7.2 sme zaviedli pojem dokonalého kódu a následne sme ukázali, že Hammingove kódy sú dokonalé binárne kódy. Okrem triviálnych kódov nepárnej dĺžky pozostávajúcich z dvoch kódových slov s maximálnou možnou vzdialenosťou slov sa nám iné dokonalé kódy nepodarilo nájsť. To naznačuje, že dokonalých kódov nemusí byť veľa, resp. že iné dokonalé kódy ani nemusia existovať. V tejto časti najprv popíšeme netriviálny binárny cyklický dokonalý kód opravujúci 3 chyby a potom v nasledujúcej časti uvedieme výsledky o (ne-)existencii dokonalých kódov.

Nutnou podmienkou existencie dokonalého q-árneho kódu opravujúceho t chýb je splnenie rovnosti

$$q^{n-k} = \sum_{j=0}^{t} \binom{n}{j} (q-1)^j,$$

t.j. mohutnosť „kódovej gule“ musí byť mocninou q. Táto podmienka je splnená pre $n = 23$, $q = 2$ a $t = 3$, nakoľko

$$\binom{23}{0} + \binom{23}{1} + \binom{23}{2} + \binom{23}{3} = 2^{11}.$$

Ukážeme, že kód s uvedenými parametrami skutočne existuje. Ide o Golayov binárny cyklický (23,12)-kód opravujúci tri chyby.

Najprv nájdeme generujúci polynóm daného kódu. (Pripomíname, že by to mal byť polynóm stupňa 11 nad poľom $GF(2)$, ktorý delí polynóm $x^{23} - 1$). Rozložíme polynóm $x^{23} - 1$ na vhodné činitele:

$$x^{23} - 1 = (x - 1) \cdot g(x) \cdot \tilde{g}(x),$$

kde

$$g(x) = 1 + x^2 + x^4 + x^5 + x^6 + x^{10} + x^{11} \quad (9.7)$$

$$\tilde{g}(x) = 1 + x + x^5 + x^6 + x^7 + x^8 + x^{11} \quad (9.8)$$

Vyberieme ako generujúci polynóm cyklického kódu $\mathcal{C}$ a ukážeme, že že tento kód skutočne opravuje 3 chyby. Kedže generujúci polynóm $g(x)$ má 7 nenulových koeficientov, mainimálna vzdialenosť kódu $\mathcal{C}$ nemôže presiahnuť hodnotu 7. Ukážeme, že minimálna vzdialenosť kódu $\mathcal{C}$ je skutočne 7. Kód $\mathcal{C}$ je lineárny, a preto by stačilo zostrojiť jeho kontrolnú maticu a ukázať, že jej ľubovoľných 6 stĺpcov je lineárne nezávislých. To by však znamenalo preveriť $\binom{23}{6} = 100947$ možností čo by sa pomocou počítača dalo spraviť. Uvedieme iný, rafinovanejší kombinatorický dôkaz [2]), ktorého podstata spočíva v tom, že vylúčime existenciu kódových slov váhy menšej ako 7 v kóde $\mathcal{C}$.[4] Dôkaz rozdelíme na tri časti, v ktorých postupne ukážeme, že kód $\mathcal{C}$ neobsahuje slová

1. váhy menšej alebo rovnej 4;
2. váhy 2,6,10,14,18,22;
3. váhy 1,5,9,13,17,21.

Najprv ukážene, že $g(x)$ a $\tilde{g}(x)$ sú ireducibilné polynómy nad poľom $GF(2)$, potom sa pozrieme na korene týchto polynómov v rozšírenom poli $GF(2^{11})$. Nech je α generátor cyklickej grupy poľa $GF(2^{11})$. To znamená, že rád prvku α je $2^{11} - 1 = 2047$. Číslo 2047 sa dá rozložiť na súčin prvočísel $23 \cdot 89$. Potom prvok $\beta = \alpha^{89}$ aj inverzný prvok k prvku β, prvok $\beta^{-1} = \alpha^{1958}$ poľa $GF(2^{11})$ majú rád 23. Označme minimálne polynómy prvkov β a β^{-1} symbolmi $m_\beta(x)$, resp. $m_{\beta^{-1}}(x)$. Koreňmi minimálneho polynómu $m_\beta(x)$ sú prvky

$$\beta^2, \beta^4, \beta^8, \beta^{16}, \beta^{32)} = \beta^9, \beta^{64} = \beta^{18}, \beta^{128} = \beta^{13}, \beta^{256} = \beta^3, \beta^{512} = \beta^6, \beta^{1024} = \beta^{12}, \beta^{2048} = \beta,$$

a koreňmi minimálneho polynómu $m_{\beta^{-1}}(x)$ sú všetky prvky poľa $GF(2^{11})$ tvaru $(\beta^{-1})^{2^j}$ $j = 1, 2, \ldots$; t.j. prvky

$$\beta^{22}, \beta^{19}, \beta^{15}, \beta^7, \beta^{14}, \beta^5, \beta^{10}, \beta^{20}, \beta^{17}, \beta^{11}.$$

[4] Idea dôkazu sa nikde inde nevyužíva, a preto dôkaz možno pri prvom čítaní preskočiť.

Oba minimálne polynómy $m_\beta(x)$, $m_{\beta^{-1}}(x)$ majú po 11 koreňov; t.j. majú stupeň 11 a dajú sa vyjadriť ako súčin lineaárnych činiteľov:

$$\begin{aligned} m_\beta(x) &= (x-\beta)\cdot(x-\beta^2)\cdot\cdots\cdot(x-\beta^{12}) \\ m_{\beta^{-1}}(x) &= (x-\beta^{22})\cdot(x-\beta^{19})\cdot\cdots\cdot(x-\beta^{11}) \end{aligned}$$

Keďže multiplikatívna podgrupa poľa $GF(2^{11})$ generovaná prvkom β má prvočíselný rád (23), každý z prvkov $\beta, \beta^2, \ldots, \beta^{22}$ má rád 23, a teda polynóm $x-\beta^j$ delí polynóm $(x-\beta^{23})$, $j = 1, \ldots, 22$ a, samozrejme, aj polynóm $x-1$ delí polynóm $(x-\beta^{23})$. To znamená, že polynóm $(x-\beta^{23})$ môžeme vyjadriť v tvare súčinu

$$(x-\beta^{23}) = m_\beta(x)\cdot m_{\beta^{-1}}(x)(x-1).$$

Rozklad polynómu na súčin ireducibilných polynómov (nad daným poľom) je jednoznačný, a to znamená, že

$$m_\beta(x)\cdot m_{\beta^{-1}}(x) = g(x)\cdot\tilde{g}(x),$$

resp. (napríklad)

$$m_\beta(x) = g(x) \qquad \tilde{g}(x) = m_{\beta^{-1}}(x),$$

t.j. generujúce polynómy $g(x)$ a $\tilde{g}(x)$ sú zároveň minimálnymi polynómami prvkov β a β^{-1} poľa $GF(2^{11})$. Teraz vylúčime existenciu nenulových kódových slov váhy menšej alebo rovnej 4 v kóde $\mathcal{C}$.

Lema 3. *Cyklický kód $\mathcal{C}$ s generujúcim polynómom $g(x)$ neobsahuje nenulové kódové slová váhy menšej alebo rovnej 4.*

Dôkaz. Keďže prvky $\beta, \beta^2, \beta^3, \beta^4$ sú koreňmi generujúceho polynómu $g(x)$ a každý kódový polynóm je násobkom generujúceho polynómu, musia byť uvedené prvky koreňmi každého kódovéhom polynómu kódu $\mathcal{C}$. V maticovom vyjadrení to vyzerá nasledovne: ak je $u(x)$ kódový polynóm kódu $\mathcal{C}$, tak $\mathbf{u}\cdot H^\top = \mathbf{0}$, kde $\mathbf{u}$ je vektor koeficientov kódového polynómu $u(x)$ a H je matica tvaru

$$H = \begin{bmatrix} 1 & \beta & \beta^2 & \ldots & \beta^{22} \\ 1 & \beta^2 & \beta^4 & \ldots & \beta^{21} \\ 1 & \beta^3 & \beta^6 & \ldots & \beta^{20} \\ 1 & \beta^4 & \beta^8 & \ldots & \beta^{19} \end{bmatrix}$$

Matica H je zrejme kontrolnou maticou kódu $\mathcal{C}$. Ak vyberieme ľubovoľné 4 stĺpce matice H, dostávame štvorcovú podmaticu, ktorej determinant sa dá vyjadriť v tvare

$$\begin{vmatrix} \beta^{i_1} & \beta^{i_2} & \beta^{i_3} & \beta^{i_4} \\ \beta^{2i_1} & \beta^{2i_2} & \beta^{2i_3} & \beta^{2i_4} \\ \beta^{3i_1} & \beta^{3i_2} & \beta^{3i_3} & \beta^{3i_4} \\ \beta^{4i_1} & \beta^{4i_2} & \beta^{4i_3} & \beta^{4i_4} \end{vmatrix} = \beta^{4(i_1+i_2+i_3+i_4)}\cdot \begin{vmatrix} 1 & 1 & 1 & 1 \\ \beta^{i_1} & \beta^{i_2} & \beta^{i_3} & \beta^{i_4} \\ \beta^{2i_1} & \beta^{2i_2} & \beta^{2i_3} & \beta^{2i_4} \\ \beta^{3i_1} & \beta^{3i_2} & \beta^{3i_3} & \beta^{3i_4} \end{vmatrix}.$$

Determinant na pravej strane je Vandermondov determinant, ktorého hodnota

$$\prod_{j>l}(\beta^{i_j} - \beta^{i_l})$$

je nenulová, nakoľko $\beta^{i_j} \neq \beta^{i_l}$ ak $j \neq l$. To znamená, že ľubovoľná štvorcová podmatica 4×4 kontrolnej matice H je regulárna, resp. minimálna vzdialenosť kódu $\mathcal{C}$ je aspoň 5. □

Teraz vylúčime existenciu kódových slov váhy 2,6,10,14,18,22.

Lema 4. *Nech $\mathcal{C}$ je cyklický (23,12) kód s generujúcim polynómom $g(x)$ a nech je $u(x)$ kódový polynóm párnej váhy. Potom je váha kódového polynómu $u(x)$ deliteľná 4.*

Dôkaz. Nech je $u(x) = u_0 + u_1x + \ldots u_{22}x^{22}$ kódový polynóm kódu $\mathcal{C}$. V dôkaze budeme využívať obe interpretácie kódových slov - aj polynomickú, aj vektorovú. Označíme preto symbolom $\mathbf{u} = (u_0, u_1, \ldots, u_{22})$ kódové slovo–vektor koeficientov–kódového polynómu $u(x)$. Keďže $\mathcal{C}$ je cyklický kód, kódový polynóm $u(x)$ sa dá vyjadriť v podobe súčinu

$$u(x) = g(x) \cdot a(x),$$

kde $a(x)$je polynóm nad $GF(2)$. Ak je váha kódového slova $\mathbf{u}$ párna, tak je hodnota $\sum_i u_i$ párna, resp.

$$\sum_i u_i = 0 \mod 2. \tag{9.9}$$

Dosadíme do kódového polynómu $u(x)$ hodnotu 1 a na základe 9.9 dostávame

$$u(1) = u_0 + u_1 + \cdots + u_{22} = 0 \mod 2.$$

To znamená, že 1 je koreňom kódového polynómu $u(x)$, resp., že

$$u(1) = g(1) \cdot a(1) = 0 \tag{9.10}$$

Keďže prvok 1 nie je koreňom generujúceho polynómu $g(x)$, zo vzťahu 9.10 vyplýva, že prvok 1 je koreňom polynómu $a(x)$, a kódový polynóm možno rozložiť na súčin nasledujúcich činiteľov:

$$u(x) = g(x) \cdot b(x) \cdot (x - 1). \tag{9.11}$$

Uvažujeme teraz zrkadlový obraz kódového slova $\mathbf{u}$, slovo $\mathbf{u}^R = u_{22} \ldots u_1 u_0$. Slovu $\mathbf{u}^R$ priradíme polynóm

$$\tilde{u}(x) = \tilde{u}_0 + \tilde{u}_1x + \cdots + \tilde{u}_{21}x^{21} + \tilde{u}_{22}x^{22} = \tag{9.12}$$

$$= u_{22} + u_{21}x + \cdots + u_1x^{21} + u_0x^{22}. \tag{9.13}$$

Dá sa ľahko overiť, že pre koeficienty polynómov $\tilde{u}(x)$ a $u(x)$ platí

$$\tilde{u}_i = u_{22-i}, \quad i = 0, \ldots, 22; \tag{9.14}$$

a samotné polynómy sú spojené vzťahom

$$\tilde{u}(x) = x^{22} \cdot u(x^{-1}). \tag{9.15}$$

V ďalšom budeme kombinovať polynómy $\tilde{u}(x)$ a $u(x)$. Prvok β^{-1} $(= \beta^{22})$ je koreňom polynómu $\tilde{u}(x)$, pretože podľa 9.15

$$\tilde{u}(\beta^{-1}) = \beta^{-22} \cdot u(\beta) = 0.$$

Minimálnym polynómom prvku β^{-1} je polynóm $\tilde{g}(x)$. Ak β^{-1} je koreňom polynómu $\tilde{u}(x)$, tak $\tilde{g}(x)$ delí $\tilde{u}(x)$; t.j. existuje polynóm $\tilde{a}(x)$ taký, že

$$\tilde{u}(x) = \tilde{g}(x) \cdot \tilde{a}(x). \tag{9.16}$$

Teraz vynásobíme polynómy $\tilde{u}(x)$ a $u(x)$:

$$\begin{aligned} \tilde{u}(x) \cdot u(x) &= \tilde{g}(x) \cdot \tilde{a}(x) \cdot g(x) \cdot b(x) \cdot (x-1) = \tilde{g}(x) \cdot g(x) \cdot (x-1) \cdot b(x) \cdot \tilde{a}(x) = \\ &= (x^{23}-1) \cdot b(x) \cdot \tilde{a}(x). \end{aligned} \tag{9.17}$$

Vyjaríme súčin polynómov $\tilde{u}(x)$ a $u(x)$ explicitne, aby sme mohli kombinovať ich koeficienty

$$\begin{aligned} \tilde{u}(x) \cdot u(x) &= \tilde{u}_0 \cdot u_0 + (\tilde{u}_0 \cdot u_1 + \tilde{u}_1 \cdot u_0) \cdot x + (\tilde{u}_0 \cdot u_2 + \tilde{u}_1 \cdot u_1 + \tilde{u}_2 \cdot u_0) \cdot x^2 + \\ &+ (\tilde{u}_0 \cdot u_3 + \tilde{u}_1 \cdot u_2 + \tilde{u}_2 \cdot u_1 + \tilde{u}_3 \cdot u_0) \cdot x^3 + \cdots + \tilde{u}_{22} \cdot u_{22} \cdot x^{44}. \end{aligned}$$

Využijeme vzťah 9.14, v súčine polynómov $\tilde{u}(x)$ a $u(x)$ nahradíme koeficienty $\tilde{u}_i$ „obyčajnými“ koeficientami a vyjadríme ho v tvare konvolúcie postupností koefiecientov.

$$\tilde{u}(x) \cdot u(x) = \sum_{j=0}^{44} \sum_{i=0}^{22} u_i \cdot u_{22+i-j} \cdot x^j.$$

Kvôli zjednodušeniu úprav položíme $u_i = 0$ pre $(i < 0) \vee (i > 22)$. Rozšírime teraz sumačný rozsah premennej i v poslednej sume

$$\tilde{u}(x) \cdot u(x) = \sum_{j=0}^{44} \sum_{i=-\infty}^{\infty} u_i \cdot u_{22+i-j} \cdot x^j \tag{9.18}$$

a rozdelíme sumačný rozsah premennej j na tri disjunktné intervaly: $(0 \le j \le 44) = (0 \le j < 22) \vee (j = 22) \vee (23 \le j \le 44)$. Potom sa súčin polynómov $\tilde{u}(x)$ a $u(x)$ rozpadáva na tri sumy:

$$\tilde{u}(x) \cdot u(x) = \sum_{j=0}^{21} \sum_{i=-\infty}^{\infty} u_i \cdot u_{22+i-j} \cdot x^j + \sum_{i=-\infty}^{\infty} u_i^2 \cdot x^{22} + \sum_{j=23}^{44} \sum_{i=-\infty}^{\infty} u_i \cdot u_{22+i-j} \cdot x^j \tag{9.19}$$

Koeficienty u_i sú prvky poľa $GF(2)$, a preto $u_i^2 = u_i$. Kódové slovo $\mathbf{u}$ má párnu váhu a teda druhá suma v súčine 9.19 je nulová[5]

$$\sum_{i=-\infty}^{\infty} u_i^2 \cdot x^{22} = x^{22} \cdot \sum_{i=-\infty}^{\infty} u_i = 0.$$

Teraz upravíme 1. a 3. sumu v súčine 9.19. Posunieme hranice sumácie v prvej z uvedených súm substitúciou $j := j - 1$, resp. $j := j + 22$ v druhej sume:

$$\tilde{u}(x) \cdot u(x) = \sum_{j=1}^{22} \sum_{i=-\infty}^{\infty} u_i \cdot u_{23+i-j} \cdot x^{j-1} + \sum_{j=1}^{22} \sum_{i=-\infty}^{\infty} u_i \cdot u_{i-j} \cdot x^{j+22} \tag{9.20}$$

[5] v modulárnej aritmetike nad poľom $GF(2)$.

Teraz zavedieme substitúciu $i := i + j$ v druhej sume 9.20

$$\tilde{u}(x) \cdot u(x) = \sum_{j=1}^{22} \sum_{i=-\infty}^{\infty} u_i \cdot u_{23+i-j} \cdot x^{j-1} + \sum_{j=1}^{22} \sum_{i=-\infty}^{\infty} u_i \cdot u_{i+j} \cdot x^{j+22} \tag{9.21}$$

a upravíme

$$\begin{aligned} \tilde{u}(x) \cdot u(x) &= \sum_{j=1}^{22} \sum_{i=-\infty}^{\infty} u_i \cdot u_{23+i-j} \cdot x^{j-1} + \sum_{j=1}^{22} \sum_{i=-\infty}^{\infty} u_i \cdot u_{i+j} \cdot x^{j-1} - \\ &- \sum_{j=1}^{22} \sum_{i=-\infty}^{\infty} u_i \cdot u_{i+j} \cdot x^{j-1} + \sum_{j=1}^{22} \sum_{i=-\infty}^{\infty} u_i \cdot u_{i+j} \cdot x^{j+22} = \end{aligned}$$

$$= \sum_{j=1}^{22} \left[\sum_{i=-\infty}^{\infty} u_i \cdot u_{23+i-j} + u_i \cdot u_{i+j} \right] \cdot x^{j-1} + \left[\sum_{j=1}^{22} \sum_{i=-\infty}^{\infty} u_i \cdot u_{i+j} \cdot x^{j-1} \right] \cdot (x^{23} - 1) \tag{9.22}$$

Súčin polynómov $\tilde{u}(x) \cdot u(x)$ je podľa vzťahu 9.17 deliteľný polynómom $(x^{23} - 1)$ Z rovnosti 9.22 vyplýva, že aj polynóm

$$\sum_{j=1}^{22} \left[\sum_{i=-\infty}^{\infty} u_i \cdot u_{23+i-j} + u_i \cdot u_{i+j} \right] \cdot x^{j-1} \tag{9.23}$$

musí byť násobkom polynómu $(x^{23} - 1)$. Polynóm 9.23 má však stupeň menší ako 23, a ak má byť násobkom polynómu $(x^{23} - 1)$, musí byť nulový. To znamená, že

$$\sum_{i=-\infty}^{\infty} u_i \cdot u_{23+i-j} + u_i \cdot u_{i+j} = 0 \mod 2, \quad j = 1, \ldots, 22. \tag{9.24}$$

Ak v sústave 9.24 prejdeme od modulárnej k celočíselnej aritmetike, sústavu rovníc 9.24 možno zapísať nasledovne

$$\sum_{i=-\infty}^{\infty} u_i \cdot u_{23+i-j} + u_i \cdot u_{i+j} = 2d_j; \quad d_j \in Z,\ j = 1, \ldots, 22. \tag{9.25}$$

Ak vo vzťahu 9.25 zavedieme substitúciu $j := 23 - j$, dostávame

$$2d_{j-23} = \sum_{i=-\infty}^{\infty} u_i \cdot u_{i+j} + u_i \cdot u_{23+i-j} = 2d_j; \quad d_j \in Z,\ j = 1, \ldots, 22. \tag{9.26}$$

To znamená, že $d_j = d_{j-23}$ pre $j = 1, \ldots, 22$. Spočítame teraz hodnotu súčtu súm z 9.25 v celočíselnom obore

$$\sum_{j=1}^{22} \sum_{i=-\infty}^{\infty} u_i \cdot (u_{23+i-j} + u_{i+j}) = 2 \sum_{j=1}^{22} d_j = 4 \sum_{j=1}^{11} d_j = 4d, \tag{9.27}$$

kde d je nejaké celé číslo. Rozdelíme sumu 9.27 na dve sumy. Vzhľadom na to, že každý zo sumandov obsahuje faktor u_0, môžeme zúžiť sumačný rozsah premennej i na interval $0 \leq i \leq 22$. Potom v prvej sume použijeme substitúciu $j := 23 - j$

$$\sum_{j=1}^{22} \sum_{i=-\infty}^{\infty} u_i \cdot (u_{23+i-j} + u_{i+j}) = \sum_{j=1}^{22} \sum_{i=0}^{22} u_i \cdot u_{i+j} + \sum_{j=1}^{22} \sum_{i=0}^{22} u_i \cdot u_{i+j}. \tag{9.28}$$

Všimneme si, že sumy v pravej časti rovnosti (9.27) obsahujú všetky dvojice $u_i \cdot u_j$ také, že $i \neq j$; t.j.

$$\sum_{j=1}^{22} \sum_{i=-\infty}^{\infty} u_i \cdot (u_{23+i-j} + u_{i+j}) = \sum_{i=0}^{22} \sum_{j \neq i} u_i \cdot u_j = 4d. \tag{9.29}$$

Nech je $\mathbf{wt}([]u] = w$. Sumu (9.29) rozložíme na dve sumy podľa toho, akú hodnotu nadobúda u_i: a upravíme:

$$\begin{aligned} \sum_{i=0}^{22} \sum_{j \neq i} u_i \cdot u_j &= \sum_{i} \sum_{j \neq i} u_i \cdot u_j \cdot (u_i = 0) + \sum_{i} \sum_{j \neq i} u_i \cdot u_j \cdot (u_i = 1) = \\ &= 0 + \sum_{i} (u_i = 1) \sum_{j \neq i} u_j = \sum_{i} (u_i = 1) \cdot (w - 1) = w \cdot (w - 1). \end{aligned}$$

Z posledného vzťahu a z (9.29) dostávame, že

$$w \cdot (w - 1) = 4d.$$

Keďže Hammingova váha slova $\mathbf{u}$ je párna, hodnota $w-1$ je nepárna, a teda w musí byť deliteľné 4. □

Golayov kód teda neobsahuje slová párnej váhy, ktorá nie je deliteľná 4. To znamená, že spomedzi kandidátov na kódové slová vypadávajú slová váh $2, 6, 10, 14, 18, 22$. Potenciálne váhy kódových slov sú teda $5, 7, 8, 9, 11, 12, 13, 15, 16, 19, 20, 21, 23$. Problematické sú slová váhy 5, pretože ak by Golayov kód obsahoval také kódové slová, znamenalo by to, že jeho minimálna vzdialenosť je 5. Existenciu kódových slov váhy 5 (a ďalších) vylúčime v nasledujúcej leme.

Lema 5. *Golayov kód neobsahuje kódové slová váh* $1, 5, 9, 13, 17, 19, 20, 21$.

Dôkaz. Pripomenieme, že polynóm $x^{23} - 1$ sa dá vyjadriť v podobe súčinu

$$x^{23} - 1 = (x - 1) \cdot g(x) \cdot \tilde{g}(x).$$

To znamená, že polynóm

$$f(x) = x^{23} - 1/(x - 1) = 1 + x + x^2 + x^3 + \cdots * x^{22}$$

je kódovým slovom Golayovho kódu, nakoľko

$$x^{23} - 1/(x - 1) = g(x) \cdot \tilde{g}(x).$$

Golayov kód je zároveň lineárnym kódom, a to znamená, že súčet dvoch kódových slov (kódových polynómov)je opäť kódovým slovom (polynómom). To znamená, že pre ľubovoľný kódový polynóm $u(x)$ je aj polynóm $u(x) + f(x)$ kódovým polynómom. Kódový polynóm $f(x)$ má Hammingovu váhu 23; Hammingova váha kódového polynómu $u(x)+f(x)$ je $23 - \mathbf{wt}(u)$. Ak teda Golayov kód neobsahuje slová váhy w, nemôže obsahovať ani slová váhy $23 - w$. Z tohto faktu a z liem 3 a 4 vyplýva tvrdenie našej lemy. □Teraz môžeme sformulovať základné tvrdenie o Golayovom kóde.

Veta 9.6.1. *Binárny Golayov* $(23, 12)$ *kód je dokonalý kód opravujúci 3 chyby.*

Dôkaz. Vyplýva priamo z tvrdení liem 3, 4 a 5. □

Blahut [2] pomocou počítača analyzoval váhy slov Golayovho kódu. Výsledky jeho skúmania sú uvedené v nasledujúcej tabuľke

váha	**počet slov**
0	1
7	253
8	506
11	1288
12	1288
15	506
16	253
23	1
spolu	4096

Tabuľka 9.4: Váhy slov Golayovho (23,12)-kódu

9.7 Dokonalé a kvázidokonalé kódy

Kapitola 10

Boseove-Chandhuryove-Hocquenghemove kódy

Boseove-Chandhuryove-Hocquenghemove (BCH) kódy[1] tvoria pomerne rozsiahlu podtriedu cyklických kódov, ktoré sa vďaka svojim dobrým vlastnostiam často používajú v praxi a sú predmetom intenzívneho teoretického štúdia. Záujem o BCH kódy vyplýva podľa [2] najmä z toho, že

1. BCH kódy existujú pre pomerne rozsiahlu množinu parametrov (dĺžka, minimálna vzdialenosť a i.); a už pre malé dĺžky (kódu) existujú dobré BCH kódy,
2. pre BCH kódy existujú jednoduché metódy kódovania a dekódovania,
3. podtriedou BCH kódov sú známe Reedove-Solomonove kódy, ktoré sú pre niektoré parametre optimálne samoopravné kódy a
4. štúdium BCH kódov je dobrým základom pre pochopenie zložitejších kódov.

Výklad BCH kódov začneme konštrukciou binárnych BCH kódov opravujúcich 2 chyby. Potom formálne zavedieme BCH kódy a dokážeme vetu o minimálnej vzdialenosti BCH kódu. V ďalšom sa budeme zaoberať rozličnými metódami dekódovania BCH kódov a kapitolu zavŕšime časťou venovanou spomínaným Reedovým-Solomonovým kódom.

10.1 Binárne BCH kódy opravujúce 2 chyby

BCH kódy sú zovšeobecnením Hammingových kódov. Kým však Hammingove kódy majú dĺžky $q^m - 1$ a opravujú jednu chybu[2], BCH kódy sú podstatne rozmanitejšie: existujú

[1] BCH kódy objavil v roku 1959 A. Hocquenghem a nezávisle od neho v roku 1960 R.C.Bose a D.K.Ray-Chaunhuri.

[2] my sme sa zaoberali binárnymi Hammingovými kódmi; t.j. prípadom $q = 2$

BCH kódy rôznych dĺžok nad poľom $GF(q)$, $q \neq 2$, ktoré opravujú t chýb, $t > 1$. Skôr, ako sa budeme konštrukciou, vlastnosťami a dekódovaním BCH kódov zaoberať vo všeobecnosti, pozrieme sa na zvláštny prípad BCH kódov—binárne BCH kódy opravujúce 2 chyby a ilustrujeme na nich konštrukcie, ktoré budeme v ďalších častiach tejto kapitoly používať.

Vráťme sa k Hammingovmu $(15, 11)$ kódu zo začiatku kapitoly 9. Tento kód sme zadali pomocou kontrolnej matice H', ktorá sa dala interpretovať dvojako: buď ako matica typu $(4, 15)$ nad poľom $GF(2)$, alebo ako matica typu $(1, 15)$ nad poľom $GF(2^4)$. V druhom prípade sa kontrolná matica redukovala na 1 riadok, obsahujúci postupnosť mocnín primitívneho prvku α poľa $GF(2^4)$:

$$H' = [1\ \alpha\ \alpha^2\ \ldots\ \alpha^{14}]$$

a dekódovanie prijatého slova $\mathbf{v} = (v_0, v_1, \ldots, v_{14}) = \mathbf{u} + \mathbf{e}$; t.j. výpočet

$$\mathbf{v} \cdot H'^{\top} = \mathbf{u} \cdot H'^{\top} + \mathbf{e} \cdot H'^{\top},$$

sa dalo interpretovať ako vyčíslenie hodnoty polynómu $v(x) = v_0 + v_1 \cdot x + v_2 \cdot x^2 + \cdots + v_{14} \cdot x^{14}$ v prvku α; t.j. výpočet hodnoty $v(\alpha)$. Ak počas prenosu kódového slova nevznikla chyba (resp. vznikla chyba, ktorá odvysielané kódové slovo $\mathbf{u}$ transformovala na iné kódové slovo $\mathbf{v}$) tak $v(\alpha) = 0$. Predpokladajme, že počas prenosu vznikla chyba váhy 1; t.j. chybový polynóm bude mať tvar $e(x) = x^i$. Potom po dosadení prvku α do prijatého polynómu $v(x)$ dostávame hodnotu syndrómu

$$v(\alpha) = u(\alpha) + e(\alpha) = e(\alpha) = \alpha^i.$$

Poznámka. V ďalšom budeme kvôli jednoduchosti hodnoty jednotlivých zložiek syndrómu označovať symbolmi S_i; $v(\alpha^i) = S_i$, $i = 1, 2, \ldots$. Pripomíname, že hodnoty S_i nie sú koeficientami syndrómového polynómu $s(x)$, ale tak syndróm $S_1, \ldots$ ako aj syndrómový polynóm $s(x)$ nesú tú istú informáciu.

V prípade Hammingovho kódu informácia, ktorú obsahuje syndróm S_1 postačuje na určenie chyby. Stačí určiť hodnotu exponentu ($S_1 = \alpha^i$) čo sa dá spraviť napríklad pomocou tabuľky a potom negovať i-ty bit prijatého slova. Prejdeme teraz od konkrétneho Hammingovmu $(15, 11)$ kódu k všeobecnému Hammingovmu kódu dĺžky $n = 2^m - 1$ s kontrolnou maticou

$$H = [1\ \alpha\ \alpha^2\ \ldots\ \alpha^{n-1}]$$

a pozrieme sa na to, čo by sa stalo, keby počas prenosu vznikli v odvysielanom kódovom slove dve chyby. Nech

$$e(x) = x^{i_1} + x^{i_2}.$$

Dosadením prvku α do prijatého polynómu dostávame hodnotu

$$v(\alpha) = e(\alpha) = \alpha^{i_1} + \alpha^{i_2}.$$

Táto hodnota na určenie pozícií chýb nestačí a preto potrebujeme nájsť ďalšie vzťahy medzi $\alpha^{i_1}, \alpha^{i_2}$. Skúsime rozšíriť kontrolnú maticu H pridaním ďalšieho riadku tak, aby násobenie prijatého vektora kontrolnou maticou viedlo k sústave rovníc s neznámymi

$\alpha^{i_1}, \alpha^{i_2}$. Nový riadok kontrolnej matice bude mať rovnaký tvar ako riadok pôvodnej kontrolnej matice; t.j. $1\ \beta\ \beta^2\ \ldots\ \beta^{n-1}$. Ak $\beta = \alpha^j$, (α je primitívny prvok poľa $GF(q^m)$) tak súčin $\mathbf{v} \cdot H^\top$ možno zapísať v podobe sústavy dvoch rovníc

$$v(\alpha) = S_1, \quad v(\beta) = v(\alpha^j) = S_j$$

Ako vybrať prvok β? Je zrejmé, že $\beta \neq \alpha$, pretože v opačnom prípade by kontrolná matica H obsahovala dva rovnaké riadky a súčin $\mathbf{v} \cdot H^\top$ by viedol k „sústave“ dvoch identických rovníc. Ako sa ukáže neskôr β nemôže patriť do množiny (cyklu) $\alpha, \alpha^2, \alpha^4, \ldots \alpha^{2^j}, \ldots$, pretože v tomto prípade by rovnice v sústave zadanej súčinom $\mathbf{v} \cdot H^\top$ boli závislé. To znamená, že spomedzi kandidátov na prvok β vypadli α, α^2 a prvým prirodzeným kandidátom je α^3. Kontrolná matica bude po rozšírení vyzerať nasledovne:[3]

$$H = \begin{bmatrix} 1 & \alpha & \alpha^2 & \ldots & \alpha^{n-1} \\ 1 & \alpha^3 & \alpha^6 & \ldots & \alpha^{3(n-1)} \end{bmatrix}.$$

Kontrolná matica H určuje cyklický kód $\mathcal{C}$, ktorý sa vyznačuje tým, že všetky kódové polynómy majú korene α, α^3, pretože

$$\left[\mathbf{u} \in \mathcal{C} \leftrightarrow \mathbf{u} \cdot H^\top = 0\right] \leftrightarrow \left[u(\alpha) = u(\alpha^3) = 0\right].$$

To znamená, že generujúcim polynómom kódu $\mathcal{C}$ bude

$$g(x) = \mathrm{lcm}(m_\alpha(x), m_{\alpha^3}(x)),$$

resp. kedže minimálne polynómy prvkov α, α^3 sú rôzne, nemôžu mať spoločný faktor, a teda

$$g(x) = m_\alpha(x) \cdot m_{\alpha^3}(x).$$

Ak pri prenose kódového slova vznikli 2 chyby; $e(x) = x^{i_1} + x^{i_2}$, tak pri dekódovaní vyčíslime hodnotu prijatého polynómu $v(x)$ v prvkoch α, α^3 a dostávame sústavu

$$\begin{aligned} v(\alpha) &= e(\alpha) = \alpha^{i_1} + \alpha^{i_2} \\ v(\alpha^3) &= e(\alpha^3) = \alpha^{3i_1} + \alpha^{3i_2}. \end{aligned} \tag{10.1}$$

Aby sme trocha zjednodušili zápis, zavedieme označenie

$$X_1 = \alpha^{i_1}, \quad X_2 = \alpha^{i_2}$$

a prepíšeme sústavu (10.1) do prehľadnejšej podoby:

$$\begin{aligned} X_1 + X_2 &= S_1 \\ X_1^3 + X_2^3 &= S_3. \end{aligned} \tag{10.2}$$

Hodnoty $X_1, X_2 \in GF(2^m)$ sa nazývajú *lokátory chýb*. Ak sa nám totiž podarí na základe hodnôt syndrómu vypočítať hodnoty lokátorov chýb X_1, X_2, tak potom vypočítame $\log_\alpha(X_1) = i_1$ a $\log_\alpha(X_2) = i_2$ a tak určíme pozície i_1, i_2 na ktorých vznikli chyby. Sústavu

[3] Kontrolná matica H zapísaná v binárnej podobe nad poľom $GF(2)$ má $2m$ riadkov.

(10.2) nebudeme riešiť priamo, ale spravíme trochu umelý krok. Zavedieme pomocný polynóm $\Lambda(x)$ nad poľom $GF(2^m)$, ktorého koreňmi budú práve lokátory chýb a ukážeme, že pomocou hodnôt syndrómu dokážeme vyjadriť koeficienty uvedeného polynómu:

$$\Lambda(x) = (x - X_1)(x - X_2) = x^2 + (X_1 + X_2) \cdot x + X_1 \cdot X_2. \tag{10.3}$$

Potrebujeme nájsť explicitné vyjadrenie súčinu $X_1 \cdot X_2$ z (10.3) pomocou známych hodnôt syndrómu. Spočítame

$$\frac{S_3}{S_1} = \frac{X_1^3 + X_2^3}{X_1 + X_2} = X_1^2 + X_1 \cdot X_2 + X_2^2 = (X_1 + X_2)^2 + X_1 \cdot X_2 = S_1^2 + X_1 \cdot X_2.$$

Teraz vyjadríme súčin lokátorov

$$X_1 \cdot X_2 = \frac{S_3}{S_1} + S_1^2 = \frac{S_3 + S_1^3}{S_1}.$$

Nakoniec vyjadríme explicitne pomocou syndrómu koeficienty polynómu lokátorov chýb:

$$\Lambda(x) = x^2 + S_1 \cdot x + \frac{S_3 + S_1^3}{S_1}. \tag{10.4}$$

Keďže koeficienty polynómu $\Lambda(x)$ dokážeme vyjadriť pomocou hodnôt syndrómu, polynóm lokátorov chýb na základe prijatého slova dokážeme zostrojiť a nájdením jeho koreňov určiť aj pozície, na ktorých sú v prijatom slove chyby. Korene polynómu $\Lambda(x)$ možno v prípade malého m nájsť úplným preberaním. Existujú však aj efektívne metódy faktorizácie kvadratických polynómov nad poľom $GF(2^m)$, pomocou ktorých možno nájsť korene polynómu $\Lambda(x)$ rýchlejšie ako úplným preberaním. Zostrojený kód opravuje 2 chyby. Pozrime sa ešte na to, čo sa stane, ak pri prenose vzniknú chyby inej váhy ako 2.

Chyba váhy 0 V tomto prípade je chybový polynóm $e(x)$ nulový, $v(x) = u(x)$ a

$$S_1 = v(\alpha) = v(\alpha^3) = S_3 = 0.$$

Chyba váhy 1 Chybový polynóm $e(x) = x^i$ a syndróm bude vyzerať nasledovne:

$$\begin{aligned} S_1 &= v(\alpha) = \alpha^i; \\ S_3 &= v(\alpha^3) = \alpha^{3i} = S_1^3. \end{aligned}$$

Chyba váhy > 2 Uvedený BCH kód opravuje chyby váhy 2. To znamená, že jeho minimálna vzdialenosť je 5. Chyby váhy 3 a väčšej môže kód odhaliť, ale nedokáže ich správne opraviť.

Zhrnieme teraz získané poznatky a uvedieme dekóder binárneho BCH kódu opravujúceho 2 chyby [16].

1. Vypočítaj hodnoty syndrómu

$$v(\alpha) = S_1 \quad v(\alpha^3) = S_3$$

2. Ak $S_1 = S_3 = 0$, bolo prijaté kódové slovo. (Predpokladáme, že pri prenose nevznikla chyba).

3. Ak $S_1 \neq 0$ a $S_3 = S_1^3$, predpokladáme, že pri prenose vznikla chyba váhy 1. Lokátor chyby je $S_1 = X_1$ a hodnota $\log_\alpha S_1 = i$ určuje miesto, na ktorom v prijatom slove vznikla chyba.

4. Ak $S_1 \neq 0$ a $S_3 \neq S_1^3$, predpokladáme, že pri prenose vznikla chyba váhy 2. Zostrojíme polynóm lokátorov chýb $\Lambda(x)$ a nájdeme jeho korene. Ak má $\Lambda(x)$ dva rozličné korene $X_1 \neq X_2$, tak chyby nastali na pozíciách $i_1, i_2; X_1 = \alpha^{i_1}, X_2 = \alpha^{i_2}$. V opačnom prípade (polynóm lokátorov chýb $\Lambda(x)$ nemá dva rozličné korene) nastala počas prenosu neopraviteľná (ale odhaliteľná) chyba.

5. Ak $S_1 = 0, S_3 \neq 0$, počas prenosu došlo ku chybe váhy ≥ 3. Túto sme síce pri dekódovaní odhalili, ale nedokážeme ju opraviť.

Ilustrujeme popísanú metódu na príklade.

Príklad 10.1. *Zostrojíme binárny BCH (15,7)-kód opravujúci 2 chyby a ukážeme, ako sa dekódujú prijaté slová, zaťažené chybami váhy 0, 1, 2 a 3. Budeme potrebovať konečné pole* $GF(2^4)$. *Jeho konštrukcia je podrobne popísaná časti 15.4 a preto len pripomenieme, že stačí faktorizovať okruh polynómov* $GF(2)[x]$ *vhodným ireducibilným polynómom stupňa 4 nad poľom* $GF(2)$. *Vyberieme polynóm* $1+x+x^4$ *a faktorový okruh polynómov* $GF(2)[x]/1+x+x^4$ *predstavuje hľadané pole. Označíme symbolom* α *primitívny prvok poľa* $GF(2^4)$. *Prvky poľa* $GF(2^4)$ *vyjadrené v binárnom tvare a pomocou mocnín primitívneho prvku sú uvedené v tabuľke 15.4. Zostrojíme generujúci polynóm BCH (15,7)-kódu. Minimálne polynómy prvkov* α, α^3 *sú*

$$\begin{aligned} m_\alpha(x) &= 1 + x + x^4, \\ m_{\alpha^3}(x) &= 1 + x + x^2 + x^3 + x^4 \end{aligned}$$

a generujúci polynóm bude mať tvar

$$\begin{aligned} g(x) &= \operatorname{lcm}(m_\alpha(x), m_{\alpha^3}(x)) = m_\alpha(x) \cdot m_{\alpha^3}(x) = \\ &= 1 + x^4 + x^6 + x^7 + x^8. \end{aligned}$$

Potrebujeme ešte nejaké kódové slovo. Vynásobíme generujúci polynóm vhodným informačným polynómom a dostávame

$$u(x) = (1 + x + x^5) \cdot g(x) = 1 + x + x^4 + x^6 + x^{11} + x^{12} + x^{13}.$$

Budeme predpokladať, že pri prenose kódového slova $u(x)$ *vznikla chyba* $e(x)$ *a bolo prijaté slovo* $v(x) = u(x) + e(x)$.

Chyba váhy 0 $e(x) = 0$, $v(x) = u(x)$. *Využijeme binárnu reprezentáciu prvkov poľa* $GF(2^4)$ *(tabuľka 15.4) a vypočítame syndróm chyby. (Pri výpočtoch budeme využívať, že pole* $GF(2^4)$ *má charakteristiku 2, t.j. pre ľubovoľný prvok* $\beta \in GF(2^4)$ *platí*

$\beta+\beta=0$ *a že* $\alpha^j=\alpha^{j \mod 15}$*.) Výpočet kvôli stručnosti uvádzame v tabuľkovej forme.*

$v(x)$	$v(\alpha)$		$v(\alpha^3)$	
1	1	0001	1	0001
x	α	0010	α^3	1000
x^4	α^4	0011	α^{12}	1111
x^6	α^6	1100	α^3	1000
x^{11}	α^{11}	1110	α^3	1000
x^{12}	α^{12}	1111	α^6	1100
x^{13}	α^{13}	1101	α^9	1010
	$S_1=$	0000	$S_3=$	0000

Keďže syndróm chyby je nulový, predpokladáme, že bolo prijaté odvysielané kódové slovo (polynóm) $u(x)=1+x+x^4+x^6+x^{11}+x^{12}+x^{13}$*. Informačný polynóm potom dostaneme vydelením prijatého polynómu generujúcim polynómom:*

$$i(x)=u(x):g(x)=1+x+x^5.$$

Chyba váhy 1 $e(x)=x^{11}$*, bolo prijaté slovo (polynóm)*

$$v(x)=1+x+x^4+x^6+x^{12}+x^{13}.$$

Vypočítame syndróm

$v(x)$	$v(\alpha)$		$v(\alpha^3)$	
1	1	0001	1	0001
x	α	0010	α^3	1000
x^4	α^4	0011	α^{12}	1111
x^6	α^6	1100	α^3	1000
x^{12}	α^{12}	1111	α^6	1100
x^{13}	α^{13}	1101	α^9	1010
	$S_1=$	1110	$S_3=$	1000

Syndróm chyby nadobúda v tomto prípade hodnoty $S_1=\alpha^{11}, S_3=\alpha^3$*. Obe zložky syndrómu sú nenulové, ale* $((\alpha)^{11})^3=\alpha^{33}=\alpha^3$*, a teda* $S_3=S_1^3$*. To znamená že v prijatom slove je chyba váhy 1. Keďže* $S_1=\alpha^{11}$*, chybový vektor je* $e(x)=x^{11}$*. Opravíme prijaté slovo:*

$$v(x)+e(x)=u(x)=1+x+x^4+x^6+x^{11}+x^{12}+x^{13}.$$

Chyba váhy 2 $e(x)=x^7+x^{12}$, *bolo prijaté slovo (polynóm)*

$$v(x)+e(x)=u(x)+e(x)=1+x+x^4+x^6+x^7+x^{11}+x^{13}.$$

Vypočítame syndróm

$v(x)$	$v(\alpha)$		$v(\alpha^3)$	
1	1	0001	1	0001
x	α	0010	α^3	1000
x^4	α^4	0011	α^{12}	1111
x^6	α^6	1100	α^3	1000
x^7	α^7	1011	α^6	1100
x^{11}	α^{11}	1110	α^3	1000
x^{13}	α^{13}	11101	α^9	1010
	$S_1=$	0100	$S_3=$	0000

t.j. $S_1 = \alpha^2, S_3 = 0$. *Keďže* $S_1 \neq 0$ *a* $S_3 \neq S_1^3$, *predpokladáme, že v slove* $v(x)$ *je chyba váhy 2. Zostrojíme polynóm lokátorov chýb (10.4)*

$$\Lambda(x) = \alpha^4 + \alpha^2 \cdot x + x^2$$

a postupným prehľadávaním poľa $GF(2^4)$ *nájdeme jeho korene.*

$$\begin{array}{lll}
v(0) = & \alpha^4 & \alpha^4 \neq 0 \\
v(1) = & \alpha^4 + \alpha^2 + 1 = & \alpha^5 \neq 0 \\
v(\alpha) = & \alpha^4 + \alpha^3 + \alpha^2 = & \alpha^{12} \neq 0 \\
v(\alpha^2) = & \alpha^4 + \alpha^4 + \alpha^4 = & \alpha^4 \neq 0 \\
v(\alpha^3) = & \alpha^4 + \alpha^5 + \alpha^6 = & \alpha^{14} \neq 0 \\
v(\alpha^4) = & \alpha^4 + \alpha^6 + \alpha^8 = & \alpha^9 \neq 0 \\
v(\alpha^5) = & \alpha^4 + \alpha^7 + \alpha^{10} = & \alpha^{12} \neq 0 \\
v(\alpha^6) = & \alpha^4 + \alpha^8 + \alpha^{12} = & \alpha^{14} \neq 0 \\
v(\alpha^7) = & \alpha^4 + \alpha^9 + \alpha^{14} = & 0 \qquad \checkmark
\end{array}$$

Našli sme prvý koreň, α^7. *Ušetríme si ďalšie prehľadávanie a vydelíme polynóm lokátorov chýb polynómom* $x + \alpha^7$. *Dostávame*

$$\Lambda(x) = (x + \alpha^7) \cdot (x + \alpha^{12}),$$

t.j. koreňmi polynómu $\Lambda(x)$ *sú* α^7, α^{12}. *Z toho vyplýva, že* $e(x) = x^7 + x^{12}$. *Opravíme prijaté slovo*

$$v(x) + e(x) = u(x) = 1 + x + x^4 + x^6 + x^{11} + x^{12} + x^{13}.$$

Chyba váhy 3 *Nech* $e(x) = x^{11} + x^{12} + x^{13}$ *a bolo prijaté slovo* $v(x) = 1 + x + x^4 + x^6$. *Syndróm chyby v tomto prípade bude*

$v(x)$	$v(\alpha)$		$v(\alpha^3)$	
1	1	0001	1	0001
x	α	0010	α^3	1000
x^4	α^4	0011	α^{12}	1111
x^6	α^6	1100	α^3	1000
	$S_1 =$	1100	$S_3 =$	1110

t.j. $S_1 = \alpha^6, S_3 = \alpha^{11}$ *a* $S_3 \neq S_1^3$. *Usúdime, že pri prenose nastala chyba váhy 2. Zostrojíme polynóm lokátorov chýb*

$$\Lambda(x) = \alpha^{14} + \alpha^6 \cdot x + x^2$$

a určíme jeho korene. Tými sú prvky α^5, α^9, *ktoré určujú chybový polynóm* $e'(x) = x^5 + x^9$. *Prijaté slovo upravíme na kódové slovo*

$$v(x) + e'(x) = 1 + x + x^4 + x^5 + x^6 + x^9,$$

ktorý sa však nezhoduje s odvysielaným kódovým slovom $u(x)$.

Tým istým spôsobom, ako sme zostrojili binárny kód opravujúci 2 chyby, možno zostrojiť aj BCH kód opravujúci tri chyby (kontrolná matica sa rozšíri o 3. riadok: $1, \alpha^5, \alpha^{10}, \ldots, \alpha^{5(n-1)}$). My sa touto konštrukciou nebudeme zaoberať, prípadných záujemcov odkazujeme na [16] a prikročíme k zovšeobecneniu a formalizácii definície BCH kódov.

10.2 Definícia BCH kódov

BCH kódy sú cyklické kódy nad poľom $GF(q)$, ktoré sú definované pomocou prvkov z rozšírenia pôvodného poľa, $GF(q^m)$.

Definícia 10.2.1. *Nech $\beta \in GF(q^m)$ je prvok rádu n; t je ľubovoľné prirodzené a l ľubovoľné celé číslo. Potom cyklický kód $\mathcal{C}$ s generujúcim polynómom*

$$g(x) = \operatorname{lcm}\left\{m_{\beta^{l+1}}(x), m_{\beta^{l+2}}(x) \ldots, m_{\beta^{l+2t}}(x)\right\},$$

kde $m_{\beta^j}(x)$ je minimálny polynóm prvku β^j, sa nazýva BCH kódom.

Ak $n = q^m - 1$; t.j. β je primitívny prvok poľa $GF(q^m)$, BCH kód dĺžky n sa nazýva *primitívnym BCH kódom*. Parameter l sa často kladie rovným nule (v mnohých ale nie všetkých prípadoch to vedie ku konštrukcii generujúceho polynómu minimálneho stupňa). BCH kódy s hodnotou parametra $l = 0$ sa nazývajú *BCH kódmi v úzkom zmysle.* Uvedieme niekoľko príkladov rôznych BCH kódov. Začneme binárnym kódom dĺžky 15, na ktorom budeme neskôr demonštrovať metódy dekódovania BCH kódov.

Príklad 10.2. *Nech $q = 2, l = 0, t = 3, m = 4$. BCH kód s danými parametrami existuje, je to binárny (15,5) kód opravujúci 3 chyby. Zadáme ho pomocou jeho generujúceho polynómu. Keďže ide o primitívny BCH kód, budeme potrebovať prvok rádu 15, t.j. primitívny prvok poľa $GF(2^4)$. Nech je α daný primitívny prvok. Koreňmi generujúceho polynómu budú prvky $\alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6$; t.j. generujúci polynóm bude*

$$g(x) = \operatorname{lcm}\{m_\alpha(x), m_{\alpha^2}(x), \ldots, m_{\alpha^6}(x)\}.$$

Prvky $\alpha, \alpha^2, \alpha^4$ však majú ten istý minimálny polynóm, podobne prvky α^3, α^6;

$$\begin{aligned} \alpha, \alpha^2, \alpha^4 &\leftrightarrow m_\alpha(x) = 1 + x + x^4, \\ \alpha^3, \alpha^6 &\leftrightarrow m_{\alpha^3}(x) = 1 + x + x^2 + x^3 + x^4, \\ \alpha^5 &\leftrightarrow m_{\alpha^5}(x) = 1 + x + x^2. \end{aligned}$$

To znamená, že

$$\begin{aligned} g(x) &= (1 + x + x^4) \cdot (1 + x + x^2 + x^3 + x^4) \cdot (1 + x + x^2) = \\ &= 1 + x + x^2 + x^4 + x^5 + x^8 + x^{10}. \end{aligned} \tag{10.5}$$

10.3 Hranica BCH kódov

Využijeme skutočnosť, že cyklické kódy sú zároveň lineárne kódy, určíme rang kontrolnej matice BCH kódu a potom na základe vety 8.1.2 a jej dôsledku určíme minimálnu vzdialenosť BCH kódov, resp. jej dolný odhad.

Veta 10.3.1. *Nech je $g(x)$ generujúci polynóm BCH kódu $\mathcal{C}$ dĺžky n, s koreňmi $\beta^{l+1}, \ldots, \beta^{l+2t}$, kde β je prvok rádu n poľa $GF(q^m)$. Potom minimálna vzdialenosť d kódu $\mathcal{C}$ je aspoň $2t + 1$.*

Dôkaz. Kontrolnú maticu BCH kódu $\mathcal{C}$ možno zapísať v tvare

$$H = \begin{bmatrix} 1 & \beta^{(l+1)} & \beta^{2(l+1)} & \dots & \beta^{(n-1)(l+1)} \\ 1 & \beta^{(l+2)} & \beta^{2(l+2)} & \dots & \beta^{(n-1)(l+2)} \\ \vdots & & & & \vdots \\ 1 & \beta^{(l+2t)} & \beta^{2(l+2t)} & \dots & \beta^{(n-1)(l+2t)} \end{bmatrix}$$

Ukážeme, že ľubovoľná štvorcová podmatica typu $(2t, 2t)$ kontrolnej matice H je regulárna; t.j. že ľubovoľných $2t$ stĺpcov kontrolnej matice je lineárne nezávislých (a teda minimálna váha kódu $\mathcal{C}$ je aspoň $2t + 1$). Určíme hodnotu determinantu štvorcovej podmatice tvorenej vybranými stĺpcami $j_1, j_2, \dots, j_{2t}$ kontrolnej matice H:

$$\det \begin{bmatrix} \beta^{j_1(l+1)} & \beta^{j_2(l+1)} & \dots & \beta^{j_{2t}(l+1)} \\ \beta^{j_1(l+2)} & \beta^{j_2(l+2)} & \dots & \beta^{j_{2t}(l+2)} \\ \vdots & & & \vdots \\ \beta^{j_1(l+2t)} & \beta^{j_2(l+2t)} & \dots & \beta^{j_{2t}(l+2t)} \end{bmatrix} =$$

$$= \beta^{(j_1+j_2+\dots+j_{2t})\cdot(l+1)} \det \begin{bmatrix} 1 & 1 & \dots & 1 \\ \beta^{j_1} & \beta^{j_2} & \dots & \beta^{j_{2t}} \\ \vdots & \vdots & & \vdots \\ \beta^{j_1(2t-1)} & \beta^{j_2(2t-1)} & \dots & \beta^{j_{2t}(2t-1)} \end{bmatrix}.$$

Druhá matica je Vandermondova matica, ktorej determinant má tvar

$$\prod_{i>k} \left(\beta^{j_i} - \beta^{j_k}\right). \tag{10.6}$$

Keďže β je prvok rádu n, prvky $\beta^{j_1}, \dots, \beta^{j_{2t}}$ sú navzájom rôzne, a teda determinant (10.6) je rôzny od nuly. □

Poznámka. Hodnota $2t + 1$ garantovaná predchádzajúcou vetou sa nazýva *konštrukčnou vzdialenosťou BCH kódu*. Je daná dĺžkou súvislej postupnosti mocnín prvku β ($\beta^{l+1}, \dots, \beta^{l+2t}$), ktoré sú koreňmi generujúceho polynómu kódu. Ale keďže generujúci polynóm $g(x)$ je definovaný ako najmenší spoločný násobok minimálnych polynómov svojich koreňov a minimálne polynómy prvkov $\beta^{l+1}, \dots, \beta^{l+2t}$ nie sú vo všeobecnosti lineárne, môžu mať aj iné korene, ktoré „predĺžia“ postupnosť $\beta^{l+1}, \dots, \beta^{l+2t}$. Potom bude skutočná minimálna vzdialenosť kódu väčšia ako jeho konštrukčná vzdialenosť. Existujú odhady skutočnej minimálnej vzdialenosti BCH kódov. Skôr, ako sa nimi budeme zaoberať, uvedieme niekoľko príkladov, na ktorých ilustrujeme rozdiel medzi konštrukčnou a minimálnou vzdialenosťou BCH kódu.

Príklad 10.3. *Uvažujme binárny BCH kód $\mathcal{C}$ dĺžky 31 (v úzkom zmysle) opravujúci 4 chyby, Nech α je primitívny prvok poľa $GF(2^5)$. Koreňmi generujúceho polynómu $g(x)$ kódu $\mathcal{C}$ musia byť prvky $\alpha, \alpha^2, \dots, \alpha^8$ poľa $GF(2^5)$; resp. generujúci polynóm $g(x)$ možno definovať ako*

$$g(x) = \operatorname{lcm} m_{\{}\alpha(x), m_{\alpha^2}(x), \dots, m_{\alpha^8}(x)\},$$

kde $m_{\alpha^j}(x)$ je minimálny polynóm prvku α^j. Bez toho, aby sme zostrojili generujúci polynóm $g(x)$, ukážeme, že minimálna vzdialenosť kódu $\mathcal{C}$ prevyšuje konštrukčnú vzdialenosť.

Využijeme na to skutočnosť, že ak má v poli charakteristiky 2 polynóm $f(x)$ *koreň* β*, tak potom má aj korene* $\beta^2, \beta^4, \ldots, \beta^{2^j}, \ldots$. *Rozdelíme prvky poľa* $GF(2^5)$ *do tried (cyklov); do jednej triedy dáme všetky prvky* $\beta, \beta^2, \beta^4, \ldots, \beta^{2^j}, \ldots$ *kde* $\beta \in GF(2^5)$. *Je zrejmé, že každej triede rozkladu zodpovedá jeden (minimálny) polynóm, ktorého koreňmi sú práve prvky z danej triedy.*

trieda					*minimálny polynóm*
0					$m_0(x)$
1					$m_1(x)$
α	α^2	α^4	α^8	α^{16}	$m_\alpha(x)$
α^3	α^6	α^{12}	α^{24}	α^{17}	$m_{\alpha^3}(x)$
α^5	α^{10}	α^{20}	α^9	α^{18}	$m_{\alpha^5}(x)$
α^7	α^{14}	α^{28}	α^{25}	α^{19}	$m_{\alpha^7}(x)$
α^{11}	α^{22}	α^{13}	α^{26}	α^{21}	$m_{\alpha^{11}}(x)$
α^{15}	α^{30}	α^{29}	α^{27}	α^{23}	$m_{\alpha^{15}}(x)$

Z uvedenej tabuľky rozkladu poľa $GF(2^5)$ *vyplýva, že*

$$g(x) = m_\alpha(x) \cdot m_{\alpha^3}(x) \cdot m_{\alpha^5}(x) \cdot m_{\alpha^7}(x)$$

a polynóm $g(x)$ *má teda okrem požadovaných koreňov* $\alpha, \alpha^2, \ldots, \alpha^8$ *aj korene* α^9 *a* α^{10}. *To však znamená, že minimálna vzdialenosť BCH kódu* C *je 11. Pre zaujímavosť dopočítame aj hodnoty jeho ostatných parametrov: stupeň generujúceho polynómu je 20, a teda počet informačných symbolov v kódovom slove je* 11. *Kód* C *je teda binárny (31,11) BCH kód (v úzkom zmysle) opravujúci 5 chýb.*

Rozdiel medzi konštrukčnou a minimálnou vzdialenosťou BCH kódu môže byť veľký. Berlekamp uvádza príklad binárneho BCH kódu v úzkom zmysle dĺžky $2^{12}-1$ s konštrukčnou vzdialenosťou 768, ktorého skutočná minimálna vzdialenosť je 819 (citované podľa [7]). Uvedené príklady môžu vzbudiť pochybnosti o tom, aká je výpovedná hodnota dolnej hranice BCH pre opravnú schopnosť BCH kódu. V práci [12] je uvedená rozsiahla tabuľka, ktorú zostavil Chen, obsahujúca všetky binárne cyklické kódy nepárnej dĺžky ($n \leq 65$) spolu s ich najdôležitejšími parametrami, vrátane dolnej hranice BCH a skutočnej minimálnej vzdialenosti. Hranica BCH sa od skutočnej minimálnej vzdialenosti odlišuje pomerne často a v niektorých prípadoch aj dosť výrazne. Vzťah medzi hranicou BCH a skutočnou minimálnou vzdialenosťou pre pomerne širokú podtriedu BCH kódov vyjadruje nasledujúca veta [12].

Veta 10.3.2. *Nech je* d_{BCH} *konštrukčná vzdialenosť primitívneho BCH kódu nad poľom* $GF(q)$. *Potom pre minimálnu vzdialenosť* d *tohto kódu platí*

$$d \leq q \cdot d_{BCH} + q - 2.$$

Dôkaz. Neuvádzame, čitateľ ho môže nájsť v práci [12]. □

Poznámka. V binárnom prípade skutočná minimálna vzdialenosť (primitívneho) BCH kódu neprevyšuje dvojnásobok konštrukčnej vzdialenosti.

10.4 PGZ algoritmus dekódovania BCH kódov

BCH kódy sú cyklické kódy, a preto na ich dekódovanie je možné použiť ľubovoľnú metódu dekódovania cyklických kódov. Existujú však efektívnejšie metódy dekódovania, navrhnuté špeciálne pre BCH kódy. V tejto časti popíšeme Petersonov-Gorensteinov-Zierlerov algoritmus dekódovania.

Predpokladáme, že je daný BCH kód $\mathcal{C}$ dĺžky n opravujúci t chýb nad poľom $GF(q)$, s generujúcim polynómom $g(x)$. Kvôli zjednodušeniu výkladu budeme tiež predpokladať, že $\mathcal{C}$ je BCH kód v úzkom zmysle, t.j. že koreňmi $g(x)$ sú prvky $\beta, \beta^2, \ldots, \beta^{2t}$. (Prvok β nemusí byť primitívnym prvkom poľa $GF(q^m)$; zrejme stačí, aby jeho rád n delil $q^m - 1$.) Nech bolo odvysielané slovo $u(x)$ a počas prenosu nastala chyba váhy ν, $0 \leq \nu \leq t$, ktorú zapíšeme v podobe chybového polynómu:

$$e(x) = e_0 + e_1 x + e_2 x^2 + \cdots + e_{n-1} x^{n-1}.$$

Chybový polynóm má práve ν koeficientov nenulových. Označme tieto koeficienty $e_{i_1}, \ldots, e_{i_\nu}$ a vynechajme v $e(x)$ nulové členy. Chybový polynóm bude po redukcii vyzerať nasledovne

$$e(x) = e_{i_1} x^{i_1} + e_{i_2} x^{i_2} + \cdots + e_{i_\nu} x^{i_\nu}. \tag{10.7}$$

Prijali sme polynóm (slovo) $v(x) = u(x) + e(x)$. Pre prijaté slovo $v(x)$ možno vypočítať dva rozličné syndrómy:

- syndrómový polynóm $s(x) = v(x) \mod g(x)$ a
- syndróm $(S_1, S_2, \ldots, S_{2t})$, ktorého zložky (parciálne syndrómy) sa dajú vypočítať zo vzťahu $S_j = v(\beta^j) = e(\beta^j)$. Pripomenieme, že pre tento syndróm platí aj

  $$[S_1, S_2, \ldots, S_{2t}] = [v_0, v_1, \ldots, v_{n-1}] \cdot H^\top,$$

  kde H je kontrolná matica kódu $\mathcal{C}$ a $(v_0, v_1, \ldots, v_{n-1})$ prijaté slovo (koeficienty prijatého polynómu).

Vypočítame parciálne syndrómy dosadením koreňov generujúceho polynómu do prijatého polynómu $v(x)$:

$$S_j = v(\beta^j) = e_{i_1} \beta^{j \cdot i_1} + e_{i_2} \beta^{j \cdot i_2} + \cdots + e_{i_\nu} \beta^{j \cdot i_\nu},\ j = 1, \ldots, 2t.$$

Aby sme zjednodušili zápis, zavedieme podobne ako v prípade binárnych BCH kódov nové označenie:

- hodnoty $\beta^{i_1}, \ldots, \beta^{i_\nu}$ označíme symbolmi $X_1, \ldots, X_\nu$; $X_k = \beta^{i_k}$ a nazveme lokátormi chýb;
- hodnoty chýb označíme symbolmi Y_k; $e_{i_1} = Y_1, \ldots, e_{i_\nu} = Y_\nu$.

Pripomíname, že lokátory chýb sú prvky poľa $GF(q^m)$ a hodnoty chýb sú prvky poľa $GF(q)$. Parciálne syndrómy môžeme potom vyjadriť pomocou sústavy rovníc:

$$\begin{array}{ccccccc} S_1 & = & Y_1X_1 & + & \dots & + & Y_\nu X_\nu \\ S_2 & = & Y_1X_1^2 & + & \dots & + & Y_\nu X_\nu^2 \\ \vdots & \vdots & & & & \vdots & \\ S_{2t} & = & Y_1X_1^{2t} & + & \dots & + & Y_\nu X_\nu^{2t} \end{array} \tag{10.8}$$

Našou úlohou je nájsť na základe známych hodnôt syndrómu $S_1, \dots, S_{2t}$ neznáme hodnoty lokátorov chýb a hodnôt chýb: $X_1, \dots, X_\nu; Y_1, \dots, Y_\nu$. Ukážeme, že systém rovníc (10.8) má práve jedno riešenie. Vzhľadom na nelineárnosť systému (10.8) by jeho priame riešenie bolo náročné. Preto podobne ako v predchádzjúcej časti zavedieme pomocný polynóm $\Lambda(x)$ lokátorov chýb; tentoraz však v mierne modifikovanej podobe: jeho koreňmi nebudú lokátory chýb, ale prevrátené hodnoty lokátorov chýb.

$$\Lambda(x) = (1 - xX_1)(1 - xX_2)\dots(1 - xX_\nu) = 1 + \lambda_1 x + \lambda_2 x^2 + \dots + \lambda_\nu x^\nu. \tag{10.9}$$

Ak by sme poznali koeficienty $\lambda_1, \dots, \lambda_\nu$, dokázali by sme zostrojiť polynóm $\Lambda(x)$ a (napríklad úplným preberaním) nájsť jeho korene $X_1^{-1}, \dots, X_\nu^{-1}$. Problém je v tom, že nepoznáme ani len stupeň ν polynómu $\Lambda(x)$, t.j. váhu chyby ktorá vznikla pri prenose. Ukážeme, ako sa dajú určiť koeficienty polynómu $\Lambda(x)$ na základe známych hodnôt parciálnych syndrómov $S_1, \dots, S_{2t}$ pomocou riešenia systému **lineárnych** rovníc. Keďže koreňmi polynómu $\Lambda(x)$ sú prevrátené hodnoty lokátorov chýb, platí

$$\Lambda(X_i^{-1}) = 1 + \lambda_1 X_i^{-1} + \lambda_2 X_i^{-2} + \dots + \lambda_\nu X_i^{-\nu} = 0, \quad i = 1, \dots, \nu \tag{10.10}$$

Zbavme sa záporných mocnín lokátorov chýb tým, že vynásobíme rovnice sústavy (10.10) členmi $Y_i X_i^{j+\nu}$, kde $1 \le j \le \nu$:

$$Y_i X_i^{j+\nu} + \lambda_1 Y_i X_i^{j+\nu-1} + \lambda_2 Y_i X_i^{j+\nu-2} + \dots + \lambda_\nu Y_i X_i^j = 0, \quad i = 1, \dots, \nu \tag{10.11}$$

Teraz sčítame (10.12) cez $i = 1, \dots \nu$:

$$\sum_{i=1}^{\nu} Y_i X_i^{j+\nu} + \lambda_1 \sum_{i=1}^{\nu} Y_i X_i^{j+\nu-1} + \lambda_2 \sum_{i=1}^{\nu} Y_i X_i^{j+\nu-2} + \dots + \lambda_\nu \sum_{i=1}^{\nu} Y_i X_i^j = 0 \tag{10.12}$$

Ale (pozri (10.8)) posledná rovnosť predstavuje

$$S_{j+\nu} + \lambda_1 S_{j+\nu-1} + \lambda_2 S_{j+\nu-2} + \dots + \lambda_\nu S_j = 0,$$

resp. po presunutí člena $S_{j+\nu}$ na pravú stranu a preusporiadaní ostatných členov na ľavej strane rovnice dostávame

$$S_j \lambda_\nu + S_{j+1} \lambda_{\nu-1} + \dots + \lambda_1 S_{j+\nu-1} = -S_{j+\nu}. \tag{10.13}$$

keďže $\lambda \le \nu$, pre $1 \le j \le \nu$, indexy zložiek syndrómu zo sústavy (10.13) sú z intervalu $\langle 1, 2t \rangle$ a teda v sústave (10.13) sa vyskytujú len známe hodnoty syndrómu $S_1, \dots, S_{2t}$.

Sústavu (10.13) možno prehľadnejšie zapísať v maticovej forme:

$$\begin{bmatrix} S_1 & S_2 & \dots & S_{\nu-1} & S_\nu \\ S_2 & S_3 & \dots & S_\nu & S_{\nu+1} \\ S_3 & S_4 & \dots & S_{\nu+1} & S_{\nu+2} \\ \vdots & & & & \vdots \\ S_\nu & S_{\nu+1} & \dots & S_{2\nu-2} & S_{2\nu-1} \end{bmatrix} \cdot \begin{bmatrix} \lambda_\nu \\ \lambda_{\nu-1} \\ \lambda_{\nu-2} \\ \vdots \\ \lambda_1 \end{bmatrix} = \begin{bmatrix} -S_{\nu+1} \\ -S_{\nu+2} \\ -S_{\nu+3} \\ \vdots \\ -S_{2\nu} \end{bmatrix}. \tag{10.14}$$

Kvôli zjednodušeniu zápisu v ďalšom texte zavedieme nasledujúce označenie

$$\begin{bmatrix} S_1 & S_2 & \dots & S_{\nu-1} & S_\nu \\ S_2 & S_3 & \dots & S_\nu & S_{\nu+1} \\ S_3 & S_4 & \dots & S_{\nu+1} & S_{\nu+2} \\ \vdots & & & & \vdots \\ S_\nu & S_{\nu+1} & \dots & S_{2\nu-2} & S_{2\nu-1} \end{bmatrix} = M_\nu. \tag{10.15}$$

Ak by matica M_ν bola regulárna, tak by bolo možné riešiť sústavu (10.14). V nasledujúcej leme dokážeme, za akých podmienok je M_ν regulárna.

Lema 6. *Nech pri prenose kódového slova došlo k chybe váhy* $\nu \leq t$. *Potom je matica* M_ν *regulárna a matica* M_μ, *kde* $\mu > \nu$ *singulárna.*

Dôkaz Maticu M_μ, $0 \leq \nu \leq \mu \leq t$ možno rozložiť na súčin troch matíc;

$$M_\mu = A \cdot B \cdot A^\top,$$

kde

$$A = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ X_1 & X_2 & X_3 & \dots & x_\mu \\ X_1^2 & X_2^2 & X_3^2 & \dots & x_\mu^2 \\ \vdots & & & \vdots & \\ X_1^{\mu-1} & X_2^{\mu-1} & X_3^{\mu-1} & \dots & x_\mu^{\mu-1} \end{bmatrix}$$

a B je diagonálna matica

$$B = \begin{bmatrix} X_1Y_1 & 0 & 0 & \dots & 0 \\ 0 & X_2Y_2 & 0\dots & 0 & \\ 0 & 0 & X_3Y_3 & \dots & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & 0 & \dots & X_\nu Y_\nu \end{bmatrix}$$

Zrejme

$$\det(M_\mu) = \det(A) \cdot \det(B) \cdot \det(A^\top).$$

Ak $\mu > \nu$, medzi hodnotami chýb $Y_1, \dots, Y_\mu$ je aspoň jedna nulová. To znamená, že v matici B sa na diagonále vyskytuje aspoň jedna nulová hodnota a jej determinant je nulový. Na druhej strane, ak $\mu = \nu$, $\det(B) \neq 0$ a keďže A je Vandermondova matica a lokátory chýb $X_1, \dots, X_\mu$ sú rôzne, $\det(A) = \det(A^\top) \neq 0$. □

Získané poznatky využijeme pri návrhu dekódera BCH kódov.

Petersonov-Gorensteinov-Zierlerov dekóder

1. Na základe prijatého slova $v(x)$ vypočítaj syndróm $S_1, \ldots, S_{2t}$; $S_j = v(\beta^j)$.

2. Nájdi najväčšie prirodzené ν také, že matica M_ν je regulárna.

3. Vyrieš systém lineárnych rovníc (10.14) a urči koeficienty $\lambda_1, \ldots, \lambda_\nu$. Zostroj polynóm lokátorov chýb $\Lambda(x)$.

4. Nájdi korene polynómu lokátorov chýb $\Lambda(x)$ (úplným preberaním, alebo faktorizáciou) a urči pozície chýb. Ak je daný BCH kód binárny, tak invertuj bity na pozíciách určených lokátormi chýb a skonči, ináč pokračuj krokom 5.

5. Vyrieš systém lineárnych rovníc pre hodnoty chýb

$$\begin{array}{ccccccccc} Y_1X_1 & + & Y_2X_2 & + & \ldots & + & Y_\nu X_\nu & = & S_1 \\ Y_1X_1^2 & + & Y_2X_2^2 & + & \ldots & + & Y_\nu X_\nu^2 & = & S_2 \\ \vdots & & & & & & & & \vdots \\ Y_1X_1^\nu & + & Y_2X_2^\nu & + & \ldots & + & Y_\nu X_\nu^\nu & = & S_\nu \end{array} \tag{10.16}$$

K úplnosti popísanej metódy zostáva ešte ukázať, že systém (10.16) má jediné riešenie.

Lema 7. *Ak sú $X_1, \ldots, X_\nu$ rozličné lokátory chýb, tak potom má systém (10.16) jediné riešenie.*

Dôkaz Determinant sústavy (10.16) sa dá upraviť na Vandermondov determinant

$$\det \begin{bmatrix} X_1 & X_2 & \ldots & X_\nu \\ X_1^2 & X_2^2 & \ldots & X_\nu^2 \\ \vdots & & & \vdots \\ X_1^\nu & X_2^\nu & \ldots & X_\nu^\nu \end{bmatrix} = (X_1X_2\ldots X_\nu) \det \begin{bmatrix} 1 & 1 & \ldots & 1 \\ X_1 & X_2 & \ldots & X_\nu \\ \vdots & & & \vdots \\ X_1^{\nu-1} & X_2^{\nu-1} & \ldots & X_\nu^{\nu-1} \end{bmatrix}$$

Keďže lokátory chýb sú nenulové a rôzne, determinant sústavy (10.16) je nenulový, a teda sústava (10.16) má riešenie. □

Ilustrujeme použitie PGZ dekódera na niekoľkých príkladoch.

Príklad 10.4. *Uvažujeme binárny (15,5) BCH kód z predchádzajúceho príkladu. Nech bolo odvysielané kódové slovo*

$$u(x) = 1 + x^3 + x^5 + x^{10} + x^{11} + x^{12} + x^{14}$$

a pri prenose vznikla chyba $e(x)$ v dôsledku čoho bolo prijaté slovo $v(x) = u(x) + e(x)$.

1. *Chyba váhy 0. Syndróm chyby je*

$$S_1 = S_2 = S_3 = S_4 = S_5 = S_6 = 0.$$

Keďže syndróm chyby je rovný nule, predpokladáme, že pri prenose nedošlo k chybe a prijaté kódové slovo je zhodné s odvysielaným.

2. *Chyba váhy 1; napr.* $e(x) = x^{14}$*, bolo prijaté slovo* $v(x) = 1 + x^3 + x^5 + x^{10} + x^{11} + x^{12}$.

$$S_1 = \alpha^{14} \quad S_2 = \alpha^{13} \quad S_3 = \alpha^{12} \quad S_4 = \alpha^{11} \quad S_5 = \alpha^{10} \quad S_6 = \alpha^9$$

Určíme počet chýb, ν.

$$M_3 = \begin{bmatrix} \alpha^{14} & \alpha^{13} & \alpha^{12} \\ \alpha^{13} & \alpha^{12} & \alpha^{11} \\ \alpha^{12} & \alpha^{11} & \alpha^{10} \end{bmatrix}.$$

Ale $\det M_3 = 0$. *Vytvoríme maticu* M_2 *a vypočítame determinant tejto matice:*

$$M_2 = \begin{bmatrix} \alpha^{14} & \alpha^{13} \\ \alpha^{13} & \alpha^{12} \end{bmatrix}.$$

Keďže $\det M_2 = 0$ *v prijatom slove je najviac jedna chyba. Ale* $M_1 = \alpha^{14} \neq 0$*, a teda predpokladáme, že v prijatom slove je chyba váhy 1,* $S_1 = X^{14}$*, t.j.* $e(x) = x^{14}$ *a odvysielané slovo bolo* $u(x) = 1 + x^3 + x^5 + x^{10} + x^{11} + x^{12} + x^{14}$.

3. *Chyba váhy 2. Chybový polynóm je (napr.)* $e(x) = x^3 + x^7$*; bolo prijaté slovo*

$$v(x) = 1 + x^5 + x^7 + x^{10} + x^{11} + x^{12} + x^{14}.$$

Syndróm chyby je

$$S_1 = \alpha^4 \quad S_2 = \alpha^8 \quad S_3 = \alpha^5 \quad S_4 = \alpha \quad S_5 = \alpha^{10} \quad S_6 = \alpha^{10}$$

Určíme počet chýb, ν.

$$M_3 = \begin{bmatrix} \alpha^4 & \alpha^8 & \alpha^5 \\ \alpha^8 & \alpha^5 & \alpha \\ \alpha^5 & \alpha & \alpha^{10} \end{bmatrix}.$$

Keďže $\det M_3 = 0$, *vytvoríme maticu* M_2 *a vypočítame jej determinant:*

$$M_2 = \begin{bmatrix} \alpha^4 & \alpha^8 \\ \alpha^8 & \alpha^5 \end{bmatrix}; \qquad \det M_2 = \alpha^3 \neq 0.$$

V prijatom slove je pravdepodobne chyba váhy 2. Určíme koeficienty polynómu lokátorov chýb:

$$\lambda_2 = \frac{\det \begin{bmatrix} \alpha^5 & \alpha^8 \\ \alpha & \alpha^5 \end{bmatrix}}{\det M_2} = \frac{\alpha^{13}}{\alpha^3} = \alpha^{10}.$$

Podobne

$$\lambda_1 = \frac{\det \begin{bmatrix} \alpha^4 & \alpha^5 \\ \alpha^8 & \alpha \end{bmatrix}}{\det M_2} = \frac{\alpha^7}{\alpha^3} = \alpha^4.$$

Vyjadríme polynóm lokátorov chýb a nájdeme jeho korene:

$$\Lambda(x) = 1 + \alpha^4 x + \alpha^{10} x^2.$$

Korene polynómu $\Lambda(x)$ *sme našli úplným preberaním (a faktorizáciou polynómu); na tomto mieste uvedieme len jeho výsledok:*

$$\Lambda(\alpha^{12}) = 1 + \alpha + \alpha^4 = 0, \quad \Lambda(\alpha^8) = 1 + \alpha^{12} + \alpha^{11} = 0;$$

t.j. hľadané korene sú α^{12}, α^8 *a chyby vznikli na pozíciách 3 (*$\alpha^{12} = \alpha^{-3}$*) a 7 (*$\alpha^7 = \alpha^{-8}$*).*

4. *Chyba váhy 3. Predpokladajme kvôli jednoduchosti, že* $e(x) = x^5 + x^{10} + x^{13}$ *a bolo prijaté slovo*

$$v(x) = 1 + x^3 + x^{11} + x^{12} + x^{13} + x^{14}.$$

Syndróm chyby je

$$S_1 = \alpha^6 \quad S_2 = \alpha^{12} \quad S_3 = \alpha^9 \quad S_4 = \alpha^9 \quad S_5 = \alpha^{10} \quad S_6 = \alpha^3.$$

Keďže $\det M_3 = 1$*, predpokladáme, že počas prenosu vznikla chyba váhy 3. Vypočítame koeficienty polynómu lokátorov chýb a zostavíme* $\Lambda(x)$*:*

$$\Lambda(x) = 1 + \alpha^6 x + \alpha^6 x^2 + \alpha^{13} x^3.$$

Koreňmi polynómu $\Lambda(x)$ *sú* $\alpha^{10}, \alpha^5, \alpha^2$*, ktoré určujú polynóm chýb* $e(x) = x^5 + x^{10} + x^{13}$*. Odčítame chybový polynóm od prijatého slova a dostaneme kódové slovo, o ktorom predpokladáme, že sa zhoduje s odvysielaným kódovým slovom.*

Na záver uvedieme ešte príklad konštrukcie a dekódovania ternárneho BCH kódu. (Tento príklad ešte doplníme a upravíme)

Príklad 10.5. [4] *Skonštruujeme ternárny BCH kód dĺžky 26 opravujúci tri chyby. Najprv zostrojíme pole* $GF(3^3)$ *tak, že faktorizujeme okruh polynómov nad poľom* $GF(3)$ *ireducibilným polynómom tretieho stupňa* $x^3 + 2x + 1$*. Tento polynóm je zároveň primitívnym polynómom a jeho koreň, ktorý označíme symbolom* α*, je primitívnym prvkom poľa* $GF(3^3) = GF(3)[x]/x^3 + 2x + 1$*. Na získanie lepšej predstavy o poli* $GF(3^3)$ *vyjadríme všetky jeho nenulové prvky pomocou mocnín primitívneho prvku, lídra triedy rozkladu faktorového okruhu-poľa* $GF(3)[x]/x^3 + 2x + 1$*, lineárnej kombinácie mocnín primitívneho prvku a vektora koeficientov tejto lineárnej kombinácie. (Vo výpočtoch budeme však využívať prvú a poslednú reprezentáciu prvkov poľa.)*

Nájdeme vhodný generujúci polynóm BCH kódu. V tabuľke sú uvedené minimálne polynómy jednotlivých prvkov konečného poľa $GF(3^3)$ *(vrátane nulového):*

0	x
α^0	$x + 2$
$\alpha^1\ \alpha^3\ \alpha^9$	$x^3 + 2x + 1$
$\alpha^2\ \alpha^6\ \alpha^{18}$	$x^3 + x^2 + x + 2$
$\alpha^4\ \alpha^{12}\ \alpha^{10}$	$x^3 + x^2 + 2$
$\alpha^5\ \alpha^{15}\ \alpha^{19}$	$x^3 + 2x^2 + x + 1$
$\alpha^7\ \alpha^{21}\ \alpha^{11}$	$x^3 + x^2 + 2x + 1$
$\alpha^8\ \alpha^{24}\ \alpha^{20}$	$2x^3 + x^2 + x + 1$
α^{13}	$x + 1$
$\alpha^{14}\ \alpha^{16}\ \alpha^{22}$	$2x^3 + x + 1$
$\alpha^{17}\ \alpha^{25}\ \alpha^{23}$	$x^3 + 2x^2 + 1$

Keďže kód, ktorý konštruujeme, má opravovať 3 chyby, koreňmi generujúceho polynómu musí byť postupnosť šiestich za sebou nasledujúcich mocnín primitívneho prvku

[4] Tento príklad je upravenou verziou domácej úlohy, ktorú vypracovala Monika Steinová

0			0			0	000
α^0			1			1	001
α^1		x			α		010
α^2	x^2			α^2			100
α^3		x	$+2$		α	$+2$	012
α^4	x^2	$+2x$		α^2	$+2\alpha$		120
α^5	$2x^2$	$+x$	$+2$	$2\alpha^2$	$+\alpha$	$+2$	212
α^6	x^2	$+x$	$+1$	α^2	$+\alpha$	$+1$	111
α^7	x^2	$+2x$	$+2$	α^2	$+2\alpha$	$+2$	122
α^8	$2x^2$		$+2$	$2\alpha^2$		$+2$	202
α^9		$+x$	$+1$		$+\alpha$	$+1$	011
α^{10}	x^2	$+x$		α^2	$+\alpha$		110
α^{11}	x^2	$+x$	$+2$	α^2	$+\alpha$	$+2$	112
α^{12}	x^2		$+2$	α^2		$+2$	102
α^{13}			2			2	002
α^{14}		$2x$			2α		020
α^{15}	$2x^2$			$2\alpha^2$			200
α^{16}		$2x$	$+1$		2α	$+1$	021
α^{17}	$2x^2$	$+x$		$2\alpha^2$	$+\alpha$		210
α^{18}	x^2	$+2x$	$+1$	α^2	$+2\alpha$	$+1$	121
α^{19}	$2x^2$	$+2x$	$+2$	$2\alpha^2$	$+2\alpha$	$+2$	222
α^{20}	$2x^2$	$+x$	$+1$	$2\alpha^2$	$+\alpha$	$+1$	211
α^{21}	x^2		$+1$	α^2		$+1$	101
α^{22}		$2x$	$+2$		2α	$+2$	022
α^{23}	$2x^2$	$+2x$		$2\alpha^2$	$+2\alpha$		220
α^{24}	$2x^2$	$+2x$	$+2$	$2\alpha^2$	$+2\alpha$	$+1$	221
α^{25}	$2x^2$		$+2$	$2\alpha^2$		$+1$	201
α^{26}			1			1	001

Tabuľka 10.1: Pole $GF(3^3)$.

α. Nech sú to napríklad prvky $\alpha, \alpha^2, \ldots, \alpha^6$. Potom je podľa definície BCH kódu generujúcim polynómom kódu polynóm

$$g'(x) = \operatorname{lcm}(m_\alpha(x), \ldots, m_{\alpha^6}(x)) = (x^3+2x+1)(x^3+x^2+x+2)(x^3+x^2+2)(x^3+2x^2+x+1).$$

Ak pridáme ku koreňom generujúceho polynómu aj prvok α^0 (ktorého minimálny polynóm je lineárny) minimálnu vzdialenosť konštruovaného BCH kódu rozšírime na 8 za cenu pridania jedného kontrolného symbolu. Generujúcim polynómom tohto kódu bude

$$g(x) = g'(x) \cdot (x+2) = x^{13} + 2x^{11} + 2x^7 + x^6 + x^4 + x^3 + 2x + 2.$$

Kód zadaný generujúcim polynómom $g(x)$ je ternárny (26,13) BCH kód v úzkom zmysle opravujúci chyby do váhy 3 a odhaľujúci chyby váhy 4.

Teraz ukážeme ako vyzerá kódovanie a dekódovanie ternárnej informácie pomocou ternárneho (26,13) BCH kódu. Nech je $i(x) = x^{11} + 2x^{10} + 2x^9 + x^8 + x^5 + 2x^4 + 2x^3 + x^2$ informačný polynóm. Pri systematickom kódovaní najprv vypočítame polynóm $r(x)$:

$$\begin{aligned} r(x) &= (x^{11} + 2x^{10} + 2x^9 + x^8 + x^5 + 2x^4 + 2x^3 + x^2) * x^{13}) \bmod g(x) = \\ &= x^{24} + 2x^{23} + 2x^{22} + x^{21} + x^{18} + 2x^{17} + 2x^{16} + x^{15} \bmod g(x) = \\ &= 2x^{12} + 2x^{11} + x^{10} + 2x^9 + 2x^6 + 2x^4 + x^3 + 2x^2 + 1 \end{aligned}$$

a potom kódové slovo $c(x)$ vyjadríme ako

$$\begin{aligned} c(x) &= i(x) * x^{13} - r(x) = x^{24} + 2x^{23} + 2x^{22} + x^{21} + x^{18} + 2x^{17} + 2x^{16} + x^{15} + \\ &+ x^{12} + x^{11} + 2x^{10} + x^9 + x^6 + x^4 + 2x^3 + x^2 + 2 \end{aligned}$$

Nech pri prenose vznikla chyba váhy 3 zadaná chybovým polynómom $e(x) = x^{14} + 2x^{12} + x^9$ a bol prijatý polynóm

$$v(x) = x^{24} + 2x^{23} + 2x^{22} + x^{21} + x^{18} + 2x^{17} + 2x^{16} + x^{15} + 2x^{14} + x^{11} + 2x^{10} + 2x^9 + x^6 + x^4 + 2x^3 + x^2 + 2.$$

Zistíme či nastala chyba a ak áno, opravime ju. Vypočítame hodnoty syndrómu $S_1, \ldots S_6; S_i = v(\alpha^i),\ i = 1, \ldots, 6.$

$$\begin{array}{ccc} S_1 = \alpha^{19} & S_2 = \alpha^2 & S_3 = \alpha^5 \\ S_4 = \alpha^0 & S_5 = \alpha^8 & S_6 = \alpha^6 \end{array}$$

Teraz zostrojíme maticu M_3 a vypočítame jej determinant:

$$\det M_3 = \det \begin{bmatrix} S_1 & S_2 & S_3 \\ S_2 & S_3 & S_4 \\ S_3 & S_4 & S_5 \end{bmatrix} = \det \begin{bmatrix} \alpha^{19} & \alpha^2 & \alpha^5 \\ \alpha^2 & \alpha^5 & \alpha^0 \\ \alpha^5 & \alpha^0 & \alpha^8 \end{bmatrix} = \alpha^{11}$$

Determinant matice M_3 nie je nulový a na základe toho predpokladáme, že pri prenose vznikli tri chyby. Hladáme teraz kubický polynóm lokátorov chyb

$$\Lambda(z) = 1 + \lambda_1 z + \lambda_2 z^2 + \lambda_3 z^3 = (1 - x_1 z)(1 - x_2 z)(1 - x_3 z)$$

Pre hladané koeficienty platí nasledujúca sústava rovníc:

$$\begin{array}{rcrcrcrcl} S_1\lambda_3 & + & S_2\lambda_2 & + & S_3\lambda_1 & + & S_4\lambda_0 & = & 0 \\ S_2\lambda_3 & + & S_3\lambda_2 & + & S_4\lambda_1 & + & S_5\lambda_0 & = & 0 \\ S_3\lambda_3 & + & S_4\lambda_2 & + & S_5\lambda_1 & + & S_6\lambda_0 & = & 0 \end{array}$$

resp.

$$\begin{array}{rcrcrcrcl} \alpha^{19}\lambda_3 & + & \alpha^2\lambda_2 & + & \alpha^5\lambda_1 & + & \alpha^0 & = & 0 \\ \alpha^2\lambda_3 & + & \alpha^5\lambda_2 & + & \alpha^0\lambda_1 & + & \alpha^8 & = & 0 \\ \alpha^5\lambda_3 & + & \alpha^0\lambda_2 & + & \alpha^8\lambda_1 & + & \alpha^6 & = & 0 \end{array}$$

Riešením tejto sústavy dostávame:

$$\lambda_3 = \alpha^{22}\ \ \lambda_2 = \alpha^{22}\ \ \lambda_1 = \alpha^{15}\ \ \lambda_0 = \alpha^0$$

a polynóm lokátorov chýb bude

$$\Lambda(z) = 1 + \alpha^{15}z + \alpha^{22}z^2 + \alpha^{22}z^3.$$

Koreňmi polynómu $\Lambda(z)$ sú prvky $\alpha^{12}, \alpha^{14}, \alpha^{17}$ a teda chyby vznikli pri x^{26-12}, x^{26-14}, a x^{26-17}. Pri binárnom kóde by sme v tomto momente skončili, pretože odhalenie pozície chyby stačí aj na opravenie tejto chyby. V ternárnom prípade však ešte treba zistiť, aká chyba nastala. Označme si teda Y_1, Y_2, Y_3 koeficienty, ktoré boli v chybovom polynóme postupne pri členoch X_1, X_2, X_3. Tieto sú viazané vzťahmi, vyjadrenými v nasledujúcej sústave (10.16):

$$\begin{array}{rcrcrcl} \alpha^{12}Y_1 & + & \alpha^{14}Y_2 & + & \alpha^9Y_3 & = & \alpha^{19} \\ \alpha^{24}Y_1 & + & \alpha^2Y_2 & + & \alpha^{18}Y_3 & = & \alpha^2 \\ \alpha^{10}Y_1 & + & \alpha^{16}Y_2 & + & \alpha^1Y_3 & = & \alpha^5 \end{array}$$

Riešením tejto sústavy rovníc je: $Y_1 = 2$, $Y_2 = 2$ a $Y_3 = 1$ a teda chyba, ktorá nastala je $e(x) = X_1Y_1 + X_2Y_2 + X_3Y_3 = 2x^{14} + 2x^{12} + x^9$.

Prijatý polynóm $v(x)$ upravíme na kódový polynóm $u(x) = v(x) - e(x)$ a vypočítame informačný polynóm $u(x)$ div $x^{(26-13)} = i(x) = x^{11} + 2x^{10} + 2x^9 + x^8 + x^5 + 2x^4 + 2x^3 + x^2$.

Informačný polynóm $i(x) = x^{11} + 2x^{10} + 2x^9 + x^8 + x^5 + 2x^4 + 2x^3 + x^2$ budeme tentoraz kódovať nesystematicky, vynásobením generujúcim polynómom: $c(x) = g(x) \cdot i(x)$. V našom prípade dostávame kódové slovo

$$\begin{aligned} c(x) &= x^{24} + 2x^{23} + x^{22} + 2x^{21} + x^{20} + 2x^{19} + x^{17} + x^{16} + x^{15} + 2x^{14} + x^{12} + \\ &+ 2x^{10} + 2x^9 + x^7 + 2x^6 + x^5 + 2x^4 + 2x^2. \end{aligned}$$

Nech pri prenose vznikla chyba $e(x) = 2x^7 + x$. Potom bolo prijaté slovo

$$\begin{aligned} v(x) &= x^{24} + 2x^{23} + x^{22} + 2x^{21} + x^{20} + 2x^{19} + x^{17} + x^{16} + x^{15} + 2x^{14} + \\ &+ x^{12} + 2x^{10} + 2x^9 + 2x^6 + x^5 + 2x^4 + 2x^2 + x. \end{aligned}$$

Zistíme či v prijatom slove nastala chyba. Syndróm chyby slova $v(x)$ je

$$S_1 = \alpha^{24},\ S_2 = \alpha^{10},\ S_3 = \alpha^{20},\ S_4 = \alpha^{14},\ S_5 = \alpha^{25},\ S_6 = \alpha^4.$$

Keďže syndróm je nenulový, prijaté slovo je zaťažené nejakou chybou. Určíme váhu chyby, ν. *Zostavíme maticu* M_3 *a vypočítame jej determinant:*

$$\det M_3 = \det \begin{bmatrix} \alpha^{24} & \alpha^{10} & \alpha^{20} \\ \alpha^{10} & \alpha^{20} & \alpha^{14} \\ \alpha^{20} & \alpha^{14} & \alpha^{25} \end{bmatrix} = 0.$$

Yo znamená, že váha chyby je menšia než 3. Zostavíme teda maticu M_2 *a vypočítame jej determinant:*

$$\det M_2 = \begin{bmatrix} \alpha^{24} & \alpha^{10} \\ \alpha^{10} & \alpha^{20} \end{bmatrix} = \alpha^{17} \neq 0.$$

Z toho, že je matica M_3 *singulárna a matica* M_2 *regulárna, vyplýva, že pri prenose vznikli pravdepodobne dve chyby. Na ich určenie potrebujeme zostrojiť polynóm lokátorov chýb*

$$\Lambda(x) = 1 + \lambda_1 x + \lambda_2 x^2.$$

Riešením sústavy rovníc

$$\begin{array}{rcrcrcl} \alpha^{24}\lambda_2 & + & \alpha^{10}\lambda_1 & + & \alpha^{20} & = & 0 \\ \alpha^{10}\lambda_2 & + & \alpha^{20}\lambda_1 & + & \alpha^{14} & = & 0 \end{array}$$

určíme koeficienty polynómu $\Lambda(x)$

$$\lambda_2 = \alpha^8, \quad \lambda_1 = \alpha^{25}, \quad \lambda_0 = 1.$$

Dostávame polynóm

$$\lambda(x) = 1 + \alpha^{25} x + \alpha^8 x^2$$

ktorého korene sú $X_1 = \alpha^1$ *a* $X_2 = \alpha^7$.

Potrebujeme už len zistiť aké chyby nastali. Znova budeme riešiť sústavu rovníc (10.16)

$$\begin{array}{rcrcl} \alpha^1 Y_1 & + & \alpha^7 Y_2 & = & \alpha^{24} \\ \alpha^7 Y_1 & + & \alpha^{14} Y_2 & = & \alpha^{10} \end{array}$$

Riešením tejto sústavy rovníc je $Y_1 = 1$ *a* $Y_2 = 2$. *Chybový polynóm je* $e(x) = 2x^7 + 1x$ *a kódové slovo* $c(x) = \nu(x) - e(x)$. *Nakoniec určíme informačné slovo. Kedže bolo použité nesystematické kódovanie informačné slovo (polynóm) dostaneme vynásobením kódového slova* $c(x)$ *kontrolným polynómom* $h(x)$. *V našom prípade* $h(x) = x^{13} + x^{11} + x^9 + 2x^7 + 2x^6 + x^2 + 2x + 1$ *a informačný polynóm je*

$$h(x)u(x) = x^{11} + 2x^{10} + 2x^9 + x^8 + x^5 + 2x^4 + 2x^3 + x^2.$$

10.5 Iné metódy dekódovania BCH kódov

Výpočtovo najnáročnejšou časťou Peterson-Gorenstein-Zierlerovho algoritmu je riešenie sústavy lineárnych rovníc (10.14). Ak túto sústavu riešime výpočtom inverznej matice syndrómov, zložitosť výpočtu je $O(\nu^3)$, čo pre veľké hodnoty ν môže byť príliš veľa. Všimnime si však štruktúru matice syndrómov zo sústavy (10.14). Nech $S_1, S_2, \ldots, S_{2\nu}$ je postupnosť hodnôt syndrómov a

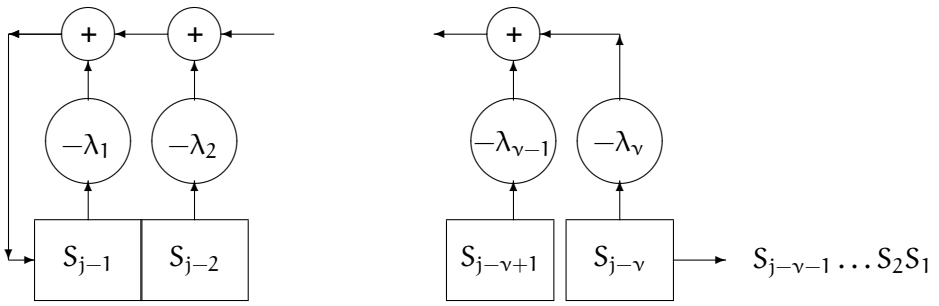
$$\Lambda(x) = 1 + \lambda_1 x + \lambda_2 x^2 + \cdots + \lambda_\nu x^\nu$$

je polynóm lokátorov chýb. Riadky matice syndrómov tvoria súvislé podpostupnosti dĺžky ν postupnosti hodnôt syndrómu: prvý riadok matice syndrómov tvorí podpostupnosť $S_1, \ldots, S_\nu$, druhý podpostupnosť $S_2, \ldots, S_{\nu+1}$, až napokon posledný riadok matice syndrómov tvorí podpostupnosť $S_\nu, \ldots, S_{2\nu-1}$. Naviac, hodnoty $S_{\nu+1}, \ldots, S_{2\nu}$ možno vyjadriť pomocou predchádzajúcich ν hodnôt syndrómu jednotným spôsobom:

$$S_j = -\sum_{i=1}^{\nu} \lambda_i S_{j-i}; \quad j = \nu + 1, \ldots, 2\nu. \tag{10.17}$$

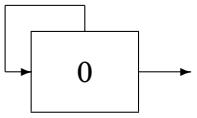
Berlekamp využil štruktúru matice syndrómov a navrhol efektívny algoritmus na riešenie sústavy (10.14). Massey preformuloval pôvodné Berlekampove riešenie do zrozumiteľnejšej podoby, v ktorej ho budeme prezentovať aj my. Budeme postupovať nasledovne: najprv uvedieme princíp algoritmu, potom ilustrujeme na príklade aké problémy algoritmus rieši. Napokon popíšeme algoritmus formálne a dokážeme jeho korektnosť a optimálnosť.

Ak zrušíme obmedzenie na j, rekurentný vzťah (10.17) popisuje nekonečnú postupnosť $S_1, S_2, \ldots$ generovanú posuvným registrom s lineárnou spätnou väzbou (linear feedback shift register, LFSR), ktorá je zadaná polynómom lokátorov chýb $\Lambda(x)$, obr. 10.1. Na to, aby sme určili polynóm lokátorov chýb $\Lambda(x)$, potrebujeme nájsť najkratší LFSR, ktorý generuje postupnosť $S_1, S_2, \ldots, S_{2\nu}$.



Obr. 10.1: LFSR so spätnou väzbou zadanou $\Lambda(x)$

Podstata Berlekampovho-Masseyovho algoritmu spočíva v iteratívnej konštrukcii registra (LFSR) generujúceho postupnosť hodnôt syndrómov $S_1, S_2, \ldots, S_{2\nu}; S_j \in GF(q)$. Ak sú známe registre $R^{(j)}$ generujúce počiatočné podpostupnosti $S_1, S_2, \ldots, S_j; \quad j = 1, \ldots,$



Obr. 10.2: LFSR $R^{(1)}$

$r-1$, tak register $R^{(r)}$ generujúci podpostupnosť $S_1, S_2, \ldots, S_{r-1}, S_r$ sa buď zhoduje s registrom $R^{(r-1)}$, alebo sa dá zostrojiť úpravou registra $R^{(r-1)}$ pomocou niektorého z predchádzajúcich registrov. Modifikácia registra $R^{(r-1)}$ musí zaručiť, aby nový register, $R^{(r)}$ spĺňal nasledujúce dve požiadavky:

1. aby správne generoval prvých r členov postupnosti,
2. aby sa dĺžka registra $R^{(r)}$ zväčšila oproti registru $R^{(r-1)}$ minimálne.

Tento postup budeme opakovať dovtedy, kým nezostrojíme register $R^{(2\nu)}$ minimálnej dĺžky, ktorý generuje postupnosť $S_1, S_2, \ldots, S_{2\nu}$.

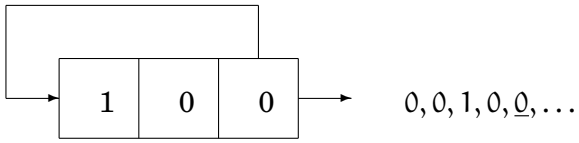
Skôr ako formálne popíšeme Berlekamp-Masseyov algoritmus a dokážeme jeho korektnosť, ilustrujeme jeho činnosť na príklade. Kvôli jednoduchosti budeme pracovať s poľom $GF(2)$.

Príklad. Nech je daná postupnosť $0,0,1,0,1,1,0,1,0,1$. Nájdeme register $R^{(10)}$, ktorý ju bude generovať. Register $R^{(1)}$ dĺžky 1 so spätnou väzbou $1+x$ a počiatočným stavom 0, obr. 10.2 generuje postupnosť $\{0\}_{k\geq 1}$. (Ten istý register by z počiatočného stavu 1 generoval postupnosť $\{1\}_{k\geq 1}$.) To znamená, že $R^{(1)}$ generuje správne prvé dva členy postupnosti, ale chybne generuje tretí ($R^{(2)} = R^{(1)}$). Musíme zväčšiť dĺžku registra $R^{(2)}$a prípadne modifikovať jeho spätnú väzbu tak, aby sme zostrojili register $R^{(3)}$. Najprv ukážeme, že žiaden register dĺžky 2 nemôže generovať podpostupnosť 001. Predpokladajme opak. Nech je R LFSR dĺžky 2 so spätnou väzbou $\Lambda(x) = 1 + \lambda_1 x + \lambda_2 x^2$. Aby R správne generoval prvé dva členy postupnosti, jeho počiatočný stav musí byť 00, ale potom tretí prvok postupnosti má hodnotu $0\lambda_1 + 0\lambda_2$ a to je pri ľubovoľnom výbere hodnôt λ_1, λ_2 vždy 0. To znamená, že register $R^{(3)}$ musí mať dĺžku aspoň 3 a na to, aby úspešne generoval postupnosť $0,0,1$, musí mať počiatočný stav (zapísané sprava doľava) 100. Spätná väzba v prvých troch taktoch neovplyvní výstup registra $R^{(3)}$, a teda pre ľubovoľný polynóm spätnej väzby

$$\Lambda(x) = 1 + \lambda_1 x + \lambda_2 x^2 + \lambda_3 x^3$$

bude register $R^{(3)}$ s počiatočným stavom 100 generovať postupnosť 001.[5] Vyberieme $R^{(3)}$ so spätnou väzbou zadanou polynómom $1 + x^3$. Tento register bude správne generovať aj štvrtý prvok postupnosti, 0, ale v piatom takte dôjde ku chybe a je potrebná korekcia spätnej väzby a možno aj predĺženie registra. Ukážeme, čo znamená modifikácia spätnej väzby registra. Register $R^{(3)}$, obr. 10.3 generuje postupnosť $0,0,1,0,0,1,0,0,1,\ldots$. Jedným z možných riešení je predĺžiť register na dĺžku 5 a nastaviť ako jeho počiatočný

[5] Ešte raz pripomenieme, že stav registra sa zapisuje sprava doľava a postupnosť ktorú generuje v opačnom poradí, t.j. zľava doprava.



Obr. 10.3: LFSR $R^{(3)}$

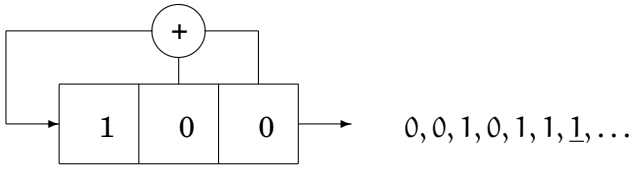
stav postupnosť, ktorú má generovať. Tento register by (bez ohľadu na to, ako by sme zvolili spätnú väzbu) generoval požadovanú postupnosť. Skúsime teraz nájsť kratší register, generujúci postupnosť $0,0,1,0,1$. Výstup registra $R^{(4)} = R^{(3)}$ spĺňa nasledujúci rekurentný vzťah:

$$S_{j+3} = S_j, \quad j = 1, 2, \ldots.$$

V prvých 3 taktoch sa na výstupe registra $R^{(4)}$ objavujú jednotlivé bity jeho počiatočného stavu, v 4. takte je výstupom (správna) hodnota, ktorá bola vypočítaná v 1. takte ($S_4 = S_1$), problematická hodnota 0, ktorá sa na výstupe registra objaví v 5. takte, bola vypočítaná v 2. takte. Ak nejako zmeníme spätnú väzbu registra a zachováme jeho dĺžku a počiatočný stav, výstup registra v prvých troch taktoch sa nezmení ale môže sa zmeniť aj výstup v 4. a 5. takte (výstup registra v ďalších taktoch nás v tomto momente nezaujíma). K spätnej väzbe registra môžeme pridať členy x, x^2, x^3 alebo nejakú ich kombináciu. Potrebujeme dosiahnuť, aby príspevok pridaných členov spätnej väzby bol v prvých 2 taktoch $0,1$. Pozrieme sa na to, aké hodnoty budú generovať možné členy spätnej väzby v prvých 2 taktoch, ak bol počiatočný stav registra $R^{(4)}$ 001.

$$\begin{array}{cc} x & 1,0 \\ x^2 & 0,1 \\ x^3 & 0,0 \end{array}$$

Požadovanú postupnosť nachádzame v druhom riadku; ak pridáme do polynómu spätnej väzby člen x^2, hodnota generovaná spätnou väzbou v prvých dvoch taktoch bude $S_1 + S_2, S_2 + S_3$, resp. $0,1$ čo zabezpečí správne generovanie 4. a 5. člena postupnosti. Výsledný register $R^{(5)}$ je na obrázku 10.4. Register $R^{(5)}$ správne generuje aj 6. prvok postupnosti,



Obr. 10.4: LFSR $R^{(5)}$

ale chyba nastáva v 7. takte. Opäť je potrebná korekcia spätnej väzby. Problém však je v tom, že žiaden LFSR dĺžky 3 postupnosť 0010110 nedokáže generovať. Ak by existoval register R dĺžky 3 s polynómom spätnej väzby

$$\Lambda(x) = 1 + \lambda_1 x + \lambda_2 x^2 + \lambda_3 x^3$$

a počiatočným stavom $1,0,0$ a generoval správne 4., 5., 6. a 7, prvok postupnosti, jeho

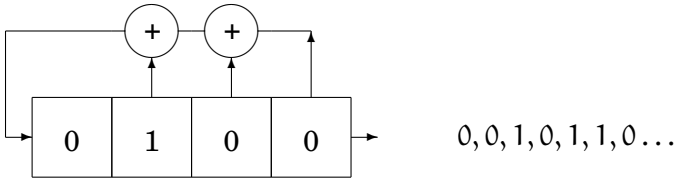
spätná väzba by musela spĺňať nasledujúce podmienky

$$\begin{aligned} \lambda_1 &= 0, \\ \lambda_2 &= 1, \\ \lambda_1 + \lambda_3 &= 1, \\ \lambda_1 + \lambda_2 &= 0 \end{aligned}$$

Z prvej, druhej a štvrtej podmienky vyplýva spor. Skúsme teraz predĺžiť register o 1; nech je R LFSR dĺžky 4, so spätnou väzbou $1+x^2+x^3$ a počiatočným stavom 0100. Register R generuje postupnosť $0,0,1,0,1,1,1,\ldots$; chyba, ktorá sa objavila na výstupe v 7. takte vznikla v 3. takte. Aby sme ju opravili, potrebujeme k spätnej väzbe pridať členy, ktoré by z počiatočného stavu 0100 generovali postupnosť $0,0,1$. V nasledujúcej tabuľke sú uvedené postupnosti generované potenciálnymi členmi spätnej väzby:

$$\begin{array}{cc} x & 0,1,1 \\ x^2 & 1,0,1 \\ x^3 & 0,1,0 \\ x^4 & 0,0,1 \end{array}$$

Do úvahy prichádza buď člen x^4, alebo kombinácia $x + x^3$ - obe riešenia zabezpečujú korekciu 7. generovaného prvku postupnosti. Vyberieme „opticky“ jednoduchšie riešenie a ako polynóm spätnej väzby registra $R^{(5)}$ vyberieme $\Lambda(x) = 1 + x^2 + x^3 + x^4$, obr. 10.5.



Obr. 10.5: LFSR $R^{(7)}$

Register $R^{(7)}$ sa v 8. takte dopúšťa chyby. Chybná hodnota bola vytvorená v 4. takte. Do spätnej väzby registra $R^{(7)}$ potrebujeme doplniť členy, ktoré by z počiatočného stavu 0100 generovali postupnosť $0,0,0,1$. Príspevky potenciálnych prvkov spätnej väzby uvádzame v nasledujúcej tabuľke

$$\begin{array}{cc} x & 0,1,1,0 \\ x^2 & 1,0,1,1 \\ x^3 & 0,1,0,1 \\ x^4 & 0,0,1,0 \end{array}$$

Požadovanú korekciu $0,0,0,1$ generuje kombinácia $x + x^3 + x^4$. Register $R^{(8)}$ má dĺžku 4, spätnú väzbu $\Lambda(x) = 1 + x + x^2$ a z počiatočného stavu 0100 generuje postupnosť $0,0,1,0,1,1,0,1$. Chyba však nastáva už v nasledujúcom 9. takte. Žiaden register dĺžky 4 nebude generovať postupnosť $0,0,1,0,1,1,0,1,0$. Zoberieme register R dĺžky 5 so spätnou väzbou $\Lambda(x) = 1+x+x^2$, ktorý bude z počiatočného stavu 10100 generovať postupnosť $0,0,1,0,1,1,0,1,\underline{1}$. Hodnota, ktorá sa na výstupe registra R objavila v 9. takte, bola generovaná v 4. takte. Na jej korigovanie potrebujeme do spätnej väzby pridať členy, ktoré

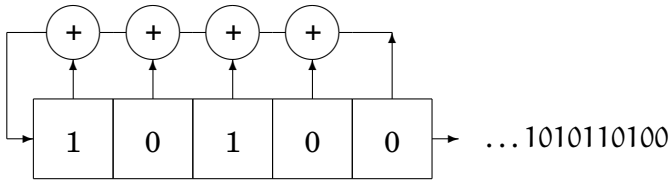
by generovali postupnosť $0,0,0,1$. Pomocou nasledujúcej tabuľky

$$\begin{array}{cc} x & 1,1,0,1 \\ x^2 & 0,1,1,0 \\ x^3 & 1,0,1,1 \\ x^4 & 0,1,0,1 \\ x^5 & 0,0,1,0 \end{array}$$

nájdeme potrebnú korekciu: $x^2 + x^4 + x^5$. Register $R^{(9)}$ dĺžky 5 so spätnou väzbou $\Lambda(x) = 1 + x + x^4 + x^5$ generuje prvých 9 prvkov postupnosti, ale chyba nastáva v poslednom, 10. prvku postupnosti. Chybná hodnota bola vytvorená v 5. kroku, a to znamená, že potrebujeme korigovať spätnú väzbu registra $R^{(9)}$tak, aby korekcia generovala postupnosť $0,0,0,0,1$. Z tabuľky

$$\begin{array}{cc} x & 1,1,0,1,0 \\ x^2 & 0,1,1,0,1 \\ x^3 & 1,0,1,1,0 \\ x^4 & 0,1,0,1,1 \\ x^5 & 0,0,1,0,1 \end{array}$$

vyplýva, že požadovaná korekcia pozostáva z členov $x + x^2 + x^3$. Register (obr. 10.6) má dĺžku 5, spätnú väzbu $\Lambda(x) = 1+x+x^2+x^3+x^4+x^5$ a z počiatočného stavu 10100 generuje postupnosť $0,0,1,0,1,1,0,1,0,1$.



Obr. 10.6: LFSR $R^{(10)}$

Ako sme videli v predchádzajúcom príklade, najväčším problémom pri konštrukcii registra bolo jednak rozhodnúť, či je potrebné a ak, tak o koľko predĺžiť register a nájsť vhodnú modifikáciu spätnej väzby. V príklade sme tento problém zakaždým nejako vyriešili, ale jednak to nebolo systematické riešenie a jednak sme nemali záruku, či získané riešenie (LFSR) bolo optimálne. Berlekampov-Masseyov algoritmus umožňuje jednoznačne stanoviť či treba, a ak áno, tak ako je potrebné v jednotlivých taktoch modifikovať aktuálnu spätnú väzbu registra, aby sme našli optimálne riešenie. Zavedieme niekoľko formálnych označení a popíšeme kľúčový krok Berlekampovho-Masseyovho algoritmu presnejšie. LFSR R dĺžky L so spätnou väzbou zadanou polynómom $\Lambda(x)$ budeme označovať $R : (L, \Lambda(x))$ a skutočnosť, že LFSR R generuje postupnosť $S_1, \ldots, S_k$ označíme $R : (L, \Lambda(x)) \overset{gen}{\hookrightarrow} S_1, \ldots, S_k$ Nech je daná postupnosť $S_1, \ldots$ prvkov poľa $GF(q)$. Predpokladáme, že pre $k = 1, \ldots, r-1$ sú už známe registre (LFSR)

$$\begin{aligned} R^{(1)} : (L_1, \Lambda^{(1)}(x)) &\overset{gen}{\leftrightarrow} S_1 \\ R^{(2)} : (L_2, \Lambda^{(2)}(x)) &\overset{gen}{\leftrightarrow} S_1, S_2 \\ &\dots \\ R^{(r-1)} : (L_{r-1}, \Lambda^{(r-1)}(x)) &\overset{gen}{\leftrightarrow} S_1, S_2, \dots S_{r-1} \end{aligned}$$

a pomocou nich zostrojíme register

$$R^{(r)} : (L_r, \Lambda^{(r)}(x)) \overset{gen}{\leftrightarrow} S_1, S_2, \dots, S_r.$$

Necháme najprv register $R^{(r-1)}$ vypočítať ešte jeden (r-tý) prvok. Jeho hodnota bude

$$S'_r = -\sum_{i=1}^{r-1} \lambda_i^{(r-1)} S_{r-i}.$$

Teraz porovnáme vypočítanú hodnotu S'_r so skutočnou hodnotou S_r. Rozdiel medzi týmito hodnotami označíme symbolom Δ_r:

$$\Delta_r = S_r - S'_r = S_r + \sum_{i=1}^{r-1} \lambda_i^{(r-1)} S_{r-i} = \sum_{i=0}^{r-1} \lambda_i^{(r-1)} S_{r-i},$$

pri poslednej úprave sme využili to, že $\lambda_0 = 1$. Ak $\Delta_r = 0$, register $R^{(r-1)}$ správne generoval aj r-tý prvok postupnosti, a teda

$$R^{(r)} = R^{(r-1)} : (L_{r-1}, \Lambda^{(r-1)}(x)) \overset{gen}{\leftrightarrow} S_1, S_2, \dots S_r.$$

Ak $\Delta_r \neq 0$, musíme spraviť korekciu spätnej väzby registra $R^{(r-1)}$ a prípadne zväčšiť jeho dĺžku. Nová spätná väzba je zadaná polynómom

$$\Lambda^{(r)}(x) = \Lambda^{(r-1)}(x) + A \cdot x^l \cdot \Lambda^{(m-1)}(x). \tag{10.18}$$

V definícii polynómu $\Lambda^{(r)}(x)$ v (10.18) vystupuje niekoľko záhadných parametrov, ktorých význam a hodnoty zakrátko určíme. Zatiaľ prezradíme, že $A \in GF(q)$, l je prirodzené číslo a $\Lambda^{(m-1)}(x)$ je jeden z polynómov spätnej väzby, ktorý bol zostrojený pri predchádzajúcich iteráciách. Vypočítame rozdiel medzi hodnotou S_r a hodnotou r-tého prvku generovaného registrom so spätnou väzbou zadanou polynómom (10.18):

$$\Delta'_r = \sum_{i=0}^{r-1} \lambda^{(r)} S_{r-i} = \sum_{i=0}^{r-1} \lambda^{(r-1)} S_{r-i} + A \cdot \sum_{i=0}^{r-1} \lambda^{(m-1)} S_{r-i-l} \tag{10.19}$$

Teraz potrebuje zvoliť parametre m, A, l tak, aby $\Delta'_r = 0$. Vyberieme $m < r$ také, že $\Delta_m \neq 0$. Položíme $l = r - m$ a nakoniec $A = -\Delta_r/\Delta_m$. Upravíme 10.19 a dostávame

$$\Delta'_r = \Delta_r - \frac{\Delta_r}{\Delta_m} \cdot \Delta_m = 0.$$

Zostrojený register $R^{(r)} : (L_r, \Lambda^{(r)}(x))$ je teda generátorom postupnosti $S_1, S_2, \ldots, S_r$. Ostala ešte otvorená jedna otázka, a to výber m, na ktorý sme zatiaľ nekládli žiadne podmienky. Ako dokážeme neskôr, ak vyberieme najväčšie m také, že $r > m$ a $L_m > L_{m-1}$, dostávame generátor $R^{(r)}$ minimálnej dĺžky.

Teraz, keď už máme dostatočnú predstavu o činnosti Berlekamp-Masseyovho algoritmu, môžeme ho popísať formálne a dokázať jeho korektnosť a optimálnosť. Formulácia vety a jej dôkazu je s malými modifikáciami prebratá z [2].

Veta 10.5.1. *(Berlekampova-Masseyova) Nech je daná postupnosť $S_1, \ldots, S_{2t}$ prvkov poľa $GF(q)$, nech pre počiatočné podmienky $\lambda^{(0)}(x) = 1, B^{(0)}(x) = 1$ a $L_0 = 0$ pre $r = 1, \ldots, 2t$*

$$\Delta_r = \sum_{i=0}^{r-1} \lambda_i^{(r-1)} S_{r-i}; \tag{10.20}$$

$$L_r = \delta_r(r - L_{r-1}) + (1 - \delta_r)L_{r-1}; \tag{10.21}$$

$$\Lambda^{(r)}(x) = \Lambda^{(r-1)}(x) - \Delta_r \cdot x \cdot B^{(r-1)}(x); \tag{10.22}$$

$$B^{(r)}(x) = \Delta_r^{-1} \cdot \delta_r \cdot \Lambda^{(r-1)}(x) + (1 - \delta_r) \cdot x \cdot B^{(r-1)}(x); \tag{10.23}$$

pričom

$$\delta_r = \begin{cases} 1 & ak \Delta_r \neq 0 \ \& \ 2L_{r-1} \leq r - 1, \\ 0 & inák. \end{cases}$$

Potom $\Lambda^{(2t)}(x)$ je polynóm minimálneho stupňa, ktorého koeficienty spĺňajú nasledujúce podmienky

$$\lambda_0^{(2t)} = 1, \qquad S_r + \sum_{i=1}^{r-1} \lambda_i^{(2t)} S_{r-i}, \quad r = L_{2t} + 1, \ldots, 2t.$$

Formulácia vety je na prvý pohľad zložitá. V podstate však ide o iteratívnu konštrukciu polynómu spätnej väzby generátora postupnosti $S_1, \ldots, S_{2t}$ popísanú vzťahom (10.18). Veličina δ_r indikuje, kedy dochádza k predĺženiu registra a zložito vyzerajúci polynóm $B^{(r)}(x)$ vyjadruje aktuálnu korekciu polynómu spätnej väzby (ak je korekcia potrebná). Pripomíname, že $\Delta_r = 0$ len v tom prípade, keď $\delta_r = 0$. To nám umožňuje definovať hodnotu $\delta_r/\Delta_r = 0$ v prípade, keď $\delta_r = 0$.

Samotný dôkaz vety rozdelíme na dôkazy dvoch pomocných tvrdení. V prvej leme odhadneme zdola dĺžku L_r registra $R^{(r)}$ pomocou dĺžky L_{r-1} jeho bezprostredného predchodcu, registra $R^{(r-1)}$. V druhej leme zostrojíme pomocou Berlekampovho-Masseyovho algoritmu generátor postupnosti $S_1, \ldots, S_r$, a dokážeme, že jeho dĺžka L_r bude dosahovať dolný odhad z prvej lemy, čím zavŕšime dôkaz optimálnosti Berlekampovho-Masseyovho algoritmu.

Lema 8. *Nech $R^{(r-1)} : (L_{r-1}, \Lambda^{(r-1)}(x))$ je LFSR minimálnej dĺžky, ktorý generuje postupnosť $S_1, \ldots, S_{r-1}$ a nech $R^{(r)} : (L_r, \Lambda^{(r)}(x))$ je LFSR minimálnej dĺžky, ktorý generuje postupnosť $S_1, \ldots, S_r$, pričom $\Lambda^{(r-1)}(x)) \neq \Lambda^{(r)}(x))$. Potom*

$$L_r \geq \max\left(L_{r-1}, r - L_{r-1}\right). \tag{10.24}$$

Dôkaz. Musíme dokázať dve nerovnosti

1. $L_r \geq L_{r-1}$,
2. $L_r \geq r - L_{r-1}$.

Prvá nerovnosť je zrejmá, pretože ak register generuje nejakú postupnosť, tak potom generuje aj jej počiatočnú podpostupnosť. Ostáva dokázať druhú nerovnosť. Ak by

$$L_{r-1} \geq r,$$

potom by $L_r \geq r - r = 0$, čo je triviálne. Preto budeme predpokladať, že

$$L_{r-1} < r.$$

Druhú nerovnosť teraz dokážeme sporom; budeme predpokladať, že druhá nerovnosť neplatí; t.j. že

$$L_r < r - L_{r-1},$$

resp.

$$L_r \leq r - 1 - L_{r-1}.$$

Pozrieme sa teraz na postupnosť generovanú registrom $R^{(r-1)}$. Z predpokladov lemy vyplýva, že register $R^{(r-1)}$ generuje prvých $r-1$ prvkov postupnosti

$$S_j = -\sum_{i=1}^{L_{r-1}} \lambda_i^{(r-1)} S_{j-i}, \quad j = L_{r-1}+1, \ldots, r-1; \tag{10.25}$$

ale negeneruje r-tý:

$$S_r \neq -\sum_{i=1}^{L_{r-1}} \lambda_i^{(r-1)} S_{r-i}. \tag{10.26}$$

Register $R^{(r)}$ generuje prvých r členov postupnosti:

$$S_j = -\sum_{i=1}^{L_r} \lambda_i^{(r)} S_{j-i}, \quad j = L_r+1, \ldots, r.$$

Vyjadríme S_r

$$S_r = -\sum_{i=1}^{L_r} \lambda_i^{(r)} S_{r-i} \tag{10.27}$$

V sume (10.27) vystupujú členy $S_{r-L_r}, \ldots, S_{r-1}$. Ale podľa predpokladu

$$r - L_r \geq r - (r - 1 - L_{r-1}) = L_{r-1} + 1,$$

a to znamená, že každý z členov $S_{r-L_r}, \ldots, S_{r-1}$ môžeme vyjadriť pomocou (10.25). Dostávame

$$S_r = -\sum_{k=1}^{L_r} \lambda_k^{(r)} S_{r-k} = \sum_{k=1}^{L_r} \lambda_k^{(r)} \sum_{i=1}^{L_{r-1}} \lambda_i^{(r-1)} S_{r-k-i}.$$

Zmeníme poradie sumácie, upravíme vnútornú sumu a dostávame

$$S_r = \sum_{i=1}^{L_{r-1}} \lambda_i^{(r-1)} \sum_{k=1}^{L_r} \lambda_k^{(r)} S_{r-k-i} = -\sum_{i=1}^{L_{r-1}} \lambda_i^{(r-1)} S_{r-i}.$$

Dostali sme spor s (10.26), čím je tvrdenie lemy dokázané. □

Z predchádzajúcej lemy vyplýva, že ak pri konštrukcii LFSR generujúceho postupnosť $S_1, \ldots, S_r$ zostrojíme v r-tom kroku register dĺžky $L_r = \max(L_{r-1}, r - L_{r-1})$, $r = 1, 2, \ldots$, tak je tento register optimálny. Ostáva dokázať, že LFSR konštruované pomocou Berlekampovho-Masseyovho algoritmu túto podmienku spĺňajú.

Lema 9. *Nech*

$$R^{(j)} : (L_j, \Lambda^{(j)}(x)) \overset{gen}{\leftrightarrow} S_1, S_2, \ldots, S_j, \quad j = 1, 2 \ldots$$

je postupnosť registrov generujúcich počiatočné podpostupnosti postupnosti $\{S_k\}_{k\geq 1}$. *Ak v registri* $R^{(r)}$ *tejto postupnosti došlo k zmene spätnej väzby v porovnaní s predchádzajúcim registrom;*

$$\Lambda^{(r)}(x) \neq \Lambda^{(r-1)}(x),$$

tak pre dĺžku registra $R^{(r)}$ *platí*

$$L_r = \max(L_{r-1}, r - L_{r-1})$$

a ľubovoľný LFSR dĺžky L_r *generujúci postupnosť* $S_1, S_2, \ldots, S_r$ *je registrom-generátorom minimálnej dĺžky.*

LFSR $R^{(r)}$ *zostrojený podľa Berlekampovho-Masseyovho algoritmu je generátorom minimálnej dĺžky postupnosti* $S_1, S_2, \ldots, S_r$.

Dôkaz. Z predchádzajúcej lemy vyplýva, že dĺžka registra $R^{(r)}$ nemôže byť menšia než $\max(L_{r-1}, r - L_{r-1})$. Ak sa preto podarí ukázať, že $L_r = \max(L_{r-1}, r - L_{r-1})$, bude to znamenať, že register $R^{(r)}$ je generátorom (postupnosti $S_1, S_2, \ldots, S_r$) minimálnej dĺžky. Ukážeme, že pre $r = 1, 2, \ldots$ sa dá zostrojiť generátor minimálnej dĺžky. Dôkaz budeme viesť matematickou indukciou vzhľadom na dĺžku generovanej postupnosti.

Pre $k = 1$ tvrdenie platí, lebo $L_0 = 0, L_1 = 1$.

Predpokladajme, že pre $k = 1, \ldots, r-1$ sú už zostrojené registre

$$R^{(k)} : (L_k, \Lambda^{(jk)}(x)) \overset{gen}{\leftrightarrow} S_1, S_2, \ldots, S_k$$

a zakaždým, keď došlo k zmene spätnej väzby; t.j. ak $\Lambda^{(k)}(x) \neq \Lambda^{(k-1)}(x)$ platí

$$L_k = \max(L_{k-1}, k - L_{k-1}).$$

Teraz vyjadríme rozdiel medzi hodnotami, ktoré generuje register $R^{(r-1)}$ v jednotlivých taktoch a príslušnými členmi postupnosti $\{S_j\}_{j\geq 1}$.

$$S_j + \sum_{i=1}^{L_{r-1}} \lambda_i^{(r-1)} S_{j-i} = \sum_{i=0}^{L_{r-1}} \lambda_i^{(r-1)} S_{j-i} = \begin{cases} 0 & j = L_{r-1}+1, \ldots, r-1; \\ \Delta_r & j = r. \end{cases}$$

Register $R^{(r-1)}$ správne generuje prvých $r-1$ členov postupnosti $\{S_j\}_{j\geq 1}$. Ak $\Delta_r = 0$, tak $R^{(r-1)}$ správne generuje aj r-tý člen postupnosti a

$$L_r = L_{r-1}; \quad \Lambda^{(r)} = \Lambda^{(r-1)}.$$

Ak $\Delta_r =\neq 0$ musíme spraviť korekciu spätnej väzby a zostrojiť nový register. Posledná zmena dĺžky registra nastala v kroku m. Potom

$$S_j + \sum_{i=1}^{L_{m-1}} \lambda_i^{(m-1)} S_{j-i} = \begin{cases} 0 & j = L_{m-1}+1, \ldots, m-1 \\ \Delta_m \neq 0 & j = m. \end{cases}$$

Využijeme indukčný predpoklad a vyjadríme L_{r-1} pomocou L_{m-1}:

$$L_{r-1} = L_m = \max(L_{m-1}, m - L_{m-1}) = m - L_{m-1}.$$

Zostrojíme nový polynóm spätnej väzby

$$\Lambda^{(r)}(x) = \Lambda^{(r-1)}(x) - \frac{\Delta_r}{\Delta_m} \cdot x^{r-m} \Lambda^{(m-1)}(x).$$

Džka registra $R^{(r)}$ je $L_r = \deg \Lambda^{(r)}(x)$. Keďže

$$\deg \Lambda^{(r-1)}(x) \leq L_{r-1}$$

a

$$\deg(x^{r-m}\Lambda^{(m-1)}(x)) \leq r - m + L_{m-1},$$

pre dĺžku registra $R^{(r)}$ dostávame nasledujúci horný odhad

$$L_r \leq \max(L_{r-1}, r - m + L_{m-1}) \leq \max(L_{r-1}, r - L_{r-1}).$$

Ukážeme, že register $R^{(r)}$ generuje postupnosť $S_1, S_2, \ldots, S_r$. Vyjadríme rozdiel medzi hodnotami generovanými registrom $R^{(r)}$ v jednotlivých taktoch a príslušnými členmi postupnosti $S_1, S_2, \ldots, S_r$.

$$\begin{aligned} S_j - \left(-\sum_{i=1}^{L_r} \lambda_i^{(r)} S_{j-i} \right) &= S_j + \sum_{i=1}^{L_r} \lambda_i^{(r-1)} S_{j-i} - \frac{\Delta_r}{\Delta_m} \left[S_{j-r+m} + \sum_{i=1}^{L_{m-1}} \lambda_i^{(m-1)} S_{j-r+m-i} \right] = \\ &= \begin{cases} 0 & j = L_{r-1}+1, \ldots, r-1 \\ \Delta_r - \frac{\Delta_r}{\Delta_m} \cdot \Delta_m = 0 & j = r. \end{cases} \end{aligned}$$

Keďže register $R^{(r)}$ generuje postupnosť $S_1, S_2, \ldots, S_r$, podľa predchádzajúcej lemy

$$L_r \geq \max\left(L_{r-1}, r - L_{r-1}\right).$$

To znamená, že

$$L_r = \max\left(L_{r-1}, r - L_{r-1}\right)$$

a register $R^{(r)} : (L_r, \Lambda^{(r)}(x))$ je optimálny LFSR generujúci postupnosť $S_1, S_2, \ldots, S_r$; resp. $R^{(2t)} : (L_{2t}, \Lambda^{(2t)}(x))$ je optimálny LFSR generujúci postupnosť $S_1, S_2, \ldots, S_{2t}$. □

Vrátime sa k príkladu zo začiatku paragrafu a zostrojíme LFSR pomocou Berlekamp Masseovho algoritmu.

Príklad. Nech je daná binárna postupnosť $0, 0, 1, 0, 1, 1, 0, 1, 0, 1$. V tabuľke 10.2 sú uvedené hodnoty parametrov Berlekamp-Masseyovho algoritmu v jednotlivých krokoch výpočtu.

r	Δ_r	δ_r	L_r	$\Lambda^{(r)}(x)$	$B^{(r)}(x)$
0	0	0	0	1	1
1	0	0	0	1	x
2	0	0	0	1	x^2
3	1	1	3	$1+x^3$	1
4	0	0	3	$1+x^3$	x
5	1	0	3	$1+x^2+x^3$	x^2
6	0	0	3	$1+x^2+x^3$	x^3
7	1	1	4	$1+x^2+x^3+x^4$	$1+x^2+x^3$
8	1	0	4	$1+x+x^2$	$x+x^3+x^4$
9	1	1	5	$1+x+x^4+x^5$	$1+x^2+x^3$
10	1	0	5	$1+x^2+x^3+x^4+x^5$	$x+x^3+x^4$

Tabuľka 10.2: Berlekamp-Massey

10.6 Zvláštnosti dekódovania binárnych BCH kódov

10.7 Reedove-Solomonove kódy

Špeciálnym prípadom BCH kódov sú Reedove-Solomonove kódy, ktoré, ako zakrátko ukážeme, sú z istého hľadiska optimálne.

Definícia 10.7.1. *Reed-Solomonov kód (RS kód) je primitívny BCH kód dĺžky* $n = q - 1$ *nad poľom* $GF(q)$.

Keďže RS kódy tvoria podtriedu BCH kódov, RS kód opravujúci t chýb možno zadať generujúcim polynómom $g(x)$ s koreňmi $\alpha^{j_0+1}, \alpha^{j_0+2}, \ldots, \alpha^{j_0+2t}$, kde α je primitívny prvok poľa $GF(q)$. Kvôli zjednodušeniu položíme $j_0 = 0$ a nájdeme explicitné vyjadrenie generujúceho polynómu. Generujúci polynóm BCH kódu je definovaný ako najmenší spoločný násobok minimálnych polynómov svojich koreňov. Minimálnym polynómom prvku α^i poľa $GF(q)$ je polynóm $m_{\alpha^i}(x) = x - \alpha^i$ nad poľom $GF(q)$. To však znamená, že generujúci polynóm $g(x)$ bude súčinom minimálnych polynómov svojich koreňov:

$$g(x) = (x-\alpha)(x-\alpha^2)\ldots(x-\alpha^{2t}).$$

Určíme opravné schopnosti RS kódu. Konštrukčná vzdialenosť RS-kódu ako špeciálneho prípadu BCH kódu je $d \geq 2t+1$. Na druhej strane, $g(x)$ je kódovým slovom a jeho váha neprevyšuje $2t+1$ (túto hodnotu by polynóm $g(x)$ dosiahol v prípade, ak by boli všetky jeho koeficienty nenulové). Stupeň generujúceho polynómu cyklického kódu zodpovedá počtu kontrolných symbolov. To znamená, že

$$\deg(g(x)) = 2t = n - k,$$

resp. $n - k + 1 = 2t + 1$. RS kód je zároveň lineárnym kódom a pre lineárne kódy platí Singeltonova hranica $d \leq n - k + 1$. To znamená, že minimálna vzdialenosť RS kódu je $d = n - k + 1$. Tým sme dokázali nasledujúcu vetu.

Veta 10.7.1. *RS kód má minimálnu vzdialenosť* $n - k + 1$ *a je kódom s maximálnou (minimálnou) vzdialenosťou.*

Z uvedenej vety vyplýva, že pre dané n, k neexistuje kód, ktorého minimálna vzdialenosť by bola väčšia ako minimálna vzdialenosť RS kódu. Túto skutočnosť však nemožno preceňovať, pretože často potrebujeme zostrojiť kód s parametrami n', k' pre ktoré neexistuje RS kód, ale existujú iné dostatočne dobré samoopravné kódy.

RS kódy nie sú binárne, ale keď zvolíme $q = 2^m$, symboly kódovej abecedy možno priamo nahrádzať binárnymi vektormi dĺžky m. Uvedieme niekoľko príkladov RS kódov a potom sa budeme zaoberať ich praktickým použitím pri opravovaní zhlukov chýb.

Príklad. Začneme krátkym RS kódom. Nech $q = 8$. Pole $GF(2^3)$ zostrojíme faktorizáciou okruhu polynómov $GF(2)$ ireducibilným polynómom $x^3 + x + 1$. Prvky poľa $GF(2^3)$ vyjadrené pomocou mocnín primitívneho prvku α a lineárnych kombinácií mocnín primitívneho prvku α sú uvedené v nasledujúcej tabuľke:

α^i	α^2	α^1	α^0	α^i	α^2	α^1	α^0
α^0	0	0	1	α^4	1	1	0
α^1	0	1	0	α^5	1	1	1
α^2	1	0	0	α^6	1	0	1
α^3	0	1	1	α^7	0	0	1

1. RS kód opravujúci chybu váhy 1 je zadaný generujúcim polynómom s koreňmi α, α^2;
$$g(x) = (x - \alpha)(x - \alpha^2) = x^2 + \alpha^4 \cdot x + \alpha^3.$$
2. RS kód opravujúci chybu váhy 2 je zadaný generujúcim polynómom s koreňmi $\alpha, \alpha^2, \alpha^3, \alpha^4$;
$$g(x) = (x - \alpha)(x - \alpha^2)(x - \alpha^3)(x - \alpha^4) = x^4 + \alpha^3 x^3 + x^2 + \alpha x + \alpha^3.$$
Keďže generujúci polynóm má stupeň 4, informačný polynóm bude mať stupeň 2. Nech $i(x) = \alpha^4 x^2 + \alpha^2 x + \alpha^6$. Kódovým polynómom bude
$$i(x) \cdot g(x) = x^6\alpha^4 + x^5\alpha^6 + x^4\alpha^2 + x^3\alpha^5 + x^2\alpha^5 + x\alpha^4 + \alpha^2.$$
Ak vyjadríme kódové slovo v tvare vektora a nahradíme prvky poľa $GF(2^3)$ binárnymi vektormi dĺžky 3, dostávame binárny vektor dĺžky 21:
$$110\ 101\ 100\ 111\ 111\ 110\ 100.$$

Pozrieme sa ešte na RS kód dĺžky 15 nad poľom $GF(2^4)$, opravujúci 3 chyby. Tento kód bude zadaný generujúcim polynómom $g(x)$ s koreňmi $\alpha, \alpha^2, \ldots, \alpha^6$;
$$g(x) = (x - \alpha)(x - \alpha^2)(x - \alpha^3)(x - \alpha^4)(x - \alpha^5)(x - \alpha^6).$$
a bude opravovať chyby váhy ≤ 3. Pri binárnom zápise kódových slov dostávame z RS $(15, 9)$ kódu $(60, 36)$ kód. Ak by sme RS kód chceli použiť na kódovanie informácie prenášanej kanálom, na ktorý pôsobí biely šum, binárny zápis kódových slov je skôr nevýhodou. Ak totiž v kódovom slove (dĺžky 60) vznikne chyba váhy 4 alebo väčšej, zmení

pravdepodobne viac než 3 hexadecimálne symboly kódového slova, čo prevýši opravnú schopnosť daného RS kódu. V mnohých aplikáciách je však predpoklad o nezávislosti chýb neopodstatnený: ak dôjde k poškodeniu CD, elektrickému výboju alebo poruche, tieto udalosti ovplyvnia pravdepodobne niekoľko susedných znakov kódového slova. Takýmto chybám sa hovorí zhluky chýb (burst errors), a RS kódy sa dajú výhodne použiť práve na ochranu informácie prenášanej kanálom, v ktorom dochádza k zhlukom chýb.

Definícia 10.7.2. *(Cyklickým) zhlukom chýb dĺžky* t

Kapitola 11

Modifikácie samoopravných kódov

Niekedy sa stane, že nevieme nájsť dobrý samoopravný kód, ktorý vyhovuje presne naším potrebám, ale vieme o existencii dobrého samoopravného kódu, ktorého parametre sa od nami požadovaných hodnôt veľmi neodlišujú. Ponúka sa prirodzená otázka, či sa známy kód nedá upraviť tak, aby si zachoval svoje dobré vlastnosti a zároveň splnil naše požiadavky. Ako zakrátko uvidíme, metódy umožňujúce transformácie samoopravných kódov skutočne existujú. V tejto krátkej kapitole stručne popíšeme šesť základných metód úprav samoopravných kódov. Budeme pracovať prevažne s lineárnymi kódmi; nie však preto, že by sa dané metódy nedali použiť na úpravy nelineárnych kódov, ale preto, že v prípade lineárnych kódov majú modifikácie, ktoré budeme uvádzať, veľmi jednoduchú interpretáciu. Metódy modifikácie samoopravných kódov možno nájsť prakticky v ľubovoľnej monografii venovanej kódovaniu; čitateľovi pre podrobnejšie štúdium problematiky odporúčame vynikajúco spracovanú kapitolu v [7], resp. prednášky [**?**], z ktorých sme prebrali niektoré príklady.

Lineárny kód $\mathcal{C}$ má tri základné parametre: dĺžku n, počet informačných symbolov (= dimenzia lineárneho podpriestoru) k a počet kontrolných symbolov $n - k$. Podstata základných metód modifikácie spočíva v tom, že sa jeden z troch uvedených parametrov fixuje a ostatné dva sa menia. Celkovo teda máme 6 základných možností ktoré kvôli prehľadnosti uvádzame v nasledujúcej tabuľke (keďže slovenská terminológia zatiaľ neexistuje, budeme sa pridržiavať aglického označenia):

No.	Názov metódy		n	k	$n-k$
1.	augmenting	zväčšenie	-	$\uparrow$	$\downarrow$
2.	expurgating	zmenšenie	-	$\downarrow$	$\uparrow$
3.	extending	rozšírenie	$\uparrow$	-	$\uparrow$
4.	puncturing	zúženie	$\downarrow$	-	$\downarrow$
5.	lengthening	predĺženie	$\uparrow$	$\uparrow$	-
6.	shortening	skrátenie	$\downarrow$	$\downarrow$	-

Pripomenieme, že lineárne (n, k) kódy je možné zadať generujúcou maticou G typu $k \times n$ alebo kontrolnou maticou H typu $n - k \times n$. Vo všeobecnosti sa pri zmenách počtov

informačných a kontrolných symbolov a nemennej dĺžke kódových slov pridávajú alebo vynechávajú riadky v generujúcej/kontrolnej matici; pri zmenách dĺžky kódového slova sa pridávajú alebo vynechávajú stĺpce generujúcej/kontrolnej matice. Pozrieme sa teraz na jednotlivé metódy modifikácie kódov podrobnejšie.

Metódy Lengthening a Shortening

Pri predlžovaní a skracovaní kódu sa mení celková dĺžka kódu a počet informačných symbolov, zostáva zachovaný počet kontrolných symbolov. Pri predlžovaní (lengthening) zväčšujeme dĺžku kódu (a počet informačných symbolov) a ku kódu pridávame nové kódové slová; pri skracovaní (shortening) kódu naopak odstraňujeme informačné symboly a tak z pôvodného kódu odstraňujeme kódové slová.

Predlžovanie lineárnych kódov sa robí pridávaním rovnakého počtu riadkov a stĺpcov ku generujúcej matici. Štandardný spôsob predlžovania lineárnych kódov spočíva v pridaní nulového stĺpca ku generujúcej matici a následne v doplnení nulového riadka, ktorý však má nenulovú hodnotu v poslednom (pridanom) stĺpci. Skracovanie lineárneho kódu predstavuje inverzný postup k predlžovaniu kódu. Najprv upravíme generujúcu maticu kódu tak, aby obsahovala jeden riadok, ktorý má nulové hodnoty na všetkých miestach s výnimkou posledného. Odstránením tohto riadku a posledného stĺpca z generujúcej matice dostaneme generujúcu maticu skráteného kódu. Iná možnosť skracovania a predlžovania kódu spočíva vo vynechávaní resp. pridávaní vybraných stĺcov kontrolnej matice.

Príklad (skrátenie kódu). Uvažujme Hammingov (15,11) binárny kód zadaný kontrolnou maticou

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}. \tag{11.1}$$

Odstránením stĺpcov 12,13 a 14 dostávame (12,8) kód opravujúci 1 chybu, zadaný kontrolnou maticou

$$H' = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

Zostrojený kód má dve prednosti: informačný vektor má dĺžku 8 a keďže sme z pôvodnej kontrolnej matice odstránili stĺpce maximálnej váhy, zjednodušili sa kontrolné sumy (namiesto pôvodných 8 členov obsahujú teraz 5 a 6 členov). Na druhej strane, počet kódových slov klesol z pôvodných 2048 na 256.

Príklad (predĺženie kódu). $(n-1, k)$ Reedov-Solomonov kód predĺžime pridaním dvoch nových informačných symbolov a zostrojíme Reedov-Solomonov kód s parametrami $(n+1, k+2)$. Aby sme zaistili opravovanie chýb v doplnených symboloch kódových slov, nové informačné symboly musíme zaradiť aj do kontrolných súm. To dosiahneme doplnením

kontrolnej matice o také dva nové stĺpce, aby sme neznížili rang kontrolnej matice:

$$H = \begin{bmatrix} 1 & \alpha & \alpha^2 & \dots & \alpha^{n-2} \\ 1 & \alpha^2 & \alpha^4 & \dots & \alpha^{2(n-2)} \\ \vdots & & & & \vdots \\ 1 & \alpha^{2t} & \alpha^{4t} & \dots & \alpha^{2t(n-2)} \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 0 & 1 & \alpha & \alpha^2 & \dots & \alpha^{n-2} \\ 0 & 0 & 1 & \alpha^2 & \alpha^4 & \dots & \alpha^{2(n-2)} \\ \vdots & & & & & & \vdots \\ 0 & 1 & 1 & \alpha^{2t} & \alpha^{4t} & \dots & \alpha^{2t(n-2)} \end{bmatrix}.$$

Metódy Augmenting a Expurgating

Pri týchto modifikáciách si kód zachováva nemennú dĺžku kódového slova, ale mení sa počet kontrolných a informačných symbolov. Keď sa zväčšuje počet informačných symbolov (augmenting), rastie aj počet kódových slov, ale zmenšuje sa počet kontrolných symbolov a môže sa zmenšiť (a spravidla sa aj zmenšuje) minimálna vzdialenosť a tým aj opravná schopnosť kódu. Naopak, pri zväčšovaní počtu kontrolných symbolov (expurgating) v slove sa zmenšuje počet informačných symbolov, klesá počet kódových slov a môže sa zväčšiť minimálna vzdialenosť kódu.

V prípade lineárneho kódu sa metódy Augmenting a Expurgating realizujú jednoducho: ak chceme kód zväčšiť, pridáme do generujúcej matice nový riadok (často sa pridáva jednotkový riadok); ak potrebujeme kód zmenšiť, z generujúcej matice odstraňujeme riadky, resp. pridávame riadky do kontrolnej matice kódu. Metódu Augmenting sme použili pri konštrukcii Reedových-Mullerových kódov; ku generujúcej matici kódu $\mathcal{R}(r-1,m)$ sme pridali maticu G_r a dostali generujúcu maticu kódu $\mathcal{R}(r,m)$:

$$G = \begin{bmatrix} G_0 \\ G_1 \\ \vdots \\ G_{r-1} \end{bmatrix} \Rightarrow \begin{bmatrix} G_0 \\ G_1 \\ \vdots \\ G_{r-1} \\ G_r \end{bmatrix}.$$

Pôvodný $\mathcal{R}(r-1,m)$ kód mal dĺžku 2^m a mal $\sum_{0\le j<r}\binom{m}{j}$ informačných symbolov, dĺžka kódových slov kódu $\mathcal{R}(r,m)$ zostala zachovaná, ale počet informačných symbolov v slove vzrástol v porovnaní s kódom $\mathcal{R}(r-1,m)$ o $\binom{m}{r}$.

Príklad (metóda Expurgation). Metódu expurgation sme použili pri konštrukcii cyklického $(15,7)$ kódu opravujúceho 2 chyby z Hammingovho $(15,11)$ kódu v úvode 8 kapitoly. Kontrolnú maticu H (11.1) rozšírime o 4 riadky a dostávame kontrolnú maticu typu 8×15,

$$H' = \left[\begin{array}{ccccccccccccccc} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ \hline 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \end{array}\right]$$

ktorú kvôli stručnosti zapíšeme ako maticu nad poľom $GF(2^4)$:

$$H = \begin{bmatrix} \alpha^0 & \alpha^1 & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 & \alpha^7 & \alpha^8 & \alpha^9 & \alpha^{10} & \alpha^{11} & \alpha^{12} & \alpha^{13} & \alpha^{14} \\ \alpha^0 & \alpha^3 & \alpha^6 & \alpha^9 & \alpha^{12} & \alpha^{15} & \alpha^{18} & \alpha^{21} & \alpha^{24} & \alpha^{27} & \alpha^{30} & \alpha^{33} & \alpha^{36} & \alpha^{39} & \alpha^{42} \end{bmatrix}.$$

Dôležitá metóda zmenšenia kódu sa zakladá na výbere podmnožiny kódovej abecedy a vylúčení všetkých slov pôvodného kódu, ktoré obsahujú aj iné symboly. Presnejšie, nech je dané konečné pole F a nech je $\mathcal{C}$ kód nad poľom F. Nech je G podpole poľa F. Podkódom kódu $\mathcal{C}$ nad podpoľom G nazveme množinu kódových slov kódu $\mathcal{C}$, ktorých všetky zložky patria do podpoľa G. Takýto podkód „zdedil“ mnohé dobré vlastnosti pôvodného kódu a mohol nadobudnúť aj ďalšie dobré vlastnosti. Túto metódu sme využili pri konštrukcii Reedových-Solomonových kódov.

Metódy Extending a Puncturing

V obidvoch prípadoch si kód zachováva počet informačných symbolov (a zároveň aj počet kódových slov). Mení sa dĺžka kódového slova a počet kontrolných symbolov.

Keď sa kód rozširuje (extension), pridávajú sa nové kontrolné symboly. Pri redukcii sa naopak počet kontrolných symbolov v kódových slovách zmenšuje. Pri znižovaní počtu kontrolných symbolov môže dôjsť k zníženiu minimálnej vzdialenosti a pri pridávaní kontrolných symbolov sa minimálna vzdialenosť kódu môže zväčšiť. Rozšírenie lineárneho kódu sa dá realizovať doplnením nového stĺpca generujúcej matice, naopak redukcia lineárneho kódu sa realizuje odstránením stĺpca generujúcej matice. Lineárny (n,k)-kód $\mathcal{C}$ možno rozšíriť tak, že sa ku každému kódovému slovu $v = (v_1, \ldots, v_n)$ pridá nový kontrolný symbol v_{n+1}, ktorý je lineárnou kombináciou symbolov pôvodného slova;

$$v_{n+1} = a_1 v_1 + a_2 v_2 + \cdots + a_n v_n.$$

Vektor $a_1, \ldots, a_n$ by nemal patriť do duálneho kódu $\mathcal{C}^{\perp}$, pretože v tom prípade by $v_{n+1} = 0$. Požiadavky na výber vektora $\mathbf{a}$ sa analyzujú v [7], my sa nimi zaoberať nebudeme. Pri rozširovaní lineárnych kódov sa najčastejšie ku kódovým slovám pridáva paritný symbol; t.j. kódové slovo $\mathbf{v} = (v_1, \ldots, v_n, v_{n+1})$ potom spĺňa nasledujúcu rovnosť

$$\sum_{i=1}^{n+1} v_1 = 0.$$

(V tomto prípade vektor $\mathbf{a} = (-1, \ldots, -1)$.)

Kapitola 12

Prínos kódovania

Keď na prenos správ používame komunikačný kanál, na ktorý vplýva nejaký zdroj šumu, hrozí že šum modifikuje prenášanú správu do takej miery, že príjemca nebude schopný zrekonštruovať pôvodne odvysielanú správu. Existujú principiálne dva spôsoby, ako eliminovať alebo aspoň znížiť riziko modifikácie, ktoré by spôsobilo nesprávnu interpretáciu prijatej správy;

- zvýšenie odolnosti signálov voči šumu a
- použitie samoopravných kódov.

Doteraz sme sa zaoberali takmer výlučne riešeniami založenými na použití samoopravných kódov. Sú však situácie, kedy samoopravný kód nedokáže zaistiť potrebnú úroveň spoľahlivosti komunikácie, inokedy je jednoduchšie zvýšiť robustnosť vysielaného signálu (napríklad zvýšiť hlasitosť vysielania v hlučnom prostredí) alebo je výhodné kombinovať technické (použitie robustnejších signálov na prenos správ) a matematické riešenie (samoopravný kód). Predpokladajme, že našou úlohou je pre daný komunikačný systém nájsť optimálne riešenie, ktoré by zaistilo požadovanú úroveň bezpečnosti komunikácie (vyjadrenú napríklad pravdepodobnosťou nesprávneho dekódovania alebo interpretácie prijatého slova), pričom máme možnosť ovplyvňovať robustnosť signálov prostredníctvom zmien výkonu vysielača a použiť nejaký samoopravný kód. Aby sme dokázali posúdiť príspevok použitia samoopravného kódu k zmene[1] spoľahlivosti komunikácie, budeme v tejto kapitole študovať vzťah medzi energiou signálu pripadajúceho na jeden symbol prenášanej správy (kódovanej alebo nekódovanej) a pravdepodobnosťou nesprávneho dekódovania prijatého slova.[2]

Na zjednodušenie výkladu prijmeme nasledujúce predpoklady:

- kódová abeceda je binárna,
- vysielač má ohraničený (konštantný výkon) a na prenos jedného bitu pripadá energia E_b joulov,

[1] zýšeniu alebo zníženiu

[2] Výklad vychádza z práce [15] a využíva matematický aparát teórie signálov z práce [5]

- na prenos správ sa používajú amplitúdovo modulované signály; 1 je reprezentovaná signálom S hodnotou $\sqrt{E_b}$ a 0 je reprezentovaná signálom s hodnotou $-\sqrt{E_b}$.

Poznámka. Signál reprezentujúci 1 môžeme popísať funkciou $S(t)$ (impulzom) s obdĺžnikovým priebehom:

$$S(t) = \begin{cases} \sqrt{E_b} & -1/2 < t < 1/2, \\ 0 & \text{ináč.} \end{cases}$$

Množstvo energie pripadajúcej na impulz je [5] tak, ako sme predpokladali E_b:

$$\int_{-\infty}^{\infty} S(t)^2 dt = E_b \int_{-1/2}^{1/2} dt = E_b.$$

Predpokladáme ďalej, že na prenosový kanál pôsobí zdroj bieleho Gaussovského aditívneho šumu[3]. Výsledkom jeho pôsobenia je náhodný signál $N(t)$ (noise), ktorý sa pripočítava k signálu prenášajúcemu správu. Aby sme sa nemuseli zaoberať časovou alebo spektrálnou reprezentáciou signálu $S(t)$ a šumu $N(t)$, signálu (aj šumu) na intervale jednotkovej dĺžky priradíme číselné hodnoty S resp. N (dané napríklad priemernou hodnotou príslušného signálu na intervale, alebo hodnotou ktorú nadobúda v strede intervalu). Šum potom môžeme chápať ako náhodnú premennú s normálnym rozdelením so strednou hodnotou 0 a disperziou σ^2. Hodnoty signálu a šumu sa sčítavajú a výsledná hodnota $S + N$ sa interpetuje (demodulátorom) nasledovne

$$S + N \to \begin{cases} 1 & S + N > 0, \\ 0 & \text{ináč.} \end{cases}$$

Je zrejmé, že demodulátor chybne interpretuje prijatý signál vtedy, ak sa v dôsledku šumu mení znamienko signálu, t.j. pôvodne kladná hodnota signálu reprezentujúceho 1 sa pripočítaním hodnoty šumu zmenila na zápornú, ktorá reprezentuje 0 a *vice versa*. To znamená, že, napríklad pri prenose 0 hodnota šumu musela prevýšiť $\sqrt{E_b}$ a pri prenose 1 dôjde ku chybe, ak $N < -\sqrt{E_b}$. Využijeme to, že náhodná premenná N popisujúca pôsobenie šumu má normálne rozdelenie a vyčíslime pravdepodobnosť toho, že šum nadobúda vyššie uvedené hodnoty.

$$P(N > \sqrt{E_b}) = \frac{1}{\sqrt{2\pi}\sigma} \int_{\sqrt{E_b}}^{\infty} e^{-t^2/2\sigma^2} dt = 1 - \Phi\left(\sqrt{\frac{E_b}{\sigma^2}}\right), \tag{12.1}$$

$$P(N < -\sqrt{E_b}) = \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{\sqrt{E_b}} e^{-t^2/2\sigma^2} dt = \Phi\left(-\sqrt{\frac{E_b}{\sigma^2}}\right), \tag{12.2}$$

kde $\Phi(x)$ je distribučná funkcia normálneho rozdelenia. Zo symetrie hustoty normálneho rozdelenia vyplýva, že $P(N > \sqrt{E_b}) = P(N < -\sqrt{E_b}) = 1 - \Phi\left(\sqrt{\frac{E_b}{\sigma^2}}\right)$. Zaujímavý je argument funkcie Φ; pomer E_b/σ^2 sa nazýva *pomer signálu k šumu (signal to noise ratio),*

[3] presnú charakteristiku takého šumu možno nájsť v [5]

SRN a ako sa ukáže neskôr, pomocou neho sa dá kvantitatívne vyjadriť odolnosť signálu voči šumu.

Vyjadrili sme pravdepodobnosť vzniku chyby pri prenose jedného bitu. Aby sme mohli určiť efekt kódovania, predpokladajme, že nekódovaná správa pozostáva zo slov dĺžky k bitov. Na kódovanie použijeme (n, k) kód, takže pri prenesení n-bitového kódového slova prenesieme v skutočnosti len k informačných bitov; ak by sme nezmenili modulátor ani vysielač, prenosová rýchlosť by po kódovaní nadobudla hodnotu $R = k/n$. Aby sme toto zníženie prenosovej rýchlosti kompenzovali, musíme za jednotku času preniesť $1/R = n/k$ bitov kódového slova. Keďže vysielač má konštantný výkon, znamená to, že po takejto úprave bude množstvo energie pripadajúce na jednotkový signál (signál reprezentujúci jeden bit prenášanej správy) RE_b, t.j. menšie. Tým sa zníži aj robustnosť signálu a pravdepodobnosť toho, že pri prenose jedného bitu dôjde vplyvom šumu k chybe stúpne na

$$p'_e = 1 - \Phi\left(\sqrt{\frac{RE_b}{\sigma^2}}\right). \tag{12.3}$$

Na prvý pohľad to nevyzerá dobre, pretože zjavne $p_e < p'_e$. Efekt samoopravného kódovania sa však prejaví až pri dekódovaní slov. Porovnáme pravdepodobnosti chybného dekódovania/interpretácie prijatého slova v prípade keď na jeho zápis pri prenose nebol/bol použitý samoopravný kód. V prvom prípade prenášame slovo dĺžky k a interpretujeme ho správne, ak pri prenose nevznikla žiadna chyba. V druhom prípade prenášame slovo dĺžky n a chyba dekódovania nastane vtedy, ak počas prenosu v slove vzniklo viac chýb, ako dekóder dokázal opraviť. Ilustrujeme tieto skutočnosti na dvoch príkladoch.

Príklad. [15] Kozmická sonda Mariner, vypustená v roku 1969 používala na kódovanie obrazov $(32, 6)$-kód opravujúci chyby váhy 7 a menšej. Predpokladajme, že tolerovateľná pravdepodobnosť nesprávneho dekódovania slova je 10^{-4}. Bez použitia samoopravných kódov bude mať prenášané slovo dĺžku 6 bitov a ak je pravdepodobnosť chyby v jednom bite p_e, pravdepodobnosť chyby v slove bude

$$P_e = 1 - (1 - p_e)^6 \leq 10^{-4} \tag{12.4}$$

Z nerovnosti 12.4 vyjadríme p_e: $1-10^{-4} \leq (1-p_e)^6$; potom $(1-10^{-4})^{1/6} \leq 1-p_e$ a napokon $p_e \leq 1-(1-10^{-4})^{1/6} = 1.66674 \cdot 10^{-5}$. Aby sme dosiahli požadovanú spoľahlivosť prenosu, hľadáme takú hodnotu pomeru signál/šum E_b/σ^2, aby

$$p_e \approx \frac{10^{-4}}{6} = 1 - \Phi\left(\sqrt{\frac{E_b}{\sigma^2}}\right); \qquad \text{t.j.} \quad \Phi\left(\sqrt{\frac{E_b}{\sigma^2}}\right) = 1 - \frac{10^{-4}}{6} = 0.999983;$$

Platí $\Phi(4.15) = .9999833763$, a teda $E_b/\sigma^2 \approx 4.15^2 = 17.22$. Pravdepodobnosť chyby v jednom bite sa pri použití 32 bitového samoopravného kódu výrazne zvýši (lebo $R = 6/32$) a dosiahne

$$p'_e = 1 - \Phi(\sqrt{17.22 * 6/32}) = 1 - \Phi(1.79687) = 0.0361757483.$$

Kód však má opravnú schopnosť 7 a to znamená, že ak v prenášanom slove nevznikne viac ako 7 chýb, dekóder dokáže prijaté slovo dekódovať správne. Pravdepodobnosť

vzniku aspoň 8 chýb v slove je

$$P'_E = \sum_{i=8}^{32} \binom{32}{i} (p'_e)^i (1-p'_e)^{32-i} = 0.00001413616427,$$

čo je podstatne lepšie ako požadovaná hodnota 10^{-4}. (Pri použití „mäkkých“ metód demodulácie a dekódovania sa dá dosiahnuť dokonca až hodnota $P'_E = 2 \cdot 10^{-11}$ [15]).

Uvedieme ešte jeden príklad. Binárny BCH (15,5) kód opravujúci tri chyby nám dobre poslúžil napríklad pri výklade metód dekódovania BCH kódov. Ale pomohol by nám zvýšiť spoľahlivosť prenosu údajov z Marinera?

Príklad 12.1. *Aby sme mohli porovnať „kvality“ BCH (15,5) kódu s kódom použitým na kódovanie údajov Marinera, zachovajme úroveň spoľahlivosti* $P_E = 10^{-4}$. *Pri prenose bez použitia samoopravného kódu budeme prenášať 5 bitové slová a pravdepodobnosť chybného prenosu 1 bitu dosiahne* $p_e = 2 \cdot 10^{-5} = 1 - \Phi(4.107479655)$ *a pomer signál/šum bude* $E_b/\sigma^2 = 16.87138910$. *Pri použití uvedeného binárneho (15,5) kódu sa prenosová rýchlosť zníži na tretinu (*$R = 5/15$*). Vyjadríme pravdepodobnosti* p'_e *a* P'_E*:*

$$p'_e = 1 - \Phi(2,37145448) = 0.0088591145$$

a

$$P'_E = 0.7776185512 \cdot 10^{-5}.$$

Pravdepodobnosť chybného dekódovania prijatého slova bude teda pri použití binárneho BCH (15,5) kódu 12 krát nižšia v porovnaní s pravdepodobnosťou chybnej interpretácie prijatej pätice (nekódovaných) informačných bitov.

Napriek optimistickým výsledkom predchádzajúcich príkladov, použitie samoopravných kódov nemusí zaistiť zvýšenie spoľahlivosti komunikácie; resp. presnejšie povedané, každý samoopravný kód má svoj rozsah použitia a jeho použitie mimo tohto rozsahu môže dokonca znížiť[4] spoľahlivosť komunikácie.

Čo sa dá spraviť v prípade, ak pomocou samoopravného kódu zvýšime spoľahlivosť prenosu nad stanovenú hodnotu (vyjadrenú pravdepodobnosťou nesprávneho dekódovania prijatého slova, P_E)? Ak nám stanovená hranica spoľahlivosti prenosu postačuje, môžeme znížiť množstvo energie použité na prenos jedného bitu

- znížením výkonu vysielača,
- zvýšením počtu bitov prenesených za časovú jednotku.

Keďže v obidvoch prípadoch ide o stanovenia množstva energie/signál prenášajúci 1 bit, ktoré postačuje na dosiahnutie želanej hodnoty P_E, stačí sa zaoberať prvým prípadom. V prípade sondy Mariner na dosiahnutie požadovanej úrovne $P_E = 10^{-4}$ postačuje hodnota SRN 14.83. Použitie samoopravného kódu v tomto prípade malo pozitívny efekt, ktorý sa

[4] použitie samoopravného kódu možno kombinovať aj so zmenou iných parametrov komunikačného systému, čo môže ovplyvniť rozsah použiteľnosti kódu

kód	t	coding gain
Hammingov$(7,4)$	1	3.066127697
Hammingov$(15,11)$	1	2.824701569
Hammingov$(31,26)$	1	2.615760124
BCH$(15,5)$	3	5.750883407
Golayov$(23,12)$	3	5.447658460
RM$(2,4)*$	1	-2.398093804
Mariner$(32,6)$	7	0.6489199602

Tabuľka 12.1: Efektívnosť vybraných samoopravných kódov

dokonca dá vyjadriť aj kvantitatívne - pomerom SRN (bez kódovania):SRN (s kódovaním); v našom prípade je hodnota pomeru 1.1611. Na dosiahnutie požadovanej spoľahlivosti prenosu by teda bolo možné znížiť výkon vysielača asi o 15%. „Pomer pomerov“, ktorý sme pred chvíľou zaviedli sa skutočne používa na meranie efektu kódovania.

Pomer SRN (nekódovaného)/SRN (kódovaného) prenosu pre rovnakú pravdepodobnosť chyby dekódovania prijatého slova[5] sa nazýva *prínos kódovania (coding gain)* .

Prínos kódovania sa vyjadruje v decibeloch (dB) a vypočítava sa ako desaťnásobok hodnoty dekadického logaritmu z pomeru SRN (nekódovaného)/SRN (kódovaného). V prípade kódovania prenosu Marinera je prínos kódovania 0.65 dB. Pre BCH $(15,5)$ kód je prínos kódovania ešte väčší (5.75 dB). V tabuľke 12.1 sú pre porovnanie uvedené niektoré zo samoopravných kódov, ktorými sme sa doteraz zaoberali a hodnoty code gain pre $P_E = 10^{-4}$. Pri Reedovom-Mullerovom kóde je minimálna vzdialenosť 4, ale opravná schopnosť kódu je len 1.

Na záver tejto kapitoly ešte pripomenieme, že to, že prínos kódovania je kladný[6] nemusí samo o sebe byť dostatočným dôvodom na použitie samoopravného kódu. Samoopravné kódy si o. i. vyžadujú (softvérovo alebo hardvérovo realizovaný) kóder a dekóder, čo sa prejavuje tak na cene komunikačného systému, ako aj na čase spracovania správ. Ten istý efekt, ako zavedenie samoopravných kódov môžeme dosiahnuť použitím iného komunikačného kanála alebo iných signálov, ktoré sú odolnejšie voči šumu. Pri hľadaní optimálneho riešenia na zaistenie požadovanej úrovne spoľahlivosti komunikácie je potrebné zvažovať tak technické možnosti a ohraničenia, ako aj možnosti samoopravných kódov.

[5] ak nebolo použité kódovanie, tak chyba dekódovania znamená, že nastala chyba v ľubovoľnom bite prenášaného slova

[6] SRN (nekódovaného)>SRN (kódovaného) prenosu

Kapitola 13

Shannonova teoréma

Samoopravné kódy umožňujú zvýšiť pravdepodobnosť správneho dekódovania prijatej informácie. Cena, ktorú za to treba zaplatiť, je doplnenie informačných symbolov v kódovom slove o kontrolné symboly. V malých kódoch, ktoré sme konštruovali v predchádzajúcich kapitolách bol podiel kontrolných symbolov na celkovej dĺžke slova pomerne značný. Zdalo by sa, že opravná schopnosť a rýchlosť prenosu sú v nepriamej úmere; t.j. že podiel počtu informačných symbolov na celkovej dĺžke kódového slova sa bude nutne zmenšovať so vzrastajúcou minimálnou vzdialenosťou kódu. Prekvapujúci výsledok prináša klasická Shannonova veta. Ukazuje sa, že pre mnohé prenosové kanály je možné

- prenášať informáciu rýchlosťou blízkou prenosovej rýchlosti kanála;
- pravdepodobnosť chybného dekódovania prijatej informácie možno stlačiť pod ľubovoľne malú, dopredu zadanú hodnotu.

V tejto časti vyslovíme a dokážeme Shannonovu vetu. Najprv uvedieme a vysvetlíme predpoklady Shannonovej vety a pripomenieme pojmy, ktoré budeme pri jej dôkaze potrebovať.

Budeme predpokladať, že prenosový kanál, ktorý sa používa na prenos správ (kódovanej informácie) je binárny symetrický kanál bez pamäte; t.j. na kódovanie správ sa používa binárna abeceda a prenosy jednotlivých binárnych symbolov sa uskutočňujú s nasledujúcimi pravdepodobnosťami:

$$\begin{aligned} p &: \quad 0 \to 0, \quad 1 \to 1 \\ 1-p=q &: \quad 0 \to 1, \quad 1 \to 0 \end{aligned}$$

Budeme pracovať s kódom $C = \{\mathbf{x}_1, \ldots, \mathbf{x}_M\}$, ktorý má M kódových slov dĺžky n. Predpokladáme, že kódové slová kódu C sa na výstupe zdroja informácie vyskytujú rovnako často; t.j., že

$$\forall \mathbf{x}_i \in C; \quad P(\mathbf{x}_i) = 1/M. \tag{13.1}$$

Predpokladajme, že bolo odvysielané kódové slovo $\mathbf{x}_i$ a prijaté slovo $\mathbf{y}$. Prijaté slovo $\mathbf{y}$ sa dekóduje ako kódové slovo $\mathbf{x}_j$ také, že

$$P(\mathbf{y}|\mathbf{x}_j) = \max_{\mathbf{x}_k \in C}\{P(\mathbf{y}|\mathbf{x}_k)\}. \tag{13.2}$$

Prijaté slovo sa teda dekóduje na základe maximálnej pravdepodobnosti; ak vznikne chyba malej váhy, prijaté slovo sa dekóduje správne, ak vznikne chyba väčšej váhy, ako je opravná schopnosť použitého kódu, prijaté slovo sa dekóduje nesprávne. Odhadneme pravdepodobnosť nesprávneho dekódovania prijatého slova. Pravdepodobnosť nesprávneho dekódovania odvysielaného (a šumom pri prenose modifikovaného kódového slova) $\mathbf{x}_i$ označíme symbolom P_i. Pravdepodobnosť nesprávneho dekódovania prijatého slova pri použití kódu C je

$$P_C = \frac{1}{M}\sum_{i=1}^{M} P_i.$$

Uvažujme teraz množinu $\mathcal{C}$, množinu všetkých binárnych kódov, ktoré majú M slov dĺžky n a zavedieme:

$$P^*(M, n, q) = \min_{C \in \mathcal{C}}\{P_C\}.$$

Pre konkrétny kód C je dĺžka kódu nemenná. V Shannonovej konštrukcii budeme potrebovať parameter n meniť. Predpokladáme, že v závislosti od n sa bude meniť aj počet kódových slov; t.j., $M = M(n)$. Kvôli stručnosti budeme namiesto $M(n)$ písať len M_n;

$$P^*(M_n, n, q) = \min_{C \in \mathcal{C}}\{P_C\}. \tag{13.3}$$

Pripomenieme ešte, že Hammingova vzdialenosť dvoch vektorov $\mathbf{u}, \mathbf{v}$, ktorú označujeme symbolom $d(\mathbf{u}, \mathbf{v})$ je daná počtom zložiek, v ktorých sa oba vektory líšia. Guľou so stredom v bode (vektore) $\mathbf{x}$ a polomerom r označíme množinu vektorov

$$B_r(\mathbf{x}) = \{y \in \{0,1\}^n; \quad d(x, y) \leq r\}. \tag{13.4}$$

Napokon, rýchlosť prenosu R kódu C definujeme ako podiel $R = |C|/2^n$.

Veta 13.0.2 (Shannonova teoréma)**.** *Nech je prenosová rýchlosť kódu* $0 \leq R \leq 1 + p\lg p + q\lg q$ *a mohutnosť kódu* $M = 2^{\lfloor R\cdot n\rfloor}$*, potom*

$$P^*(M_n, n, q) \to 0$$

pre $n \to \infty$.

Dôkaz. Pravdepodobnosť toho, že v kódovom slove $\mathbf{x}_i$ vznikne pri prenose k chýb, je $\binom{n}{k} \cdot p^{n-k}(1-p)^k$. Táto hodnota závisí len od parametrov p, n a nezávisí od toho, aké slovo bolo odvysielané. Zavedieme náhodnú premennú $\xi_{p,n}$, vyjadrujúcu počty chýb v odvysielaných slovách;

$$P(\xi_{p,n} = k) = \binom{n}{k} \cdot p^{n-k}(1-p)^k.$$

Je zrejmé, že náhodná premenná $\xi_{p,n}$ má binomické rozdelenie pravdepodobností so strednou hodnotou $E(\xi_{p,n}) = nq$ a disperziou $Var(\xi_{p,n}) = npq$. Pre ľubovoľnú náhodnú

premennú ζ so strednou hodnotou $E(\zeta)$, disperziou $Var(\zeta)$ a ľubovoľné reálne číslo $\alpha > 0$ platí (Čebyševova nerovnosť)

$$P(|\zeta - E(\zeta)| > \alpha) < \frac{Var(\zeta)}{\alpha^2}.$$

Položíme v Čebyševovej nerovnosti $\alpha = \sqrt{2npq/\varepsilon}$, $\zeta = \xi_{p,n}$ a upravíme:

$$P(|\xi_{p,n} - nq| > \sqrt{2npq/\varepsilon}) < \frac{npq}{2npq/\varepsilon} = \frac{\varepsilon}{2}. \tag{13.5}$$

Z nerovnosti (13.5) vyplýva, že pre počet chýb v prenášanom slove s pravdepodobnosťou $1 - \varepsilon/2$ platí

$$nq - \sqrt{2npq/\varepsilon} < \xi_{p,n} < nq + \sqrt{2npq/\varepsilon}. \tag{13.6}$$

Nás zaujímajú chyby väčšej váhy, ktoré vedú k nesprávnemu dekódovaniu. Z (13.5) vyplýva, že

$$P(\xi_{p,n} > nq + \sqrt{2npq/\varepsilon}) < \frac{\varepsilon}{2}.$$

Všimneme si, že ak $n \to \infty$, $0 < p, q < 1$ a $\varepsilon > 0$ sú konštanty,

$$\xi_{p,n} = nq + O(\sqrt(n)),$$

t.j. dá sa očakávať, že v prenášanom slove vznikne asi nq chýb. Položíme

$$t = \lfloor nq + \sqrt{2npq/\varepsilon} \rfloor \tag{13.7}$$

a zistíme, aká je pravdepodobnosť toho, že pri prenose vznikne v slove viac chýb ako t.

Zavedieme dve funkcie. Prvá je indikátor, ktorý pre dvojicu vektorov $\mathbf{u}, \mathbf{v}$ určí, či sú vo vzdialenosti $\leq t$ alebo nie:

$$f(\mathbf{u}, \mathbf{v}) = \begin{cases} 0 & d(\mathbf{u}, \mathbf{v}) > t \\ 1 & d(\mathbf{u}, \mathbf{v}) \leq t \end{cases} \tag{13.8}$$

Pre ľubovoľné kódové slovo $\mathbf{x}_i \in C$ a ľubovoľný binárny vektor dĺžky n, $\mathbf{y} \in \{0,1\}^n$ zavedieme nasledujúcu funkciu:

$$g_i(\mathbf{y}) = 1 - f(\mathbf{y}, \mathbf{x}_i) + \sum_{j \neq i} f(\mathbf{y}, \mathbf{x}_j). \tag{13.9}$$

Posledná funkcia si zaslúži podrobnejšie vysvetlenie: ak je $\mathbf{x}_i$ odvysielané a $\mathbf{y}$ prijaté slovo, tak môžu nastať tri možnosti:

1. $\mathbf{x}_i$ je jediné kódové slovo také, že $d(\mathbf{x}_i, \mathbf{y}) \leq t$. Potom $f(\mathbf{x}_i, \mathbf{y}) = 1$, ale $\forall j \neq i \quad f(\mathbf{x}_j, \mathbf{y}) = 0$. To znamená, že v tomto prípade

$$g_i(\mathbf{y}) = 1 - 1 + 0 = 0$$

2. $d(\mathbf{x}_i, \mathbf{y}) \leq t$ a existuje ešte aspoň jedno kódové slovo $\mathbf{x}_j; j \neq i$ také, že $d(\mathbf{x}_j, \mathbf{y}) \leq t$. Potom
$$g_i(\mathbf{y}) = 1 - 1 + 1 + \sum_{k \neq i,j} f(\mathbf{y}, \mathbf{x}_k) = 1 + \sum_{k \neq i,j} f(\mathbf{y}, \mathbf{x}_k).$$
Suma $\sum_{k \neq i,j} f(\mathbf{y}, \mathbf{x}_k)$ je nezáporná, a teda v tomto prípade
$$g_i(\mathbf{y}) \geq 1.$$

3. $d(\mathbf{x}_i, \mathbf{y}) > t$; t.j. $f(\mathbf{x}_i, \mathbf{y}) = 0$ a
$$g_i(\mathbf{y}) = 1 - 0 + \sum_{j \neq i} f(\mathbf{y}, \mathbf{x}_j) = 1 + \sum_{j \neq i} f(\mathbf{y}, \mathbf{x}_j).$$
Suma $\sum_{j \neq i} f(\mathbf{y}, \mathbf{x}_j)$ je nezáporná, a teda aj v tomto prípade $g_i(\mathbf{y}) \geq 1$.

Funkcia $g_i(\mathbf{y})$ teda nadobúda hodnotu 0 v prípade, ak prijaté slovo $\mathbf{y}$ leží vo vzdialenosti menšej alebo rovnej t od kódového slova $\mathbf{x}_i$ a leží vo vzdialenosti väčšej ako t od ostatných kódových slov. V opačnom prípade nadobúda hodnotu väčšiu alebo rovnú jednej.

Pristúpime teraz k dôkazu Shannonovej vety. Vyberieme kódové slová kódu C náhodne a nezávisle na sebe:
$$C = \{\mathbf{x}_1, \ldots, \mathbf{x}_M\}.$$
Predpokladáme, že sme odvysielali kódové slovo $\mathbf{x}_i$ a prijali slovo $\mathbf{y}$. Prijaté slovo dekódujeme nasledovne:

- Ak existuje jediné kódové slovo $\mathbf{x}_j$ také, že $d(\mathbf{x}_j, \mathbf{y}) \leq t$ dekódujeme prijaté slovo ako $\mathbf{x}_j$. Ak $i = j$, dekódovali sme správne, v opačnom prípade sme sa dopustili chyby dekódovania.
- Ak existuje niekoľko kódových slov ležiacich vo vzdialenosti $\leq t$ od prijatého slova $\mathbf{y}$; resp. neexistuje žiadne kódové slovo, ktoré by ležalo vo vzdialenosti $\leq t$ od prijatého slova $\mathbf{y}$[1]) vyhlásime chybu.

Aká je pravdepodobnosť chybného dekódovania?

$$\begin{aligned} P_i &\leq \sum_{\mathbf{y} \in \{0,1\}^n} P(\mathbf{y}|\mathbf{x}_i) \cdot g_i(\mathbf{y}) = \sum_{\mathbf{y} \in \{0,1\}^n} P(\mathbf{y}|\mathbf{x}_i) \cdot \left[1 - f(\mathbf{y}, \mathbf{x}_i) + \sum_{j \neq i} f(\mathbf{y}, \mathbf{x}_j)\right] = \\ &= \sum_{\mathbf{y} \in \{0,1\}^n} P(\mathbf{y}|\mathbf{x}_i) \cdot [1 - f(\mathbf{y}, \mathbf{x}_i)] + \sum_{\mathbf{y} \in \{0,1\}^n} \sum_{j \neq i} P(\mathbf{y}|\mathbf{x}_i) \cdot f(\mathbf{y}, \mathbf{x}_j) \end{aligned} \tag{13.10}$$

Prvá suma z výrazu (13.10) sa dá zapísať takto

$$\sum_{\mathbf{y} \in \{0,1\}^n} P(\mathbf{y}|\mathbf{x}_i) \cdot [d(\mathbf{x}_i, \mathbf{y}) > t] = P(d(\mathbf{x}_i, \mathbf{y}) > t). \tag{13.11}$$

[1] kód C sme vytvárali náhodne

Výraz (13.11) vyjadruje pravdepodobnosť toho, že prijaté slovo nepatrí do gule $B_t(\mathbf{x}_i)$ a dá sa odhadnúť zhora nasledovne

$$P(d(\mathbf{x}_i, \mathbf{y}) > t) < \frac{1}{2}\varepsilon,$$

pretože to, že $d(\mathbf{x}_i, \mathbf{y}) > t$ znamená, že pri prenose (slova $\mathbf{x}_i$) vznikla chyba váhy väčšej ako t. Spočítame pravdepodobnosť nesprávneho dekódovania prijatého slova, ak bolo odvysielané niektoré kódové slovo.

$$P_C = \frac{1}{M} \cdot \sum_{i=1}^{M} P_i \leq \frac{1}{2}\varepsilon + \frac{1}{M} \sum_{i=1}^{M} \sum_{\mathbf{y}\in\{0,1\}^n} \sum_{j\neq i} P(\mathbf{y}|\mathbf{x}_i) \cdot f(\mathbf{y}, \mathbf{x}_j). \tag{13.12}$$

Hlavná myšlienka dôkazu spočíva v tom, že

$$P^*(M_n, n, q) \leq E(P_C),$$

t.j. minimálnu hodnotu chyby dekódovania možno odhadnúť zhora strednou hodnotou chyby dekódovania, ktorú berieme cez všetky možné kódy C:

$$P^*(M_n, n, q) \leq E\left[\frac{1}{2}\varepsilon + \frac{1}{M} \sum_{i=1}^{M} \sum_{\mathbf{y}\in\{0,1\}^n} \sum_{j\neq i} P(\mathbf{y}|\mathbf{x}_i) \cdot f(\mathbf{y}, \mathbf{x}_j)\right]. \tag{13.13}$$

Upravíme (13.13) pomocou nasledujúcich pravidiel pre strednú hodnotu:

- pre ľubovoľnú konštantu c, $E(c) = c$;
- pre ľubovoľné dve náhodné premenné φ, ψ, $E(\varphi + \psi) = E(\varphi) + E(\psi)$.

Dostávame

$$P^*(M_n, n, q) \leq \frac{1}{2}\varepsilon + \frac{1}{M} \sum_{i=1}^{M} \sum_{\mathbf{y}\in\{0,1\}^n} \sum_{j\neq i} E\left[P(\mathbf{y}|\mathbf{x}_i) \cdot f(\mathbf{y}, \mathbf{x}_j)\right]. \tag{13.14}$$

Zmeníme poradie sumácie v (13.14) tak, aby sme najprv počítali sumu vzhľadom na $\mathbf{y} \in \{0,1\}^n$:

$$P^*(M_n, n, q) \leq \frac{1}{2}\varepsilon + \frac{1}{M} \sum_{i=1}^{M} \sum_{j\neq i} \sum_{\mathbf{y}\in\{0,1\}^n} E\left[P(\mathbf{y}|\mathbf{x}_i) \cdot f(\mathbf{y}, \mathbf{x}_j)\right]. \tag{13.15}$$

Vnútornú sumu z (13.15)

$$\sum_{\mathbf{y}\in\{0,1\}^n} E\left[P(\mathbf{y}|\mathbf{x}_i) \cdot f(\mathbf{y}, \mathbf{x}_j)\right] \tag{13.16}$$

rozdelíme na dve časti

$$\sum_{\mathbf{y}\in\{0,1\}^n} E\left[P(\mathbf{y}|\mathbf{x}_i) \cdot f(\mathbf{y}, \mathbf{x}_j)\right] = \sum_{\mathbf{y}\in\{0,1\}^n \cap B_t(\mathbf{x_j})} E\left[P(\mathbf{y}|\mathbf{x}_i) \cdot f(\mathbf{y}, \mathbf{x}_j)\right] + \sum_{\mathbf{y}\notin\{0,1\}^n \cap B_t(\mathbf{x_j})} E\left[P(\mathbf{y}|\mathbf{x}_i) \cdot f(\mathbf{y}, \mathbf{x}_j)\right]$$

Keďže $f(\mathbf{y}, \mathbf{x}_j) = 0$ pre $\mathbf{y} \notin \{0,1\}^n \cap B_t(\mathbf{x_j})$, a $f(\mathbf{y}, \mathbf{x}_j) = 1$ pre $\mathbf{y} \in \{0,1\}^n \cap B_t(\mathbf{x_j})$

$$\sum_{\mathbf{y}\in\{0,1\}^n} E\left[P(\mathbf{y}|\mathbf{x}_i) \cdot f(\mathbf{y}, \mathbf{x}_j)\right] = \sum_{\mathbf{y}\in\{0,1\}^n \cap B_t(\mathbf{x_j})} E\left[P(\mathbf{y}|\mathbf{x}_i)\right].$$

Podmienenú pravdepodobnosť $P(\mathbf{y}|\mathbf{x}_i)$ vyjadríme nasledovne:

$$P(\mathbf{y}|\mathbf{x}_i) = \frac{P(\mathbf{y}, \mathbf{x}_i)}{P(\mathbf{x}_i)}.$$

Ale $P(\mathbf{x}_i) = 1/M$, a teda

$$\sum_{\mathbf{y}\in\{0,1\}^n} E\left[P(\mathbf{y}|\mathbf{x}_i) \cdot f(\mathbf{y}, \mathbf{x}_j)\right] = M \cdot \sum_{\mathbf{y}\in\{0,1\}^n \cap B_t(\mathbf{x_j})} E(P(\mathbf{y}, \mathbf{x}_i)). \tag{13.17}$$

Dosadíme (13.17) do vzťahu (13.15):

$$P^*(M_n, n, q) \leq \frac{1}{2}\varepsilon + \sum_{i=1}^{M} \sum_{j\neq i} \sum_{\mathbf{y}\in\{0,1\}^n \cap B_t(\mathbf{x_j})} E(P(\mathbf{y}, \mathbf{x}_i)) \tag{13.18}$$

a upravíme trojitú sumu. Postupne dostávame

$$\begin{aligned}
\sum_{i=1}^{M} \sum_{\mathbf{y}\in\{0,1\}^n \cap B_t(\mathbf{x_j})} \sum_{j\neq i} E(P(\mathbf{y}, \mathbf{x}_i)) &= (M-1) \sum_{i=1}^{M} \sum_{\mathbf{y}\in\{0,1\}^n \cap B_t(\mathbf{x_j})} E(P(\mathbf{y}, \mathbf{x}_i)) = \\
(M-1) \sum_{\mathbf{y}\in\{0,1\}^n \cap B_t(\mathbf{x_j})} \sum_{i=1}^{M} E(P(\mathbf{y}, \mathbf{x}_i)) &= (M-1) \sum_{\mathbf{y}\in\{0,1\}^n \cap B_t(\mathbf{x_j})} E(P(\mathbf{y})) = \\
(M-1) \sum_{\mathbf{y}\in\{0,1\}^n \cap B_t(\mathbf{x_j})} E(2^{-n}) &= (M-1) \cdot \frac{|B_t|}{2^n}.
\end{aligned} \tag{13.19}$$

Pre pravdepodobnosť $P^*(M_n, n, q)$ sme zatiaľ z (13.15) a (13.19)odvodili nasledujúci horný odhad

$$P^*(M_n, n, q) \leq \frac{1}{2}\varepsilon + (M-1) \cdot \frac{|B_t|}{2^n}. \tag{13.20}$$

Upravíme (13.20) (najprv odčítame od oboch strán $\frac{1}{2}\varepsilon$, výsledok logaritmujeme a napokon obe strany vydelíme hodnotou n):

$$\begin{aligned}
P^*(M_n, n, q) - \frac{1}{2}\varepsilon &\leq (M-1) \cdot \frac{|B_t|}{2^n} \\
\lg(P^*(M_n, n, q) - \frac{1}{2}\varepsilon) &\leq \lg(M-1) + \lg(|B_t|) - n \\
\frac{\lg(P^*(M_n, n, q) - \frac{1}{2}\varepsilon)}{n} &\leq \frac{\lg M}{n} + \frac{\lg(|B_t|)}{n} - 1.
\end{aligned} \tag{13.21}$$

Ostáva odhadnúť výraz $\frac{\lg|B_t|}{n}$ z výrazu (13.21). Odhadneme najprv mohutnosť gule s polomerom t (v binárnom vektorovom priestore dimenzie n) zhora (pripomíname, že

$t < n/2$)

$$\begin{aligned} |B_t| &= \sum_{k=0}^{t} \binom{n}{k} \leq t\binom{n}{t} \leq \frac{1}{2}n \cdot \frac{n!}{t!(n-t)!} < \frac{1}{2}n \cdot \frac{n^n}{t^t(n-t)^{n-t}} = \\ &= \frac{1}{2}n \frac{1}{\left(\frac{t}{n}\right)^t \left(1-\frac{t}{n}\right)^{(n-t)}}. \end{aligned} \tag{13.22}$$

Teraz na základe odhadu (13.21) zostrojíme horný odhad pre $\frac{\lg |B_t|}{n}$:

$$\begin{aligned} \frac{\lg |B_t|}{n} &\leq \frac{1}{n}\left[\lg \frac{1}{2}n - t \lg \frac{t}{n} - (n-t)\lg\left(1-\frac{t}{n}\right)\right] = \\ &= -\frac{1}{n} + \frac{\lg n}{n} - \frac{t}{n}\lg\frac{t}{n} - \left(1-\frac{t}{n}\right)\lg\left(1-\frac{t}{n}\right) \end{aligned}$$

Pripomíname, že $t = \lfloor nq + \sqrt{2npq/\varepsilon} \rfloor$. Odhadneme výrazy $\frac{t}{n}\lg\frac{t}{n}$ a $\left(1-\frac{t}{n}\right)\lg\left(1-\frac{t}{n}\right)$:

$$\begin{aligned} \frac{t}{n} &= \frac{\lfloor nq + \sqrt{2npq/\varepsilon} \rfloor}{n} = \frac{nq + \sqrt{2npq/\varepsilon} + O(1)}{n} = q + O\left(\frac{1}{\sqrt{n}}\right) \\ \frac{t}{n}\lg\frac{t}{n} &= \left[q + O\left(\frac{1}{\sqrt{n}}\right)\right]\lg\left[q + O\left(\frac{1}{\sqrt{n}}\right)\right] = q\lg q + O\left(\frac{1}{\sqrt{n}}\right). \end{aligned} \tag{13.23}$$

Podobne odhadneme výraz $\left(1-\frac{t}{n}\right)\lg\left(1-\frac{t}{n}\right)$

$$\left(1-\frac{t}{n}\right)\lg\left(1-\frac{t}{n}\right) = p\lg p + O\left(\frac{1}{\sqrt{n}}\right). \tag{13.24}$$

Pripomenieme ešte, že prenosová rýchlosť kódu je $0 \leq R \leq 1 + p\lg p + q\lg q$ a mohutnosť kódu je $M = 2^{\lfloor R\cdot n \rfloor}$, t.j. existuje kladná konštanta $\beta > 0$ taká, že $R = 1 + p\lg p + q\lg q - \beta$. Potom

$$\begin{aligned} \frac{\lg M_n}{n} &= \frac{\lg 2^{\lfloor n(1+p\lg p+q\lg q-\beta)\rfloor}}{n} = \frac{\lfloor n(1+p\lg p+q\lg q-\beta)\rfloor}{n} \leq \\ (1+p\lg p+q\lg q-\beta) \end{aligned} \tag{13.25}$$

Teraz dosadíme odhady (13.24) a (13.25) do (13.21)

$$\begin{aligned} \frac{\lg(P^*(M_n,n,q)-\frac{1}{2}\varepsilon)}{n} &\leq \frac{\lg M_n}{n} - \left[1 + p\lg p + q\lg q + O\left(\frac{1}{\sqrt{n}}\right)\right] \leq \\ \leq (1+p\lg p+q\lg q-\beta) - \left[1+p\lg p+q\lg q+O\left(\frac{1}{\sqrt{n}}\right)\right] &= -\beta + O\left(\frac{1}{\sqrt{n}}\right). \end{aligned}$$

Nakoniec odhadneme zhora samotnú pravdepodobnosť $P^*(M_n,n,q)$:

$$\begin{aligned} \lg(P^*(M_n,n,q)-\frac{1}{2}\varepsilon) &\leq -\beta n + O\left(\sqrt{n}\right) = -\beta' n \\ P^*(M_n,n,q) &\leq \frac{1}{2}\varepsilon + 2^{-\beta' n}. \end{aligned}$$

kde $\beta' > 0$ je konštanta. Veta je dokázaná. □

Kapitola 14

Hranice parametrov samoopravných kódov

Pozrieme sa najmä na dolné odhady minimálnej vzdialenosti kódov.

Časť III

Matematické základy teórie kódovania

Konštrukcia kódov, skúmanie ich vlastností a návrh efektívnych metód kódovania a dekódovania si vyžadujú pomerne rozsiahle vedomosti z matematiky, informatiky a niektorých technických vied. Predpokladáme, že čitateľ absolvoval základné kurzy z matematiky (najmä z algebry a lineárnej algebry) a má aspoň základné poznatky z teórie pravdepodobnosti. V tejto časti uvedieme prehľad tých poznatkov z matematiky, ktoré čitateľ potrebuje na štúdium našej knihy. Prehľad matematiky má slúžiť na rýchle pripomenutie si zabudnutých poznatkov, prípadne doplnenie chýbajúcich fragmentov znalostí, ale v žiadnom prípade nemá ambíciu nahradiť systematický výklad uvedenej problematiky ako je napríklad [8, 9]. Čitateľovi, ktorý má záujem o hlbšie štúdium problematiky uvedenje v tejto kapitole, resp. objavil vo svojich vedomostiach hlbšie medzery, odporučíme práce, v ktorých nájde potrebné informácie spracované v dostupnej forme.

Kapitola 15

Algebra

Konštrukcie viacerých kódov vychádzajú z takých algebraických štruktúr, ako sú grupy, vektorové priestory, konečné polia, konečné geometrie a iné. Zavedieme tieto algebraické štrukúry a popíšeme ich najdôležitejšie vlastnosti.

Budeme predpokladať, že čitateľovi sú známe pojmy *množiny, podmnožiny, usporiadanej dvojice, kartézskeho súčinu množín, relácie a zobrazenia*, pozri napr. **??**.

15.1 Grupy

Nech je M nejaká množina. Zobrazenie $f : M \times M \to M$ budeme nazývať *binárnou operáciou na množine* M. Často budeme pracovať s nejakými podmnožinami základnej množiny a vtedy môže dôjsť k prípadu, keď výsledok operácie nad prvkami z podmnožiny už nebude prvkom danej podmnožiny. Ak pre ľubovoľné dva prvky $x, y \in M'$ a binárnu operáciu f platí $f(x, y) \in M'$, budeme hovoriť, že množina M' je *uzavretá vzhľadom na binárnu operáciu* f. *Algebraickou štruktúrou* alebo *algebraickým systémom* budeme nazývať množinu M s jednou alebo viacerými operáciami na M.

V ďalšom nebudeme skúmať nejaké abstraktné binárne operácie, ale takmer výlučne sa budeme zaoberať aditívnymi a multiplikatívnymi operáciami. Preto budeme namiesto zápisu $f(x, y)$ používať pre binárnu operáciu tradičné označenie $x \circ y$, kde $\circ$ označuje operátor "+"v prípade aditívnej operácie a "*"v prípade multiplikatívnej operácie.

Príklad. Symbolmi N, Z, Q, R budeme (aj v ďalších častiach knihy, ak nebude povedané inak) označovať množiny prirodzených, celých, racionálnych a reálnych čísel. Pripomíname, že nula (0) je prirodzené číslo. Na množine R definujeme štandardným spôsobom operácie sčítania ("$+$"), odčítania ("$-$"), násobenia ("$*$") a na množine $R - \{0\}$ aj operáciu delenia ("/"). Potom

1. množiny N, Z, Q, R sú uzavreté vzhľadom na operácie $+$ a $*$;
2. množiny Z, Q, R sú uzavreté vzhľadom na operáciu $-$;
3. množiny $Q - \{0\}, R - \{0\}$ sú uzavreté vzhľadom na operáciu $/$.

Operácia $\circ$ na množine M sa nazýva

- *asociatívnou*, ak pre ľubovoľné prvky $a, b, c \in M$ platí:

$$a \circ (b \circ c) = (a \circ b) \circ c$$

- *komutatívnou*, ak pre ľubovoľné prvky $a, b \in M$ platí:

$$a \circ b = b \circ a.$$

Množina M s operáciou $\circ$ (v ďalšom budeme takúto algebraickú štruktúru označovať $(M, \circ)$) sa nazýva *pologrupou*, ak je M uzavretá vzhľadom na operáciu $\circ$ a operácia $\circ$ je (na M) asociatívna.

Prvok $u \in (M, \circ)$ nazýva *neutrálnym prvkom množiny* M vzhľadom na operáciu $\circ$, ak pre ľubovoľný prvok $a \in M$ platí $a \circ u = u \circ a = a$. Pologrupa $(M, \circ)$ s neutrálnym prvkom u sa nazýva *monoidom.*

Nech je a ľubovoľný prvok monoidu $(M, \circ)$, potom prvok $b \in (M, \circ)$, pre ktorý platí

$$a \circ b = b \circ a = u,$$

sa nazýva *opačným* alebo *inverzným prvkom* prvku a[1]

Príklad. Uvažujme množinu Q racionálnych čísel s operáciami sčítania a násobenia. Neutrálnym prvkom pre operáciu sčítania je číslo 0, pre operáciu násobenia je neutrálnym prvkom číslo 1. Pre ľubovoľné racionálne číslo a existuje opačné číslo $-a \in Q$. Ak $a \neq 0$, tak pre a existuje inverzné číslo $1/a$.

Ďalší pojem je natoľko dôležitý, že si zasluhuje explicitnú definíciu.

Definícia 15.1.1. *Množina* G *s operáciou* $\circ$*, ktorá spĺňa nasledujúce podmienky*

1. G *je uzavretá vzhľadom na operáciu* $\circ$*,*
2. *operácia* $\circ$ *je asociatívna,*
3. G *obsahuje neutrálny prvok vzhľadom na operáciu* $\circ$*,*
4. *ku každému prvku množiny* G *existuje opačný prvok vzhľadom na operáciu* $\circ$

sa nazýva grupou.

[1] pojem opačný prvok sa zvykle používať v súvislosti s aditívnou operáciou a pojem inverzný prvok zasa v súvislosti s multiplikatívnou operáciou.

Poznámka. Je zrejmé, že pologrupa, v ktorej ku každému prvku existuje opačný prvok, je grupou.

Ak je operácia $\circ$ grupy $(G, \circ)$ komutatívna, grupa $(G, \circ)$ sa nazýva *komutatívnou* alebo *abelovskou grupou*.

Príklad. (Komutatívnymi) grupami sú napríklad nasledujúce algebraické štruktúry: $(Z, +), (Q, +), (R, +), (Q - \{0\}, *), (R - \{0\}, *)$. Uvedieme ešte jednu, menej obvyklú grupu. Množina $Z_3 = \{0, 1, 2\}$ s operáciou modulárneho sčítania $\oplus$ definovaného nasledujúcou tabuľkou je komutatívna grupa:

$\oplus$	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

Grupa, ktorá má konečný počet prvkov, sa nazýva *konečnou grupou*. Počet prvkov konečnej grupy G budeme označovať symbolom $|G|$ a nazývať *rádom grupy*. V grupe môže existovať podmnožina prvkov, ktoré spolu s operáciou defiinovanou na nadmnožine tvoria grupu. Takáto podmnožina s operáciou prevzatou s grupy (nadmnožiny) sa bude nazývať *podgrupou*. Zrejme najmenšou podgrupou grupy je jednoprvková množina obsahujúca neutrálny prvok grupy. Definujeme teraz pojem podgrupy exaktne.

Nech je $(G, \circ)$ grupa. Podmnožina $G_1 \subset G$, taká, že $G_1, \circ$ je grupa, sa nazýva *podgrupou grupy* G. Podgrupu možno charakterizovať aj nasledujúcim spôsobom (predpokladajme, že operácia $\circ$ je aditívna):

Veta 15.1.1. *Podmnožina* G_1 *grupy* $(G, +)$ *s operáciou* $+$ *je podgrupou (grupy* G*) práve vtedy, ak pre ľubovoľné prvky* $a, b \in G_1$ *platí* $a - b \in G_1$.

Dôkaz. Ak je $(G_1, +)$ podgrupou, tak spĺňa všetky štyri axiómy grupy. To znamená. že pre ľubovoľný prvok $b \in G_1$ je aj $-b \in G_1$; a pre ľubovoľné dva prvky z G_1 je prvkom G_1 aj ich súčet.

Nech na druhej strane platí, že ak $a, b \in G_1$ tak potom aj $a - b \in G_1$. Zoberieme ľubovoľný prvok $a \in G_1$, podľa predpokladu je aj $a - a = 0$ prvkom G_1. To však znamená, že pre ľubovoľný prvok $c \in G_1$ je aj k nemu opačný prvok $c = 0 - c \in G_1$; resp. pre ľubovoľné prvky $a, b \in G_1$ je aj $a + b = a - (-b) \in G_1$. To znamená, že G_1 je uzavretá vzhľadom na asociatívnu operáciu $+$, obsahuje neutrálny prvok a ku každému prvku obsahuje opačný prvok. T.j. $(G_1, +)$ je grupa. □

Poznámka. Tvrdenie predchádzajúcej vety možno zovšeobecniť nasledovne: Podmnožina G_1 grupy $(G, \circ)$ s operáciou $\circ$ je podgrupou (grupy G) práve vtedy, ak pre ľubovoľné prvky $a, b \in G_1$ platí $a \circ b' \in G_1$, kde b' je opačný/inverzný prvok k prvku b.

Príklad. 1. Uvažujme množinu $Z_6 = \{0, 1, 2, 3, 4, 5\}$ s operáciou $+$ definovanou nasle-

dovne:

$+$	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	2	3	4	5	0
2	2	3	4	5	0	1
3	3	4	5	0	1	2
4	4	5	0	1	2	3
5	5	0	1	2	3	4

Je zrejmé, že $(Z_6, +)$ je grupa. Jej podgrupami sú nasledujúce množiny $\{0\}, \{0, 3\}, \{0, 2, 4\}$ s operáciou $+$.

2. Nech je m ľubovoľné celé číslo. Dá sa ľahko ukázať, že množnina $mZ = \{0, m, -m, 2m, -2m, \dots\}$ všetkých celočíselných násobkov čísla m tvorí s operáciou $+$ aditívnu grupu, ktorá je podgrupou aditívnej grupy celých čísel $(Z, +)$. (Stačí dokázať, že rozdiel ľubovoľných dvoch celočíselných násobkov čísla m je opäť celočíselným násobkom čísla m.)

Uvažujme teraz grupu $(G, *)$ s multiplikatívnou operáciou, nech je $a \in G$ ľubovoľný prvok, a $m \in N$. Definujeme mocniny prvku a nasledovne:

1. $a^0 = 1$, kde 1 je neutrálny prvok grupy G vzhľadom na multiplikatívnu operáciu,
2. $a^{m+1} = a^m * a = \underbrace{a * a * \cdots * a}_{m+1}$,
3. $a^{-m} = (a^{-1})^m = \underbrace{a^{-1} * \cdots * a^{-1}}_{m}$.

Kvôli zjednodušeniu označenia budeme namiesto a^1 písať len a. Podobnm spôsobom možno zaviesť "mocniny"prvku a v prípade aditívnej operácie:

1. $a^0 = 0$, kde 0 je neutrálny prvok grupy G vzhľadom na aditívnu operáciu,
2. $a^{m+1} = a^m + a = \underbrace{a + a + \cdots + a}_{m+1}$,
3. $a^{-m} = (-a)^m = \underbrace{-a - a - \ldots - a}_{m}$

Tam kde nebude potrebné odlišovať multiplikatívne a aditívne grupy, budeme používať terminológiu multiplikatívnych grúp (inverzný prvok, jednotkový prvok, mocnina prvku a pod.)

Definícia 15.1.2. *Nech je* $(G, *)$ *grupa s multiplikatívnou operáciou, potom sa grupa* G *nazýva* cyklickou grupou, *ak existuje taký prvok* $g \in G$*, že pre ľubovoľné* $a \in G$ *existuje prirodzené číslo* $j \in N$ *také, že* $a = g^j$*; t.j., že všetky prvky grupy* G *možno vyjadriť v podobe mocnín prvku* g*. Prvok* g *sa nazýva* generátorom cyklickej grupy G *a cyklická grupa generovaná prvkom* a *sa označuje symbolom* (a).

Príklad. 1. Uvažujeme množinu $Z_5 = \{0, 1, 2, 3, 4\}$ s multiplikatívnou operáciou $*$ definovanou tabuľkou

$*$	0	1	2	3	4
0	0	0	0	0	0
1	0	2	3	4	5
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

$(Z_5, *)$ je síce len monoid, ale ak z neho odstránime problematický prvok 0, pre ktorý v $(Z_5, *)$ neexistuje inverzný prvok, dostávame multiplikatívnu grupu $(Z_5 - \{0\}, +)$. Táto grupa je cyklická a prvky $2, 3, 4$ sú jej generátormi.

2. Grupa $(Z_6, +)$ z príkladu 15.1 je cyklická. jej generátorom sú prvky $1, 5$.

3. Aditvna grupa celých čísel $(Z, +)$ je cyklická, jej generátormi sú čísla $1, -1$.

Uvažujme konečnú grupu $(G, *)$ a prvok $a \in G$. Množina mocnín $\{a, a^2, \dots\}$ je v dôsledku konečnosti G a uzavretosti G vzhľadom na operáciu $*$ konečná a s operáciou $*$ tvorí podgrupu nazývanú *podgrupou generovanou prvkom* a. Pre ľubovoľné $a \in G$ existuje prirodzené číslo k také, že $a^k = 1$, kde 1 je neutrálny prvok gupy G. Je zrejmé, že podgrupa generovaná prvkom a má prvky $a, a^2, \dots, a^k = 1$. *Rádom prvku* $a \in G$ budeme nazývať rád podgrupy generovanej prvkom a. Množinu mocnín $a, a^2, \dots, a^k = 1$ budeme nazývať *cyklom*.

Budeme skúmať niektoré vzťahy medzi grupami a ich pologrupami. Na to potrebujeme zaviesť pojem rozkladu alebo faktorizácie grupy.

Nech je M ľubovoľná množina. Potom $\mathcal{M} = \{M_1, \dots \ M_i \subseteq M\}$ systém podmožín množiny M sa nazýva *rozkladom množiny* M, ak spĺňa nasledujúce tri podmienky:

1. $\forall i \ M_i \neq \emptyset$,
2. $\bigcup_i M_i = M$,
3. $M_i \bigcap M_j = \emptyset; \quad i \neq j$.

Prvky systému $\mathcal{M}$ budeme nazývať *triedami rozkladu*.

Existujú zvláštne relácie na množine, ktoré definujú rozklady tejto množiny.

Nech je M ľubovoľná množina. Potom relácia $R \subseteq M \times M$ na množine M nazýva *reláciou ekvivalencie (ekvivalenciou)* , ak

1. pre ľubovoľné $x \in M, \quad (x, x) \in R$ (reflexívnosť),
2. pre ľubovoľné $x, y \in M$ platí, ak $(x, y) \in R$, tak potom aj $(y, x) \in R$ (symetria),
3. pre ľubovoľné $x, y, z \in M$ platí, ak $(x, y) \in R$ a $(y, z) \in R$ tak potom aj $(x, z) \in R$ (tranzitívnosť).

Poznámka. Relácia ekvivalencie sa zvykne označovať symbolom $\sim$ alebo $\equiv$. Namiesto $\equiv (x, y)$ budeme písať $x \equiv y$.

Príkladom ekvivalencie je rovnosť. Relácia ekvivalencie určuje rozklad množiny. Triedy rozkladu množiny M určené reláciou ekvivalencie $\equiv$ na množine M sa nazývajú *triedami ekvivalencie* a sú definované nasledovne:

$$[a] = \{x \in M;\ x \equiv a\};$$

t.j. jednu triedu rozkladu/ekvivalencie tvoria všetky tie prvky množiny M, ktoré sú navzájom ekvivalentné. Trieda rozkladu je jednoznačne určená ľubovoľným svojím prvkom, a preto sa z každej triedy rozkladu vyberie nejaký prvok, ktorý reprezentuje danú triedu, nazývaný *reprezentantom* alebo *predstaviteľom triedy.*

Príklad. Nech je $m \in N$ prirodzené číslo, $m > 1$. Relácia $\equiv$ definovaná na množine celých čísel nasledujúcim spôsobom

$$a \equiv b \Leftrightarrow m|(a - b)$$

je ekvivalencia. Rozklad množiny celých čísel definovaný touto ekvivalenciou pozostáva z tried $[0], ...[m - 1]$. Prvkami triedy ekvivalencie $[t]$ sú všetky celé čísla, ktoré majú rovnaký zvyšok po delení číslom m ako reprezentant triedy. Túto ekvivalenciu budeme ešte potrebovať, a preto pre ňu zavedieme špeciálne označenie; to že pre celé čísla a, b platí $m|(a - b)$, budeme označovať nasledovne

$$a \equiv b \mod m.$$

Uvažujme grupu $(G, *)$ a nejakú jej podgrupu $(H, *)$; $H = \{h_1 \dots h_k\}$. Zostrojíme teraz rozklad grupy G nasledujúcim spôsobom:

1. prvou triedou rozkladu je podgrupa H; reprezentantom tejto triedy rozkladu je prvok 1;
2. predpokladáme, že sme zostrojili $i-1$ tried rozkladu; ak existuje prvok $a_i \in G$, torý nepatrí do žiadnej z prvých $i - 1$ tried rozkladu, vyberieme ho ako reprezentanta i-tej triedy, ktorú zostrojíme nasledujúcim spôsobom:

$$[a_i] = a_i * h_1, \dots, a_i * h_k$$

3. krok (2) opakujeme dovtedy, kým budú existovať prvky množiny G, ktoré nepatria do žiadnej triedy rozkladu.

Je zrejmé, že triedy $[a_i]$ sú neprázdne a že každý prvok množiny G patrí do niektorej triedy. Ukážeme ešte, že $[a_i] \bigcap [a_j] = \emptyset$ $i \neq j$. Predpokladajme opak, t.j. že existuje prvok $x \in [a_i] \bigcap [a_j]$. To však znamená, že existujú prvky $h_r, h_s \in H$ také, že

$$x = a_i * h_r = a_j * h_s.$$

Keďže G je grupa, obsahuje prvok a_i^{-1}, inverzný k prvku a_i. To znamená, že

$$a_i^{-1} * (a_i * h_r) = (a_i^{-1} * a_i) * h_r = h_r = a_i^{-1} * (a_j * h_s) = (a_i^{-1} * a_j) * h_s;$$

t.j. $(a_i^{-1} * a_j) \in H$, a teda $a_i \in a_j * H$, z čoho vyplýva, že $[a_i] = [a_j]$. Systém $\{[a_i]\}_i$ teda naozaj predstavuje rozklad grupy G, ktorý budeme nazývať *rozkladom grupy* G *podľa podgrupy* H. Rozklad grupy podľa podgrupy je uvedený v nasledujúcej tabuľke.

	h_1	h_2	$\dots$	h_k
a_2	$a_2 * h_1$	$a_2 * h_2$	$\dots$	$a_2 * h_k$
$\vdots$	$\vdots$	$\vdots$	$\dots$	$\vdots$
a_i	$a_i * h_1$	$a_i * h_2$	$\dots$	$a_i * h_k$
$\vdots$	$\vdots$	$\vdots$	$\dots$	$\vdots$
a_m	$a_m * h_1$	$a_m * h_2$	$\dots$	$a_m * h_k$

Poznámka. V prípade ak multiplikatívna operácia $*$ nie je komutatívna, bude potrebné rozlišovať násobenie sprava a zľava, pretože vo všeobecnosti $a * H \neq H * a$. Triedy $a * H$ ($H * a$) budeme nazývať *ľavými*, resp.*pravými* triedami rozkladu. Podgrupa H grupy $(G, *)$ sa nazýva *normálna*, ak pre každý prvok $a \in G$ platí $a * H = H * a$; t.j. ak sa pravé a ľavé triedy rozkladu grupy G podľa H rovnajú. My budeme pracovať s abelovskými grupami, ktorých podgrupy sú normálne podgrupy.

Z konštrukcie rozkladu konečnej grupy podľa jej podgrupy vyplýva nasledujúci klasický výsledok.

Veta 15.1.2 (Lagrange)**.** *Rád konečnej grupy je celočíselným násobkom rádu každej jej podgrupy.*

Pripomíname, že rád prvku bol definovaný ako rád podgrupy, ktorú daný prvok generoval. Z Lagrangeovej vety potom vyplýva nasledujúci

Dôsledok. Rád každého prvku konečnej grupy G je deliteľom rádu grupy G.

Vráťme sa ešte k systému tried, vytvorených pri rozklade grupy G podľa normálnej podgrupy H; ktorý budeme označovať symbolom G/H;

$$G/H = \{[a_i] = a_i * H\}_i.$$

Definujeme na systéme G/H binárnu operáciu $*$ nasledovne:

$$[a_i] * [a_j] = [a_i * a_j].$$

Dá sa ľahko overiť, že $(G/H, *)$ tvorí grupu, ktorú budeme nazývať *faktorovou grupou (grupy* G *podľa (normálnej) podgrupy* H*)*.

Príklad. Faktorizujeme grupu $(Z, +)$ podľa podgrupy $(2Z, +)$, kde $2Z$ je množina všetkých párnych celých čísel. Rozklad $Z/2Z$ bude mať dve triedy, $2Z, Z - 2Z$, pričom trieda $Z - 2Z$ pozostáva zo všetkých nepárnych celých čísel. Vyberme ako reprezentantov tried rozkladu čísla $0, 1$. Potom $2Z = [0]$ a $Z - 2Z = [1]$. Operácia sčítania na Z/H je definovaná nasledujúcou tabuľkou:

$+$	$[0]$	$[1]$
$[0]$	$[0]$	$[1]$
$[1]$	$[1]$	$[0]$

Faktorová grupa $(Z/2Z, +)$ zodpovedá (až na označenie prvkov) grupe $(Z_2, \oplus)$, kde $Z_2 = \{0, 1\}$ a $\oplus$ označuje sčítanie $\mod 2$.

Zovšeobecníme predchádzajúci príklad. Pre ľubovoľné prirodzené číslo $m \geq 2$ [2] označíme symbolom $(Z_m, +)$ faktorovú grupu $(Z/mZ, +)$ s operáciou sčítania $\mod m$. Kvôli zjednodušeniu zápisu budeme prvky množiny Z_m označovať symbolmi $0, \ldots, m-1$. Grupy Z_m budeme v ďalšom často používať.

15.2 Okruhy

Doteraz sme sa zaoberali algebraickými štruktúrami s jednou binárnou operáciou. V tejto časti zavedieme agebraické štruktúry s dvoma binárnymi operáciami - sčítaním a násobením.

Definícia 15.2.1. Okruh $(A, +, \cdot)$ *je množina* A *spolu s dvoma binárnymi operáciami, označenými ako* $+$ *a* $\cdot$*, ktorá spĺňa nasledujúce podmienky:*

1. A *je abelovská grupa vzhľadom na aditívnu operáciu* $+$*,*
2. *multiplikatívna operácia* $\cdot$ *je asociatívna, t.j. pre ľubovoľné* $a, b, c \in A$ *platí* $(a \cdot b) \cdot c = a(\cdot b \cdot c)$*;*
3. *platí distributívny zákon; t.j. pre ľubovoľné* $a, b, c \in A$ *platí* $a \cdot (b + c) = a \cdot b + a \cdot c$ *a* $(b + c) \cdot a = b \cdot a + c \cdot a$.

Okruh $(A, +.\cdot)$ budeme kvôli zjednodušeniu zápisu označovať symbolom A, symbolom 0 budeme označovať neutrálny prvok vzhľadom na aditívnu operáciu a symbolom 1 neutrálny prvok vzhľadom na multiplikatívnu operáciu (ak okruh taký prvok obsahuje); opačný prvok k prvku a vzhľadom na aditívnu operáciu budeme označovať symbolom $-a$. Namiesto $a + (-b)$ budeme písať $a - b$ a súčin $a \cdot b$ budeme stručnejšie zapisovať ako ab. Dá sa ľahko odvodiť, že pre ľubovoľné $a, b \in A$ platí $a \cdot 0 = 0$ a $(-a)b = a(-b) = -ab$. Ak $ab = c$, budeme hovoriť, že prvky a, b sú *deliteľmi prvku* c. Je zrejmé, že $1 \cdot c = c$; t.j. prvky 1 a c sú deliteľmi c. Ak sú delitele a, b, rôzne od prvkov $c, 1$ tak sa nazývajú *vlastnými deliteľmi* prvku c.

Teraz zavedieme niektoré algebraické štruktúry, ktoré majú okrem vlastností okruhu aj ďalšie vlastnosti.

Definícia 15.2.2. *Okruh* A *sa nazýva*

1. *unitárnym, ak v* A *existuje prvok* 1*, neutrálny prvok vzhľadom na multiplikatívnu operáciu;*
2. *komutatívnym okruhom, ak je operácia* $\cdot$ *komutatívna;*
3. *oborom integrity, ak je komutatívnym unitárnym okruhom,* $1 \neq 0$ *a z rovnosti* $ab = 0$ *vyplýva* $a = 0$ *alebo* $b = 0$*,*

[2] Posúdenie prípadov $m \leq 1$ ponechávame na čitateľa

4. *telesom, ak je* $(A - \{0\}, \cdot)$ *multiplikatívna grupa,*

5. *poľom, ak je komutatívnym telesom.*

Kľúčovým pojmom, s ktorým budeme v teórii samoopravných kódov neustále pracovať, je pojem (konečného) poľa. Konečným poliam sa budeme v tejto kapitole venovať podrobnejšie, a preto si len stručne zrekapitulujeme vlastnosti poľa. Pole je algebraická štruktúra s aditívnou a multiplikatívnou operáciou. Tvorí abelovskú grupu vzhľadom na aditívnu operáciu a jeho nenulové prvky tvoria abelovskú grupu vzhľadom na multiplikatívnu operáciu. Obe operácie sú navzájom zviazané distributívnym zákonom: $a(b + c) = ab + ac$. Pole je aj oborom integrity, to znamená, že nemá vlastných deliteľov nuly; resp. z toho, že $ab = 0$ vyplýva $a^{-1}ab = b = 0$; resp. $abb^{-1} = a = 0$.

Vrátime sa ešte k pojmu okruhu. Podobne ako sme pre grupu definovali pojem podgrupy, zavedieme pre okruh najprv pojem podokruhu a potom definujeme podokruhy so špeciálnymi vlastnosťami.

Definícia 15.2.3. *Podmnožina* S *okruhu* A *sa nazýva podokruhom okruhu* A*, ak je uzavretá vzhľadom na operácie* $+$ *a* $\cdot$ *a tvorí vzhľadom na tieto operácie okruh.*

Pri konštrukcii polynómov budeme potrebovať k podokruhu pridávať nové prvky. Nasledujúca veta [8] hovorí, ako to možno urobiť tak, aby výsledná algebraická štruktúra zostala okruhom.

Veta 15.2.1. *Nech* A *je podokruh unitárneho komutatívneho okruhu* $(B, +, \cdot)$ *a nech* $1 \in A$*. Potom pre ľubovoľný prvok* $b \in B$ *je najmenší podokruh generovaný množinou* $A \cup \{b\}$ *okruh* $C, +\cdot)$*, kde*

$$C = \{a_0 + a_1 b + \cdots + a_n b^n; n \in N, a_0, a_1, \ldots, a_n \in A\}.$$

Dôkaz. Dá sa ľahko overiť, že $(C, +, \cdot)$ je okruh. Keďže $A \cup \{b\} \subseteq B$, $(C, +, \cdot)$ je podokruh okruhu $(B, +, \cdot)$. Ukážeme, že $(C, +, \cdot)$ je najmenší podokruh okruhu $(B, +, \cdot)$ obsahujúci množinu $A \cup \{b\}$. Predpokladajme, že existuje okruh $(C', +, \cdot)$ taký, že $A \cup \{b\} \subseteq C' \subset C \subseteq B$. Nech $d \in C - C'$, potom existuje $m \in N$ a prvky $a_i \in A$, $i = 0 \ldots m$ také, že

$$d = a_0 + a_1 b + \cdots + a_m b^m.$$

Kedže $b \in C'$ a $(C', +, \cdot)$ je okruh, potom pre ľubovoľné $k \in N$ aj $b^k \in C'$ a teda aj $a_k b^k \in C'$ pre $a_k \in A \subseteq C'$. To znamená, že $d \in C'$. Dostávame spor s predpokladom, čo dokazuje platnosť vety. □

Poznámka. Podokruh $(C, +, \cdot)$ generovaný množinou $A \cup \{b\}$ budeme označovať symbolom $(A[b], +, \cdot)$.

Definícia 15.2.4. *Podokruh* J *okruhu* A *sa nazýva ideálom ak pre každé* $a \in J$ *a* $r \in A$ *platí* $ar \in J$ *a* $ra \in J$.

Príklad. Uvažujme okruh $(Z, +, \cdot)$. Množina všetkých párnych čísel tvorí podokruh $(Z_2, +, \cdot)$ okruhu Z. Keďže súčin ľubovoľného celého čísla a párneho čísla je párne číslo, $(Z_2, +, \cdot)$ je ideálom okruhu Z.

Pri štúdiu vlastností tzv. cyklických kódov budeme pracovať s algebraickou štruktúrou nazvanou okruhom hlavných ideálov.

Definícia 15.2.5. *Nech je A komutatívny okruh. Ideál J okruhu A sa nazýva hlavným ideálom, ak v okruhu A existuje prvok a, ktorý je generátorom cyklickej grupy ideálu J. Ideál J sa nazýva hlavným ideálom generovaným prvkom a.*

Definícia 15.2.6. *Komutatvny okruh A nazývame okruhom hlavných ideálov, ak je každý ideál okruhu A hlavný.*

Grupy bolo možné faktorizovať podľa ich normálnych podgrúp a vytvárať faktorové grupy. Podobne je možné faktorizovať okruhy. Úlohu normálnej podgrupy pri faktorizácii okruhov zohráva ideál.

Veta 15.2.2. *Nech je I ideál okruhu $A, +, \cdot)$; nech $[a] = \{a+I,\ a \in A\}$ je trieda rozkladu A/I aditívnej grupy $(A, +)$ podľa normálnej podgrupy $(I, +)$ obsahujúca prvok (reprezentanta) a. Potom množina A/I tried rozkladu s operáciami $+, \cdot$ definovanými nasledovne*

$$[a] + [b] = [a + b], \quad [a] \cdot [b] = [a \cdot b],$$

kde $a, b \in A$, tvorí okruh. Ak je okruh A komutatívny, tak je aj okruh $(A/I, +, \cdot)$ komutatívny.

Dôkaz. Je priamočiary; stačí overiť platnosť axióm okruhu pre $(A/I, +, \cdot)$. Prenecháme preto túto úlohu čitateľovi. □

Poznámka. Okruh $(A/I, +, \cdot)$ nazýva *faktorovým okruhom* okruhu A podľa I.

Príklad. Uvažujme aditívnu faktorovú grupu $(Z/Z_q, +)$ Táto grupa má q prvkov: $[0], \ldots [q-1]$; $[j] = \{m \in Z; m \mod q = j; j = 0, \ldots q-1\}$. Operácia násobenia prvkov faktorovej grupy $(Z/Z_q, +)$ je asociatívna

$$\forall [a], [b], [c] \in Z/Z_q; [a] \cdot ([b] \cdot [c]) = ([a] \cdot [b]) \cdot [c]$$

a platí distributívny zákon

$$\forall [a], [b], [c] \in Z/Z_q; [a] \cdot ([b] + [c]) = ([a] \cdot [b]) + [a] \cdot [c], \quad ([b] + [c]) \cdot [a] = [b] \cdot [a] + [c] \cdot [a].$$

To znamená, že Z/Z_q s uvedenou aditívnou a multiplikatívnou operáciou tvorí faktorový okruh $(Z/Z_q, +, \cdot)$.

Nie všetky okruhu celých čísel sa prenášajú automaticky do faktorového okruhu $(Z/Z_q, +, \cdot)$. Okruh celých čísel je oborom integrity, t.j. z rovnosti $a \cdot b = 0$ vyplýva, že aspoň jeden z prvkov a, b je nulový. Uvažujme teraz faktorový okruh $(Z/Z_6, +, \cdot)$. Je zrejmé, že $[2][3] = [2 \cdot 3] = [6] = [0]$, pričom $[2] \neq [0], [3] \neq [0]$.

Na záver tejto časti zavedieme ešte jeden dôležitý pojem.

Definícia 15.2.7. *Nech je* R *ľubovoľný okruh a existuje také kladné celé číslo* n*, že pre ľubovoľné* $r \in R$ *platí* $nr = 0$*. Potom najmenšie také číslo* n *sa nazýva* charakteristikou okruhu R*. O okruhu* R *v takom prípade hovoríme, že má kladnú charakteristiku. Ak kladné celé číslo* n *s požadovanými vlastnosťami neexistuje, hovoríme, že okruh* R *má charakteristiku* 0*.*

Príklad. Okruhy komplexných, reálnych, racionálnych, celých, prirodzených čísel majú charakteristiku 0, okruh $(Z/Z_6, +, \cdot)$ má charakteristiku 6.

Charakteristika okruhu nie je nezávislá od ostatných vlastností okruhu.

Veta 15.2.3. *Nech je* $R \neq \{0\}$ *obor integrity s jednotkou a kladnou charakteristikou. Potom je charakteristika* R *prvočíslo.*

Dôkaz. Keďže R obsahuje nenulové prvky, R má charakteristiku $n \geq 2$. Ak by n nebolo prvočíslo, dalo by sa zapísať v podobe súčinu celých čísel, $n = km$, kde $k, m \in Z, 1 < k, m < n$. Nech e označuje multiplikatívnu jednotku oboru integrity R. Potom z vyššie uvedeného výplýva, že $0 = ne = (km)e = (ke)(me)$. Nakoľko R je obor integrity, z poslednej rovnosti vyplýva, že buď $ke = 0$ alebo $me = 0$. Ale potom je charakteristika R buď k alebo m – spor s minimálnosťou n. □

Predchádzajúca veta má zaujímvý a veľmi dôležitý dôsledok pre konečné polia.

Veta 15.2.4. *Konečné pole má prvočíselnú charakteristiku.*

Dôkaz. Konečné pole je oborom integrity a obsahuje jednotkový prvok (e). To znamená, že stačí dokázať, že má konečnú charakteristiku. Uvažujme postupnosť kladných celočíselných násobkov jednotkového prvku

$$e, 2e, 3e, \ldots.$$

Vzhľadom na to, že pole je uzavreté na operáciu sčítania, predchádzajúca postupnosť obsahuje prvky daného konečného poľa. Keďže pole je konečné, budú sa v uvedenej postupnosti prvky opakovať. Zoberme najmenšie také $k_1 > k_2$, že $k_1e = k_2e$. Potom $(k_1 - k_2)e = 0$, a teda dané pole má kladnú charakteristiku. □

15.3 Polynómy a okruhy polynómov

Ďalším dôležitým pojmom teórie samoopravných kódov je pojem *polynómu*. Kódové slová sa dajú reprezentovať pomocou polynómov, syndrómy chýb sa dajú vypočítať dosadzovaním istých hodnôt do plynómov reprezentujcich prijaté slová a pozície chýb v kódových slovách sa dajú určiť pomocou koreňov polynómu nazývaného lokátorom chýb. Zavedieme preto pojem polynómu a popíšeme najdôležitejšie vlastnosti polynómov.

Nech je A unitárny komutatívny okruh[3] a nech je B jeho podokruh, obsahujúci jednotkový prvok. Nech $x \in A - B$ je ľubovoľný prvok. Prvky podokruhu $B[x]$ sa podľa vety

[3] v ďalšom budeme najčastejšie predpokladať, že A je pole

15.2.1 dajú vyjadriť v tvare

$$a_0 + a_1x + a_2x^2 \cdots + a_nx^n;\ a_k \in B,\ n \in N; \tag{15.1}$$

pričom pre $n > 0$ predpokladáme, že $a_n \neq 0$. V niektorých prípadoch by výber prvku x mohol viesť k nejednoznačnosti vyjadrenia prvkov z okruhu $B[x]$. Predpokladajme, že existuje prvok, označme ho symbolom $a(x)$, ktorý je možné vyjadriť v tvare 15.1 dvoma rozličnými spôsobmi

$$a(x) = a_0 + a_1x + a_2x^2 \cdots + a_nx^n = b_0 + b_1x + b_2x^2 \cdots + b_mx^m.$$

Bez ujmy na všeobecnosti budeme predpokladať, že $n \geq m$. Ukážeme, že jednoznačnosť vyjadrenia prvku $a(x)$ je ekvivalentná jednoznačnosti vyjadrenia nulového prvku okruhu $B[x]$ v tvare 15.1. Vypočítame rozdiel dvoch rozličných reprezentácií prvku $a(x)$:

$$\begin{aligned} 0 &= a(x) - a(x) = a_0 + a_1x + a_2x^2 \cdots + a_nx^n - (b_0 + b_1x + b_2x^2 \cdots + b_mx^m) = \\ &= (a_0 - b_0) + (a_1 - b_1)x + (a_2 - b_2)x^2 \cdots + (a_m - b_m)x^m + a_{m+1}x^{m+1} + \\ &\quad \cdots + a_nx^n. \end{aligned} \tag{15.2}$$

Z rovnosti 15.2 vyplýva, že ľubovoľný prvok okruhu $B[x]$ je možné vyjadriť jednoznačne v tvare 15.1 práve vtedy, ak rovnosť

$$0 = c_0 + c_1x + \cdots c_nx^n$$

platí práve vtedy, ak

$$c_0 = c_1 \cdots = c_n = 0. \tag{15.3}$$

Ak totiž platí 15.3, pre vyjadrenie $a(x)$ platí $n = m$ a $a_k = b_k$ $k = 0, \ldots, n$. Prvok $x \in A$, pre ktorý je možné nulový prvok okruhu $B[x]$ vyjadriť jednoznačne, t.j. platí 15.3, sa nazýva *transcendentým prvkom* nad B; v opačnom prípade sa x nazýva *algebraickým prvkom* nad B. Ak je prvok x transcendentný nad okruhom B, budeme okruh $B[x]$ nazývať *okruhom polynómov neurčitej* x nad B. Prvky okruhu $B[x]$ budeme nazývať *polynómami v neurčitej/premennej* x, alebo len stručne, polynómami. Nech je $a(x) \in B[x]$ polynóm, $a(x) = a_0 + a_1x + a_2x^2 \cdots + a_nx^n$. Prvky $a_0, \ldots, a_n \in B$ nazývame koeficientami a sčítance a_kx^k členmi polynómu. Ak $n \neq 0$, číslo n nazývame *stupňom polynómu*, koeficient a_n *vedúcim koeficientom* a člen a_nx^n *vedúcim členom* polynómu $a(x)$. Stupeň polynómu $f(x)$ budeme označovať symbolom $\deg(f(x))$. Polynóm $f(x) = 0$ nazvame *nulovým polynómom* a stupeň nulového polynómu definujeme ako $\deg(0) = -\infty$. Ak je vedúci člen polynómu $a(x)$, $a_n = 1$, polynóm $a(x)$ nazývame *normovaným polynómom*. Uvedieme ešte, že ak sú $a(x), b(x)$ dva polynómy, $a(x) = a_0 + a_1x + \cdots + a_nx^n$; $b(x) = b_0 + b_1x + \cdots + b_mx^m$, tak ich súčinom je polynóm $a(x)b(x) = a_0b_0 + (a_1b_0 + a_0b_1)x + \cdots + (a_0b_k + a_1b_{k-1} + \cdots + a_kb_0)x^k + \cdots + a_nb_mx^{m+n}$.

Príklad. Množiny reálnych R a racionálnych Q čísel s operáciami sčítania a násobenia tvoria okruhy. Je zrejmé, že okruh racionálnych čísel je podokruhom reálnych čísel. Vyberieme rôzne reálne čísla x a vytvoríme okruhy $Q[x]$.

1. Vyberme najprv ako neurčitú racionálne číslo; $x \in Q$. Potom však $Q[x] = Q$, lebo $a_0 + a_1x + \cdots + a_nx^n \in Q$ pre $x,\ a_k \in Q$.

2. Položíme teraz $x = \sqrt{2}$; je zrejmé, že $x \notin Q$. Nakoľko však $\left(\sqrt{2}\right)^{2k} = 2^k \in Q$, $Q[x] = \{a_0 + a_1\sqrt{2};\ a_0, a_1 \in Q\}$. Naviac, napríklad prvok 4 sa dá vyjadriť v tvare 15.2.1 viacerými spôsobmi: $4 = 4 + 0x + 0x^2 + \cdots = 2 + 0x + 1x^2 = 2x^2 = x^4$, atď.

3. Napokon, položme $x = e$. Prvok e je transcendentný a okruh $Q[e]$ tvoria prvky, ktoré sa dajú jednoznačne vyjadriť v tvare

$$a_0 + a_1 e + \cdots + e_n e^n;\ n \in N,\ a_k \in Q.$$

V ďalšom sa budeme zaoberať deliteľnosťou polynómov, a preto budeme skúmať polynómy nad nejakým poľom F. Okruh polynómov nad poľom F budeme označovať symbolom $F[x]$.

Definícia 15.3.1. *Nech je* $F[x]$ *okruh polynómov nad poľom* F *a nech sú* $f(x), g(x)$ *polynómy z okruhu* $F[x]$. *Budeme hovoriť, že polynóm* $g(x)$ *delí polynóm (je deliteľom polynómu)* $f(x)$, *ak v okruhu* $F[x]$ *existuje taký polynóm* $q(x)$, *že* $f(x) = g(x) \cdot q(x)$.

Je zrejmé, že každý polynóm je deliteľný sebou samým, resp. (vzhľadom na to, že F je pole) prvkami poľa F, ktoré predstavujú v okruhu $F[x]$ polynómy nultého stupňa, resp. konštanty. Tieto delitele sú triviálne delitele polynómu. Ak polynóm $f(x)$ nemá v okruhu $F[x]$ iných deliteľov okrem triviálnych, budeme ho nazývať *ireducibilným polynómom v okruhu* $F[x]$. Polynóm, ktorý nie je ireducibilný, budeme nazývať *reducibilným polynómom*. Pripomíname, že ireducibilita polynómu sa vzťahuje na istý okruh polynómov. Napríklad, polynóm $f(x) = x^2 - 2$ je ireducibilný v okruhu polynómov $Q[x]$, ale v okruhu $R[x]$ [4] má netriviálne delitele $x - \sqrt{2}$ a $x + \sqrt{2}$. Nech je $f(x) = f_0 + f_1 x + \cdots + f_n x^n$ je ľubovoľný polynóm okruhu $F[x]$. Pre ľubovoľný prvok $a \in F$ definujeme hodnotu $f(a) = f_0 + f_1 a + \cdots + f_n a^n$. Potom polynóm $f(x)$ predstavuje zobrazenie (polynomickú funkciu) $f : F \to F$. Hodnotu $f(a)$ budeme nazvať *hodnotou polynómu* $f(x)$ *pre prvok* a. Dôležité sú tie prvky poľa F, ktoré sa polynomickou funkciou zobrazujú na nulový prvok poľa F.

Definícia 15.3.2. *Nech je* $f[x]$ *okruh polynómov nad poľom* F *a nech* $f(x) \in F[x]$ *je polynóm. Prvok* $a \in F$ *budeme nazývať koreňom polynómu* $f(x)$, *ak* $f(a) = 0$.

V okruhu polynómov nemôžeme vo všeobecnosti zaviesť delenie polynónov, ale podobne ako pre okruh celých čísel môžeme aj v okruhu polynómov $F[x]$ zaviesť delenie so zvyškom.

Veta 15.3.1 (O deliteľnosti polynómov)**.** *Nech sú* $f(x), g(x)$ *ľubovoľné polynómy nad poľom* F *a nech* $g(x) \neq 0$. *Potom v okruhu* $F[x]$ *existujú polynómy* $q(x), r(x)$ *také, že*

$$f(x) = q(x)g(x) + r(x), \tag{15.4}$$

kde $\deg(r(x)) < \deg(g(x))$ *a polynómy* $q(x), r(x)$ *sú určené jednoznačne.*

[4] stačil by aj okruh polynómov $Q[\sqrt{2}][x]$

Dôkaz. Budeme robiť indukciou vzhľadom na stupeň polynómu $f(x)$.

1. Nech $\deg(f(x)) < \deg(g(x))$. Potom $q(x) = 0$ a $r(x) = f(x)$.
2. Predpokladajme, že tvrdenie vety platí pre $\deg(f(x)) \geq \deg(g(x))$; $\deg(f(x)) < n$.
3. Dokážeme platnosť tvrdenia vety pre $\deg(f(x)) = n$, $\deg(f(x)) \geq \deg(g(x))$. Nech $f(x) = f_n x^n + \cdots + f_1 x + f_0$; $g(x) = g_m x^m + \cdots + g_1 x + g_0$, $n > m$. Odčítame od polynómu $f(x)$ polynóm $f_n g_m^{-1} x^{n-m} \cdot g(x)$, kde g_m^{-1} je prvok poľa F inverzný k vedúcemu koeficientu polynómu $g(x)$ a dostaneme polynóm $f_1(x)$. Tento polynóm má stupeň $\deg(f_1(x)) < n$, a teda podľa indukčného predpokladu existujú také polynómy $q_1(x), r_1(x)$ nad poľom F, že

$$f_1(x) = q_1(x)g(x) + r_1(x).$$

Potom však možno v tvare 15.4 vyjadriť aj polynóm $f(x)$:

$$f(x) = f_1(x) + f_n g_m^{-1} x^{n-m} \cdot g(x) = (f_n g_m^{-1} x^{n-m} + q_i(x)) \cdot g(x) + r_1(x);$$

t.j. $r(x) = r_1(x)$ a $q(x) = f_n g_m^{-1} x^{n-m} + q_i(x)$.

Predpokladajme ešte, že existujú polynómy $q'(x) \neq q(x)$ a $r'(x) \neq r(x)$, také, že

$$q'(x)g(x) + r'(x) = f(x) = q(x)g(x) + r(x).$$

Potom však

$$0 = q(x)g(x) + r(x) - q(x)'g(x) - r'(x) = (q(x) - q'(x))g(x) + (r(x) - r'(x)).$$

Predpokladajme, ža polynóm $(r(x) - r'(x))$ je nenulový. Keďže polynóm $(q(x) - q'(x))g(x)$ je buď nulový, alebo má stupeň

$$\deg((q(x) - q'(x))g(x)) \geq \deg(g(x)) > \max\{\deg(r(x)), \deg(r'(x))\} \geq \deg(r(x) - r'(x)),$$

dostávame, že $(q(x) - q'(x))g(x) + (r(x) - r'(x)) \neq 0$. Spor. To znamená, že $(r(x) - r'(x)) = 0$ a $(q(x) - q'(x))g(x) = 0$. Keďže $g(x) \neq 0$, musí byť $(q(x) - q'(x) = 0$, a teda polynómy $q(x)$ (podiel) a $r(x)$ (zvyšok) sú určené jednoznačne. □

Vrátime sa ku skúmaniu vlastností okruhu polynómov $F[x]$. Zo skutočnosti, že v okruhu $F[x]$ je definované delenie so zvyškom (veta 15.3.1), vyplýva známy fakt, že každý ideál okruhu $F[x]$ je hlavný; t.j. že okruh $F[x]$ je okruhom hlavných ideálov.

Veta 15.3.2. *Nech je* $F[x]$ *okruh polynómov nad poľom* F. *Potom je* $F[x]$ *okruhom hlavných ideálov.*

Dôkaz. Ukážeme, že pre každý ideál $J \neq (0)$ okruhu $F[x]$ existuje jednoznačne určený normovaný polynóm $g(x) \in F[x]$ taký, že $J = (g(x))$. Keďže F je pole, okruh $F[x]$ je oborom integrity. Nech je $J =\neq (0)$ ideál okruhu $F[x]$ a nech je $h(x)$ polynóm najmenšieho stupňa, ktorý sa v J nachádza; nech je b vedúci koeficient polynómu $h(x)$. Položíme $g(x) = b^{-1}h(x)$. Je zrejmé, že $g(x) \in J$ a $g(x)$ je normovaný polynóm. Zoberieme teraz

ľubovoľný polynóm $f(x) \in J$ a vyjadríme ho v tvare 15.4: $f(x) = q(x)g(x) + r(x)$, pričom $\deg(r(x)) < \deg(g(x)) = \deg(h(x))$. Keďže J je ideál, polynóm $f(x) - q(x)g(x) = r(x) \in J$. Nakoľko $h(x)$ bol polynóm najmenšieho stupňa v J, polynóm $r(x)$ je nulový. To znamená, že (ľubovoľný polynóm z ideálu J) $f(x)$ je násobkom polynómu $g(x)$ a teda, $J = (g(x)$. Ostáva ešte ukázať jednoznačnosť výberu polynómu $g(x)$. Predpokladajme, že existuje iný normovaný polynóm $g_1(x) \in F[x]$, ktorý je generátorom ideálu J. Potom však $g(x) = c_1(x)g_1(x)$ a $g_1(x) = c_2(x)g(x)$. Z uvedených rovností vyplýva, že $g(x) = c_1(x)c_2(x)g(x)$, a teda polynómy $c_1(x), c_2(x)$ sú konštantné. Keďže obidva polynómy $g(x), g_1(x)$ sú normované, $c_1c_2 = 1$, a teda $g(x) = g_1(x)$. Tým je dokázaná jednoznačnosť určenia generátora ideálu J. □

Každý nenulový polynóm $f(x)$ okruhu $F[x]$ definuje (hlavný) ideál, $(f(x))$. Rozložíme teraz okruh $F[x]$ podľa ideálu $(f(x))$; triedy rozkladu budú množiny polynómov $g(x)+(f(x))$, kde $g(x) \in F[x]$. (Triedu rozkladu $g(x) + (f(x))$ budeme označovať symbolom $[g(x)]$.) Dve triedy rozkladu, $[a(x)], [b(x)]$ sa budú zhodovať práve vtedy, ak $a(x) - b(x) \in (f(x))$; t.j. ak $f(x)|(a(x)(x)$. Táto podmienka sa dá vyjadriť aj tak, že polynómy $a(x), b(x)$ dávajú po delení polynómom $f(x)$ rovnaký zvyšok. Každá z tried rozkladu $[g(x)]$ obsahuje jediný polynóm $r(x) \in F[x]$, stupňa $\deg(r(x)) < \deg(f(x))$. Tento polynóm sa dá vypočítať ako zvyšok po delení polynómu $g(x)$ polynómom $f(x)$ a nazýva sa reprezentantom triedy $[g(x)]$. Ukžeme ešte, že v každej triede rozkladu sa nachádza jediný polynóm stupňa $< \deg(f(x))$. Predpokladajme opak, t.j. nech $r_1(x), r(x) \in [g(x)]$ sú dva polynómy stupňa $< \deg(f(x))$, ktoré patria do tej istej triedy. Ale potom platí $f(x)|r(x) - r_1(x)$ a $\deg(r(x) - r_1(x)) < \deg(f(x))$. To znamená, že $r(x) - r_1(x) = 0$ a $r(x) = r_1(x)$. Keďže každá trieda rozkladu obsahuje jediný polynóm stupňa $< \deg(f(x))$, môžeme explicitne charakterizovať triedy rozkladu $F[x]/(f(x))$: sú to množiny polynómov $r(x)+(f(x))$, kde $r(x) \in F[x]$ a $\deg(r(x)) < \deg(f(x))$. Ak na triedach z rozkladu $F[x]/(f(x))$ definujeme operácie sčítania a násobenia tradičným spôsobom; t.j. pre ľubovoľné $[a(x)], [b(x)] \in F[x]/(f(x))$ položíme

$$[a(x)] + [b(x)] = [a(x) + b(x)], \qquad [a(x)] \cdot [b(x)] = [a(x) \cdot b(x)],$$

dostávame okruh, ktorý budeme nazývať faktorovým okruhom polynómov nad poľom F podľa polynómu $f(x)$. Faktorové okruhy polynómov budeme v ďalšom využívať pri konštrukcii konečných polí.

Podobne ako v okruhu celých čísel, môžeme aj v okruhu polynómov nad poľom F zaviesť pojem najväčšieho spoločného deliteľa a najmenšieho spoločného násobku polynómov.

Definícia 15.3.3. *Nech je* $F[x]$ *okruh polynómov nad poľom* F *a nech sú* $f(x), g(x)$ *polynómy z okruhu* $F[x]$.

1. *Normovaný polynóm* $d(x) \in F[x]$ *nazveme najväčším spoločným deliteľom polynómov* $f(x), g(x)$*, ak* $d(x)|f(x)$*,* $d(x)|g(x)$ *a pre ľubovoľný polynóm* $h(x) \in F[x]$*, ktorý delí polynómy* $f(x), g(x)$ *platí* $h(x)|d(x)$*. Najväčší spoločný deliteľ polynómov* $f(x), g(x)$ *budeme označovať symbolom* $\gcd(f(x), g(x))$

2. *Normovaný polynóm* $a(x) \in F[x]$ *nazveme najmenším spoločným násobkom polynómov* $f(x), g(x)$ *(označenie* $\mathrm{lcm}(f(x), g(x))$*), ak* $f(x)|a(x)$*,* $g(x)|a(x)$ *a pre ľubovoľný polynóm* $b(x) \in F[x]$*, ktorý je deliteľný polynómami* $f(x), g(x)$ *platí, že* $a(x)|b(x)$.

Veta 15.3.3. *Nech sú* $f(x), g(x)$ *dva nenulové polynómy okruhu* $F[x]$. *Potom existujú také polynómy* $a(x), b(x)$, *že*

$$\gcd(f(x), g(x)) = a(x)f(x) + b(x)g(x).$$

Dôkaz. Uvažujme množinu polynómov $J = \{c_1(x)f(x) + c_2(x)g(x)\}$ $c_1(x), c_2(x) \in F[x]$. Je zrejmé, že J je ideál a že $J \neq (0)$. Okruh $F[x]$ je okruhom hlavných ideálov, a preto existuje $d(x) \in F[x]$, ktorý generuje ideál J. Vzhľadom na to, ako sú vyjadrené prvky ideálu J z toho, že $d(x) \in J$ vyplýva existencia polynómov $a(x), b(x)$ takých, že $d(x) = a(x)f(x)+b(x)g(x)$. Ukážeme ešte, že $d(x) = \gcd(f(x), g(x))$. Oba polynómy f(x),g(x) patria do J, a preto $d(x)|f(x)$ a $d(x)|g(x)$. Ak by existoval iný (normovaný) polynóm, $d_1(x)$ taký, že $(d_1(x)) = J$, $d_1(x)|d(x)$ a $d(x)|d_1(x)$, t.j. $d(x) = d_1(x)$. □

Najväčší spoločný deliteľ dvoch polynómov $f(x), g(x) \in F[x]$ možno vypočítať pomocou *Euklidovho algoritmu.* Predpokladajme kvôli jednoduchosti, že $g(x) \neq 0$ a že $g(x)$ nie je deliteľom polynómu $f(x)$, potom budeme postupne deliť:

$$\begin{aligned}
f(x) &= q_1(x)g(x) + r_1(x) \quad 0 \leq \deg(r_1(x)) < \deg(g(x)) \\
g(x) &= q_2(x)r_1(x) + r_2(x) \quad 0 \leq \deg(r_2(x)) < \deg(r_1(x)) \\
r_1(x) &= q_3(x)r_2(x) + r_3(x) \quad 0 \leq \deg(r_3(x)) < \deg(r_2(x)) \\
&\vdots \qquad\qquad\qquad \vdots \\
r_{s-2} &= q_s(x)r_{s-1}(x) + r_s(x) \quad 0 \leq \deg(r_s(x)) < \deg(r_{s-1}(x)) \\
r_{s-1} &= q_{s+1}(x)r_s(x).
\end{aligned}$$

V tejto postupnosti sú $q_1(x), \ldots, q_{s+1}(x); r_1(x) \ldots, r_s(x)$ polynómy okruhu $F[x]$. Kedže $\deg(g(x))$ je konečný a v každom kroku sa stupeň polynómu $r_i(x)$ zmenšuje, procedúra po konečnom počte krokov skončí. Nech má polynóm $r_s(x)$ vedúci koeficient a, potom najväčší spoločný deliteľ polynómov $f(x), g(x)$ vyjadríme nasledovne: $\gcd(f(x), g(x)) = a^{-1}r_s(x)$. Normované polynómy $f(x), g(x) \in F[x]$ nazveme nesúdeliteľnými (relatively prime), ak $\gcd(f(x), g(x) = 1$.

Dôležitú úlohu pri štúdiu vlastností okruhu polynómov $F[x]$ zohrávajú ireducibilné polynómy. Každý polynóm z $F[x]$ sa dá totiž jednoznačne vyjadriť ako súčin ireducibilných polynómov. Skôr ako formulujeme a dokážeme tento poznatok, využijeme vetu o deliteľnosti polynómov na ustanovenie vzťahu medzi koreňmi polynómu a deliteľnosťou polynómu.

Veta 15.3.4. *Nech je* $f(x)$ *ľubovoľný polynóm nad poľom* F *a nech je* c *ľubovoľný prvok poľa* F. *Potom polynóm* $(x - c)$ *delí polynóm* $f(x)$ *práve vtedy, ak je* c *koreňom polynómu* $f(x)$.

Dôkaz. Nech polynóm $(x - c)$ delí polynóm $f(x)$, potom existuje taký polynóm $f_1(x)$, že $f(x) = (x-c)f_1(x)$. Potom však $f(c) = (c-c)f_1(c) = 0$, a teda c je koreňom polynómu $f(x)$. Nech na druhej strane $f(c) = 0$; t.j. prvok c je koreňom polynómu $f(x)$. Podľa vety 15.3.1 sa polynóm $f(x)$ dá vyjadriť nasledovne:

$$f(x) = (x - c)q(x) + r(x),$$

pričom $\deg(r(x)) < \deg(x-c) = 1$. To však znamená, že $r(x)$ musí byť konštantný polynóm. Ale $f(c) = (c-c)q(c) + r(c) = r(c) = 0$, a teda $r(x) = 0$. □

Veta 15.3.5. *Nech sú* $f_1(x), \ldots, f_m(x) \in F[x]$*, nech je* $g(x) \in F[x]$ *ireducibilný polynóm. Potom platí: ak* $g(x)$ *delí súčin* $f_1(x) \cdot f_2(x) \ldots f_m(x)$*, tak potom* $g(x)$ *delí aspoň jeden z polynómov* $f_1(x), \ldots, f_m(x)$.

Dôkaz. Bez ujmy na všeobecnosti môžeme predpokladať, že polynómy $g(x), f_1(x), f_2(x), \ldots, f_m(x)$ sú normované. Ak je $g(x)$ ireducibilný polynóm, tak potom $\gcd(g(x), f_i(x)) = g(x)$ ak je polynóm $f_i(x)$ násobkom polynómu $g(x)$, v opačnom prípade $\gcd(g(x), f_i(x)) = 1$. Ak by totiž $\gcd(g(x), f_i(x)) = d(x) \notin \{1, g(x)\}$, potom by $d(x)|g(x)$, čo je v spore s ireducibilitou $g(x)$. Teda musí existovať i také, že $g(x)|f_i(x)$. □

Veta 15.3.6 (O jednoznačnej faktorizácii polynómov)**.** *Nech je* $f(x) \in F[x]$ *ľubovoľný polynóm stupňa* $\deg(f(x)) \geq 0$*. Potom sa* $f(x)$ *dá zapísať v tvare súčinu*

$$f(x) = a f_1(x)^{e_1} \cdots f_m(x)^{e_m}, \tag{15.5}$$

kde $a \in F$*,* $e_1, \ldots, e_m \in N$ *a* $f_1(x), \ldots, f_m(x)$ *sú navzájom rôzne normované ireducibilné polynómy z* $F[x]$*. Naviac, odhliadnuc od poradia činiteľov v rozklade 15.5, je rozklad polynómu* $f(x)$ *určený jednoznačne.*

Dôkaz budeme viesť matematickou indukciou vzhľadom na stupeň polynómu. Prípad $n = 1$ je triviálny, nakoľko polynómy stupňa 1 sú ireducibilné nad $F[x]$. Predpokladajme, že sa ľubovoľný polynóm stupňa menšieho ako n dá zapísať v tvare 15.5. Ukážeme, že aj polynóm stupňa n možno rozložiť na súčin ireducibilných polynómov v tvare 15.5. Ak je $f(x)$ ireducibilný polynóm, stačí ho normovať, t.j. vyjadriť v tvare $a^{-1}f(x)$, kde a je vedúci koeficient polynómu $f(x)$. Ak polynóm $f(x)$ nie je ireducibilný, možno ho vyjadriť v tvare súčinu aspoň dvoch polynómov; $f(x) = g_1(x)g_2(x)$. Oba polynómy $g_1(x), g_2(x)$ majú stupeň $1 \leq \deg(g_1(x)), \deg(g_2(x)) < n$, a preto ich podľa indukčného predpokladu možno vyjadriť v tvare 15.5.

Ostáva ukázať jednoznačnosť rozkladu 15.5. Predpokladajme, že existujú dva rozličné rozklady polynómu $f(x)$; t.j.

$$f(x) = a f_1(x)^{e_1} \cdots f_m(x)^{e_m} = b g_1(x)^{d_1} \cdots g_s(x)^{d_s}. \tag{15.6}$$

Vedúce koeficienty v rozličných vyjadreniach toho istého polynómu sa musia zhodovať, preto $a = b$. Zoberieme teraz napríklad polynóm $f_1(x)$. Keďže $f_1(x)$ delí polynóm $f(x)$, musí deliť aj $g_1(x)^{d_1} \cdots g_s(x)^{d_s}$. Ale $f(x)$ je ireducibilný polynóm, a potom podľa predchádzajúcej vety musí deliť niektorý z polynómov $g_j(x)$, napríklad $g_k(x)$. Ale aj $g_k(x)$ je ireducibilný nad $F[x]$, a teda $f_1(x) = c g_k(x)$, kde $c \in F$. Oba polynómy $f_1(x), g_k(x)$ sú normované, a teda $f_1(x) = g_k(x)$. Vydelíme rovnosť 15.6 polynómom $f_1(x) (= g_k(x))$ a analogickým spôsobom budeme riešiť novú identitu. Nakoľko v každom kroku sa zníži stupeň polynómov v identite, po konečnom počte iterácií dostaneme identitu $1 = 1$. Tým sme dokázali, že obe faktorizácie polynómu $f(x)$ sú, až na poradie činiteľov v súčine, identické. □

15.4 Konečné polia

Pole, ako sme uviedli v definícii 15.2.2, je okruh, ktorého množina nenulových prvkov tvorí komutatívnu grupu vzhľadom na multiplikatívnu operáciu. Príkladmi polí sú množiny komplexných, reálnych a racionálnych čísel s operáciami sčítania a násobenia. Na druhej strane, celé čísla tvoria okruh (dokonca obor integrity), ale nie pole. Spomínané polia sú nekonečné. V teórii kódovania pracujeme a konečnými množinami, a preto budeme využívať polia s konečným počtom prvkov — konečné polia. Uvedieme najprv konečné polia založené na okruhu celých čísel a potom sa budeme zaoberať konečnými poľami vychádzajúcimi z okruhu polynómov.

Pripomenieme, že $(Z, +, \cdot)$ je okruh (obor integrity) a $(Z/Z_q, +, \cdot)$ je faktorový okruh, ktorého prvkami sú triedy rozkladu $[0], \ldots, [q-1]$. Kým okruh $(Z, +, \cdot)$ nemohol byť poľom (s výnimkou 1 a -1 inverzné prvky k celým číslam nie sú celé čísla), faktorový okruh $(Z/Z_q, +, \cdot)$ za istých podmienok môže byť poľom .

Veta 15.4.1. *Faktorový okruh* $(Z, +, \cdot)$ *je poľom práve vtedy, ak je* q *prvočíslo.*

Dôkaz. Nech je q prvočíslo. Potrebujeme ukázať, že ku každému nenulovému prvku okruhu $(Z/Z_q, +, \cdot)$ existuje v tomto okruhu inverzný prvok. Pripomenieme najprv, že nulovým prvkom okruhu $(Z/Z_q, +, \cdot)$ je $[0]$ a jednotkovým trieda $[1]$. Nech je s celé číslo, $s \in \{1, \ldots, q-1\}$. Keďže q je prvočíslo, platí $\gcd(s, q) = 1$ a teda existujú také dve celé čísla a, b, že

$$aq + bs = \gcd(s, q) = 1.$$

To znamená, že

$$[1] = [aq + bs] = [aq] + [bs] = [0] + [bs] = [bs] = [b][s],$$

a teda $[b]$ je inverzným prvkom k prvku $[s]$ a faktorový okruh $(Z/Z_q, +, \cdot)$ je poľom.

Na druhej strane, predpokladajme, že okruh $(Z/Z_q, +, \cdot)$ je pole, ale q nie je prvočíslo, t.j. q je zložené číslo a dá sa zapísať ako súčin čísel $q = q_1 q_2$, kde $1 < q_1, q_2 < q$. Kedže $(Z/Z_q, +, \cdot)$ je pole, k nenulovému prvku $[q_1]$ existuje v poli inverzný prvok, $[q_1^{-1}]$. Potom platí

$$[0] \neq [q_2] = [q_1][q_1^{-1}][q_2] = [(q_1 \cdot q_1^{-1}) \cdot q_2] = q_1^{-1} \cdot (q_1 \cdot q_2)] = [q_1^{-1} \cdot q] = [0],$$

spor. □

Zjednodušíme trocha výpočty v poli $(Z/Z_q, +, \cdot)$ zavedením vhodnejšej reprezentácie. Uvažujme množinu celých čísel $F_q = \{0, \ldots, q-1\}$ a definujeme operácie sčítania $\oplus$ a násobenia $\otimes$ prvkov z F_q nasledovne ($a, b \in F_q$):

$$a \oplus b = (a + b) \mod q, \qquad a \otimes b = ab \mod q.$$

Definujeme teraz zobrazenie $\varphi : Z/Z_q \to F_q$; $\varphi([a]) = a \mod q$. Je zrejmé, že φ je bijekcia. Ukážeme, že je aj homomorfizmus:

$$\begin{aligned} \varphi([a] + [b]) &= \varphi([a + b]) = a + b \mod q = a \oplus b = \varphi([a]) \oplus \varphi([b]) \\ \varphi([a][b]) &= \varphi([a \cdot b]) = a \cdot b \mod q = a \otimes b = \varphi([a]) \otimes \varphi([b]). \end{aligned}$$

Z vyššie uvedeného vyplýva, že φ je izomorfizmus, $(F_q, \oplus, \otimes)$ je konečné pole a polia $(Z/Z_q, +, \cdot)$ a $(F_q, \oplus, \otimes)$ sú izomorfné. Pole $(F_q, \oplus, \otimes)$ budeme nazývať *Galoisovým poľom* a označovať symbolom $\mathbf{GF}(q)$. Tam, kde to nepovedie k nedorozumeniu, budeme operácie poľa $\mathbf{GF}(q)$ označovať štandardným spôsobom — ako $+$ a $\cdot$.

Podobne ako sme zaviedli konečné polia $\mathbf{GF}(q)$ pomocou okruhu celých čísel, zavedieme teraz rozšírenia konečných polí pomocou okruhu polynómov.

Veta 15.4.2. *Nech je* $F[x]$ *okruh polynómov nad poľom* F, $f(x) \in F[x]$ *a* $F[x]/(f(x))$ *je faktorový okruh polynómov nad poľom* F. *Potom* $F[x]/(f(x))$ *je poľom práve vtedy, ak je polynóm* $f(x)$ *ireducibilný polynóm okruhu* $F[x]$.

Dôkaz. Predpokladajme, že polynóm $f(x)$ je ireducibilný polynóm okruhu $F[x]$. Ukážeme, že k ľubovoľnému nenulovému prvku faktorového okruhu $F[x]/(f(x))$ existuje v tomto okruhu inverzný prvok; t.j.

$$\forall(g(x)) \in F[x]/(f(x)); (g(x)) \neq (0)\exists(h(x)) \in F[x]/(f(x)); \quad (f(x))(h(x)) = (1)$$

Nech $(g(x)) \in F[x]/(f(x))$ je ľubovoľný nenulový prvok faktorového okruhu. Bez ujmy na všeobecnosti môžeme predpokladať, že $\deg(g(x)) < \deg(f(x))$. Keďže polynóm $f(x)$ je ireducibilný polynóm nad poľom F, $\gcd(f(x), g(x)) = 1$ a existujú také dva polynómy $a(x), b(x) \in F[x]$, že

$$a(x)f(x) + b(x)g(x) = \gcd(f(x), g(x)) = 1.$$

To znamená, že

$$(a(x)f(x) + b(x)g(x)) = (a(x)f(x)) + (b(x)g(x)) = (0) + (b(x))(g(x)) = (b(x))(g(x)) = (1)$$

a prvok $(b(x))$ je inverzným prvkom prvku $(g(x))$ faktorového okruhu $F[x]/(f(x))$.

Opačne, nech faktorový okruhu $F[x]/(f(x))$ je pole a nech polynóm $f(x)$ je reducibilný polynóm okruhu $F[x]$.To znamená, že v okruhu $F[x]$ existujú polynómy $f_1(x)$ a $f_2(x)$ také, že $f(x) = f_1(x)f_2(x)$ a $0 < \deg(f_1(x)), \deg(f_2(x)) < \deg(f(x))$. Keďže $F[x]/(f(x))$ je pole k prvku $(f_1(x)) \in F[x]/(f(x))$ existuje v tomto poli inverzný prvok, trieda $(f_1(x)^{-1})$. Potom platí

$$\begin{aligned}(0) &\neq (f_2(x)) = (1)(f_2(x)) = (f_1(x)^{-1})(f_1(x))(f_2(x)) = (f_1(x)^{-1}f_1(x)f_2(x)) = \\ &= (f_1(x)^{-1}f(x)) = (f_1(x)^{-1})(0) = (0)\end{aligned}$$

spor. □

Z predchádzjúcej vety vyplýva, že ak je dané konečné pole $\mathbf{GF}(q)$ a polynóm $f(x) \in \mathbf{GF}(q)[x]$ ireducibilný v okruhu polynómov $\mathbf{GF}(q)[x]$, tak môžeme skonštruovať konečné pole $\mathbf{GF}(q)[x]/(f(x))$, ktoré bude mať rád $q^{\deg(f(x))}$. Otvorenou zostáva otázka, či existujú ireducibilné polynómy potrebných stupňov. Skôr, ako sa budeme zaoberať týmito problémami, ilustrujeme na príklade konštrukciu konečného poľa pomocou faktorového okruhu polynómov.

Príklad. Pri konštrukcii tzv. BCH kódov budeme využívať konečné pole $\mathbf{GF}(2^4)$. Pri jeho konštrukcii budeme vychádzť z binárneho konečného poľa $\mathbf{GF}(2)$. Potrebujeme nájsť

No	stupeň	polynóm	rozklad	poznámka
1	0	0	0	prvok poľa
2	0	1	1	prvok poľa
3	1	x	x	ireducibilný
4	1	$x+1$	$x+1$	ireducibilný
5	2	x^2	$x \cdot x$	reducibilný
6	2	x^2+1	$(x+1)^2$	reducibilný
7	2	x^2+x	$x \cdot (x+1)$	reducibilný
8	2	x^2+x+1	x^2+x+1	ireducibilný
9	3	x^3	$x \cdot x \cdot x$	reducibilný
10	3	x^3+1	$(x+1)(x^2+x+1)$	reducibilný
11	3	x^3+x	$x(x+1)^2$	reducibilný
12	3	x^3+x^2	$x^2(x+1)$	reducibilný
13	3	x^3+x+1	x^3+x+1	ireducibilný
14	3	x^3+x^2+1	x^3+x^2+1	ireducibilný
15	3	x^3+x^2+x	$x \cdot (x^2+x+1)$	reducibilný
16	3	x^3+x^2+x+1	$(x+1)^3$	reducibilný

Tabuľka 15.1: Polynómy stupňa $0, 1, 2, 3$ nad poľom $\mathbf{GF}(2)$

ireducibilný polynóm stupňa 4 nad poľom $\mathbf{GF}(2)$. Polynóm stupňa 4 nad binárnym poľom má tvar

$$a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4, \qquad a_i \in \{0,1\}, \quad i = 0,\ldots,4.$$

Aby mal polynóm požadovaný stupeň (4), $a_4 = 1$ a na výber ostatných 4 koeficientov zostáva $2^4 = 16$ možností. Aby bol hľadaný polynóm ireducibilný, nesmie byť deliteľný iným polynómom nižšieho stupňa. To znamená, že stačí overiť, či je daný polynóm deliteľný ireducibilnými polynómami stupňa 1 a 2. V tabuľke 15.1 uvádzame polynómy stupňa 3 a menšieho nad poľom $\mathbf{GF}(2)$.

Preveríme teraz 16 polynómov stupňa 4 nad poľom $\mathbf{GF}(2)$ na deliteľnosť ireducibilnými polynómami stupňa $0, 1, 2, 3$. Výsledky sú uvedené v tabuľke 15.2.

Okruh polynómov $\mathbf{GF}(2)[x]$ budeme faktorizovať pomocou ireducibilného polynómu $f(x) = x^4 + x + 1$. Prvkami poľa $\mathbf{GF}(2)[x]/x^4+x+1$ sú triedy rozkladu okruhu polynómov $\mathbf{GF}(2)[x]$ podľa ireducibilného polynómu $f(x)$. V jednej triede rozkladu sú tie polynómy okruhu polynómov $\mathbf{GF}(2)[x]$, ktorých rozdiel je deliteľný polynómom $f(x)$. Je zrejmé, že v každej triede rozkladu existuje práve jeden polynóm stupňa 3 alebo menšieho, ktorý budeme nazývať predstaviteľom triedy. Keďže polynómov okruhu $\mathbf{GF}(2)[x]$ stupňa 3 a menšieho je 16, pole $\mathbf{GF}(2)[x]/x^4 + x + 1$ obsahuje 16 prvkov. V tabuľke 15.3 uvádzame prvky poľa $\mathbf{GF}(2)[x]/x^4+x+1$; prvok (trieda rozkladu) je reprezentovaný predstaviteľom triedy.

Kvôli lepšiemu prehľadu zhrnieme najdôležitejšie poznatky o konečných poliach v nasledujúcej tabuľke [2]. Časť z nich sme už dokázali, dokazovaním ostatných poznatkov sa budeme zaoberať.

1. Rád (počet prvkov) ľubovoľného konečného poľa je mocninou prvočísla.

No	stupeň	polynóm	rozklad	poznámka
1	4	x^4	x^4	R
2	4	x^4+1	$(x+1)^4$	R
3	4	x^4+x	$x\cdot(x+1)(x^2+x+1)$	R
4	4	x^4+x+1	x^4+x+1	I
5	4	x^4+x^2	$x^2\cdot(x+1)^2$	R
6	4	x^4+x^2+1	$(x^2+x+1)^2$	R
7	4	x^4+x^2+x	$x\cdot(x^3+x+1)$	R
8	4	x^4+x^2+x+1	$(x+1)\cdot(x^3+x^2+1)$	R
9	4	x^4+x^3	$x\cdot(x^3+x^2+1)$	R
10	4	x^4+x^3+1	x^4+x^3+1	I
11	4	x^4+x^3+x	$x\cdot(x^3+x^2+1)$	R
12	4	$x^4+x^3+x^2$	$x^2(x^2+x+1)$	R
13	4	x^4+x^3+x+1	$(x+1)^2\cdot(x^2+x+1)$	R
14	4	$x^4+x^3+x^2+1$	$(x+1)\cdot(x^3+x+1)$	R
15	4	$x^4+x^3+x^2+x$	$x\cdot(x+1)^3$	R
16	4	$x^4+x^3+x^2+x+1$	$x^4+x^3+x^2+x+1$	I

Tabuľka 15.2: Polynómy stupňa 4 nad poľom $\mathbf{GF}(2)$

1	(x)
2	(x^2)
3	(x^3)
4	$(x+1)$
5	(x^2+x)
6	(x^3+x^2)
7	(x^3+x+1)
8	(x^2+1)
9	(x^3+x)
10	(x^2+x+1)
11	(x^3+x^2+x)
12	(x^3+x^2+x+1)
13	(x^3+x^2+1)
14	(x^3+1)
15	(1)
16	(0)

Tabuľka 15.3: Prvky poľa $\mathbf{GF}(2)[x]/x^4+x+1$

2. Pre ľubovoľné prvočíslo p a celé kladné číslo m je najmenším podpoľom poľa $\mathbf{GF}(p^m)$ pole $\mathbf{GF}(p)$. Prvky poľa $\mathbf{GF}(p)$ sa nazývajú *celými číslami* poľa $\mathbf{GF}(p^m)$ a číslo p jeho charakteristikou

3. V konečnom poli charakteristiky 2 pre ľubovoľný prvok poľa β platí $\beta = -\beta$.

4. Pre ľubovoľné prvočíslo p a celé kladné číslo m existuje konečné pole s p^m prvkami.

5. Každé konečné pole $\mathbf{GF}(q)$ obsahuje aspoň jeden primitívny prvok.

6. Nad každým konečným poľom existuje pre ľubovoľné kladné celé číslo m primitívny polynóm stupňa m.

7. Každý primitívny prvok poľa $\mathbf{GF}(q)$ má nad ľubovoľným podpoľom poľa $\mathbf{GF}(q)$ ireducibilný minimálny polynóm.

8. Dve konečné polia s tým istým počtom prvkov sú izomorfné.

9. Pre ľubovoľné q, ktoré je mocninou prvočísla a ľubovoľné celé kladné číslo m je pole $\mathbf{GF}(q)$ podpoľom poľa $\mathbf{GF}(q^m)$ a pole $\mathbf{GF}(q^m)$ je rozšírením poľa $\mathbf{GF}(q)$.

10. Ak číslo n nie je deliteľom čísla m, tak pole $\mathbf{GF}(q^n)$ nie je podpoľom poľa $\mathbf{GF}(q^m)$.

11. Pre ľubovoľný prvok poľa $\mathbf{GF}(q^m)$ stupeň jeho minimálneho polynómu nad $\mathbf{GF}(q)$ delí m.

Konečné pole predstavuje aditívnu abelovskú grupu a množina jeho nenulových prvkov je multiplikatívna abelovská grupa. V ďalšom budeme pracovať s multiplikatívnou grupou konečného poľa.

Veta 15.4.3. *Nech je* $\mathbf{GF}(q)$ *konečné pole a* $\beta_1, \ldots, \beta_{q-1}$ *sú jeho nenulové prvky. Potom platí*

$$x^{q-1} - 1 = (x - \beta_1)(x - \beta_2) \ldots (x - \beta_{q-1}).$$

Dôkaz. Stačí ukázať, že ľubovoľný nenulový prvok poľa $\mathbf{GF}(q)$ je koreňom polynómu $x^{q-1} - 1$. Uvažujme prvok β. Jeho mocniny $\beta, \beta^2, \ldots, \beta^h = 1$ tvoria podgrupu multiplikatívnej grupy poľa $\mathbf{GF}(q)$. Rád podgrupy, generovanej prvkom β delí rád multiplikatívnej grupy poľa $\mathbf{GF}(q)$; $h|(q-1)$. To znamená, že existuje kladné celé číslo k také, že $hk = (q-1)$. Potom však $\beta^q = \beta^{hk} = (\beta^h)^k = 1^k = 1$, a teda β je koreňom polynómu $x^{q-1} - 1$. □

Reprezentácia konečných polí pomocou tried polynómov bola trocha neprehľadná. Ukážeme, že multiplikatívna grupa konečného poľa je cyklická a budeme reprezentovať (nenulové) prvky konečného poľa mocninami generátora jeho cyklickej multiplikatívnej grupy.

Veta 15.4.4. *Multiplikatívna grupa konečného poľa* $\mathbf{GF}(q)$ *je cyklická.*

Dôkaz. Budeme postupovať podľa [2]. Multiplikatívna grupa konečného poľa $\mathbf{GF}(q)$ má rád $q-1$. Ak by bolo číslo $q-1$ prvočíslo (napr. $3,7,31,127,\dots$), tvrdenie vety by bolo triviálne. Podľa predchádzajúcej vety musí rád každého nenulového prvku deliť $q-1$. To znamená, že nenulové prvky majú buď rád 1 alebo rád $q-1$. Jednotkový prvok poľa $\mathbf{GF}(q)$ má rád 1 a všetky ostatné nenulové prvky majú rád $q-1$ a teda sú generátormi multiplikatívnej grupy konečného poľa $\mathbf{GF}(q)$.

Nech je číslo $q-1$ zložené. Potom ho možno jednoznačne rozložiť na súčin prvočísel:

$$q-1 = p_1^{\nu_1} \cdots p_s^{\nu_s}$$

Polynóm $x^{(q-1)/p_i}-1$ môže mať najviac $(q-1)/p_i$ koreňov, to znamená, že v poli $\mathbf{GF}(q)$ existuje nenulový prvok, ktorý **nie je** koreňom polynómu $x^{(q-1)/p_i}-1$. Označíme tento prvok symbolom a_i. Je zrejmé, že pre ľubovoľné i, $i=1,\dots,s$ existuje nenulový prvok a_i poľa $\mathbf{GF}(q)$ taký, že $a_i^{(q-1)/p_i} \neq 1$. Na základe prvkov a_i zostrojíme teraz prvky b_i a b poľa $\mathbf{GF}(q)$:

$$b_i = a_i^{(q-1)/p_i^{\nu_i}} \qquad \text{a} \quad b = b_1 b_2 \dots b_s$$

a ukážeme, že rád prvku b je $q-1$; t.j. že b je generátor multiplikatívnej grupy poľa $\mathbf{GF}(q)$ a tým aj to, že táto grupa je cyklická.

Najprv ukážeme, že rád prvku b_i je $p_i^{\nu_i}$. Platí

$$b_i^{p_i^{\nu_i}} = \left(a_i^{(q-1)/p_i^{\nu_i}}\right)^{p_i^{\nu_i}} = a_i^{(q-1)} = 1.$$

To znamená, že rád prvku b_i delí $p_i^{\nu_i}$, t.j. má tvar $p_i^{n_i}$, pričom $n_i \leq \nu_i$. Predpokladajme, že $n_i < \nu_i$. Potom by aj

$$b_i^{p_i^{\nu_i-1}} = b_i^{p_i^{n_i} \cdot p_i^{\nu_i-n_i}} = \left(b_i^{p_i^{n_i}}\right)^{p_i^{\nu_i-n_i}} = (1)^{p_i^{\nu_i-n_i}} = 1.$$

Ale

$$b_i^{p_i^{\nu_i-1}} = \left(a_i^{(q-1)/p_i^{\nu_i}}\right)^{p_i^{\nu_i-1}} = a_i^{(q-1)/p_i} \neq 1.$$

To znamená, že $n_i = \nu_i$. Teraz ukážeme, že rád prvku b sa rovná $q-1$. Predpokladajme, že rád prvku b je n; t.j. $b^n = 1$. Pre ľubovoľné $i = 1,\dots,s$ platí

$$b^{n \cdot p_1^{\nu_1} \cdots p_s^{\nu_s}/p_i^{\nu_i}} = 1.$$

Vyjadríme teraz prvok b pomocou prvkov b_i a využijeme to, že $b_i^{p_i^{\nu_i}} = 1$:

$$\begin{aligned}
& (b_1 b_2 \dots b_s)^{n \cdot p_1^{\nu_1} \cdots p_s^{\nu_s}/p_i^{\nu_i}} = (b_1)^{n \cdot p_1^{\nu_1} \cdots p_s^{\nu_s}/p_i^{\nu_i}} \dots (b_{i-1})^{p_{i-1}^{\nu_{i-1}} n \cdot p_1^{\nu_1} \cdots p_s^{\nu_s}/(p_{i-1}^{\nu_{i-1}} \cdot p_i^{\nu_i})} \cdot \\
\cdot\ & (b_i)^{n \cdot p_1^{\nu_1} \cdots p_s^{\nu_s}/p_i^{\nu_i}} (b_{i+1})^{p_{i+1}^{\nu_{i+1}} n \cdot p_1^{\nu_1} \cdots p_s^{\nu_s}/(p_{i+1}^{\nu_{i+1}} \cdot p_i^{\nu_i})} \dots (b_s)^{p_s^{\nu_s} n \cdot p_1^{\nu_1} \cdots p_{s-1}^{\nu_{s-1}}/p_i^{\nu_i}} = \\
= \ & (1)^{n \cdot p_2^{\nu_2} \cdots p_s^{\nu_s}/p_i^{\nu_i}} \dots (1)^{n \cdot p_1^{\nu_1} \cdots p_s^{\nu_s}/p_i^{\nu_i} p_{i-1}^{\nu_{i-1}}} \cdot (b_i)^{n \cdot p_1^{\nu_1} \cdots p_s^{\nu_s}/p_i^{\nu_i}} (1)^{n \cdot p_1^{\nu_1} \cdots p_s^{\nu_s}/(p_{i+1}^{\nu_{i+1}} \cdot p_i^{\nu_i})} \dots \\
& \dots (1)^{n \cdot p_1^{\nu_1} \cdots p_{s-1}^{\nu_{s-1}}/(p_i^{\nu_i})} = (b_i)^{n \cdot p_1^{\nu_1} \cdots p_s^{\nu_s}/(p_i^{\nu_i})} = 1.
\end{aligned}$$

Z toho, že rád prvku b_i je $p_i^{\nu_i}$ a z poslednej rovnosti vyplýva, že, že $p_i^{\nu_i}$ delí n pre $i, \ldots, s$. Čísla $p_i^{\nu_i}$ sú však navzájom nesúdeliteľné, a to znamená, že

$$n = p_1^{\nu_1} \cdots p_s^{\nu_s} = q - 1.$$

□

Prvok b rádu $q-1$, ktorého existenciu sme dokázali v predchádzajúcej vete, je primitívnym prvkom poľa $\mathbf{GF}(q)$. Tým sme zároveň dokázali nasledujúce dôležité tvrdenie.

Dôsledok 2. *V každom konečnom poli existuje primitívny prvok.*

Vrátime sa k poľu $GF(2^4) = GF(2)[x]/x^4 + x + 1$ z príkladu 15.4 a nájdeme jeho primitívny prvok. Pole $GF(2^4)$ sme zostrojili faktorizáciou okruhu polynómov $GF(2)[x]$ pomocou polynómu $f(x) = x^4 + x + 1$. Uvažujme teraz prvok (x) poľa $GF(2^4)$; (x) predstavuje triedu polynómov z okruhu $GF(2)[x]$, ktoré po delení polynómom $f(x)$ dávajú zvyšok x. Dosadíme prvok (x) do polynómu $f(x)$. Vzhľadom na uzavretosť poľa $GF(2^4)$ na sčítanie a násobenie dostaneme opäť prvok poľa $GF(2^4)$. Pripomenieme ešte, že pre ľubovoľné $a(x), b(x) \in GF(2)[x]$ platí

$$(a(x)) + (b(x)) = (a(x) + b(x)), \quad (a(x))(b(x)) = (a(x)b(x)).$$

Postupne dostávame

$$f((x)) = (x)^4 + (x) + 1 = (x^4) + (x + 1) = (x^4 + x + 1) = (0).$$

To znamená, že prvok (x) je koreňom polynómu $x^4 + x + 1$. Označíme prvok (x) symbolom α a ukážeme, že (zhodou okolností) je α primitívnym prvkom poľa $GF(2^4)$. Vyjadrenie prvkov poľa $GF(2^4)$ v podobe mocnín primitívneho prvku α je uvedené v tabuľke 15.4

V prvom stĺpci tabuľky 15.4 je exponent mocniny primitívneho prvku α, v druhom je uvedená mocnina α^i, v treťom je binárny kód príslušného prvku a vo štvrtom je prvok α^i vyjadrený v podobe lineárnej kombinácie mocnín prvkov $\alpha^3, \alpha^2, \alpha^1, \alpha^0$. Posledný stĺpec si zasluhuje vysvetlenie. Zápis prvkov konečného poľa pomocou mocnín primitívneho prvku umožňuje jednoducho realizovať násobenie prvkov poľa. Pre ľubovoľné dva prvky poľa $\alpha^i, \alpha^j \in GF(2^4)$ platí $\alpha^i\alpha^j = \alpha^{i+j} = \alpha^{(i+j) \bmod 15}$. Na druhej strane, aj keď je pole $GF(2^4)$ aditívna abelovská grupa, určiť prvok, ktorý predstavuje súčet $\alpha^i + \alpha^j$ nie je pri tejto reprezentácii prvkov poľa $GF(2^4)$jednoduché.

Ukážeme, že sa každý prvok poľa $GF(2^4)$ dá zapísať pomocou lineárnej kombinácie prvkov $\alpha^3, \alpha^2, \alpha^1, \alpha^0$ jednoznačným spôsobom. Nulový prvok poľa sa dá zapísať v podobe lineárnej kombinácia s nulovými koeficientami. Vieme, že $1 = \alpha^0$ a predpokladajme, že sa všetky mocniny $\alpha^i, \quad 0 \leq i \leq n$ dajú vyjadriť v tvare lineárnej kombinácie prvkov $\alpha^3, \alpha^2, \alpha^1, \alpha^0$. Ukážeme, že potom takto dá vyjadriť aj prvok α^{n+1}. Nech $\alpha^n = a_3\alpha^3 + a_2\alpha^2 + a_1\alpha^1 + a_0$. Potom

$$\alpha^{n+1} = \alpha^n \cdot \alpha = a_3\alpha^4 + a_2\alpha^3 + a_1\alpha^2 + a_0\alpha.$$

Keďže α je koreňom polynómu $x^4 + x + 1$, platí $\alpha^4 + \alpha + 1 = 0$ a (keďže pole $GF(2^4)$ má charakteristiku 2, a teda $-\alpha^4 = \alpha^4$)platí $\alpha^4 = \alpha + 1$. Využijeme tento vzťah a upravíme lineárnu kombináciu pre α^{n+1}:

$$\alpha^{n+1} = a_2\alpha^3 + a_1\alpha^2 + a_0\alpha + a_3(\alpha + 1) = a_2\alpha^3 + a_1\alpha^2 + (a_0 + a_3)\alpha + a_3.$$

0	α^0	0001				1
1	α^1	0010			α	
2	α^2	0100		α^2		
3	α^3	1000	α^3			
4	α^4	0011			$+\alpha$	$+1$
5	α^5	0110		α^2	$+\alpha$	
6	α^6	1100	α^3	$+\alpha^2$		
7	α^7	1011	α^3		$+\alpha$	$+1$
8	α^8	0101		α^2		$+1$
9	α^9	1010	α^3		$+\alpha$	
10	α^{10}	0111		α^2	$+\alpha$	$+1$
11	α^{11}	1110	α^3	$+\alpha^2$	$+\alpha$	
12	α^{12}	1111	α^3	$+\alpha^2$	$+\alpha$	$+1$
13	α^{13}	1101	α^3	$+\alpha^2$		$+1$
14	α^{14}	1001	α^3			$+1$
15	α^{15}	0001				1

Tabuľka 15.4: Reprezentácia nenulových prvkov $GF(2^4)$

Ostáva ešte ukázať, že vyjadrenie α^i je jednoznačné. Predpokladajme opak, t.j.

$$\alpha^i = a_3\alpha^3 + a_2\alpha^2 + a_1\alpha^1 + a_0 = b_3\alpha^3 + b_2\alpha^2 + b_1\alpha^1 + b_0.$$

Potom však

$$\begin{aligned} 0 &= a_3\alpha^3 + a_2\alpha^2 + a_1\alpha^1 + a_0 - b_3\alpha^3 + b_2\alpha^2 + b_1\alpha^1 + b_0 = \\ &= (a_3 - b_3)\alpha^3 + (a_2 - b_2)\alpha^2 + (a_1 - b_1)\alpha^1 + (a_0 - b_0). \end{aligned}$$

Z poslednej rovnosti vyplýva, že $a_3 = b_3, a_2 = b_2, a_1 = b_1, a_0 = b_0$, a teda vyjadrenie prvku poľa v podobe lineárnej kombinácie mocnín primitívneho prvku je jednoznačné.

Poznámka. Ak budeme reprezentovať prvky poľa $GF(2^4)$ pomocou 4-bitových celých čísel, výpočet α^{i+1} možno realizovať nasledovne:

if $(a > 7)$ then	:	$a_3 = 1$
$a = (a << 1) + 3;$	:	$a = a_2a_1a_00 + 0011 = a_2a_1\overline{a_0}1$
else	:	$a_3 = 0$
$a = (a << 1);$	:	$a = a_2a_1a_00$

Zovšeobecnením predchádzajúcej konštrukcie dokážeme nasledujúcu vetu.

Veta 15.4.5. *Nech je* $GF(q^m)$ *ľubovoľné konečné pole, nech je* α *primitívny prvok tohto poľa. Potom ľubovoľný prvok* β *poľa* $GF(q^m)$ *možno jednoznačným spôsobom vyjadriť v tvare*

$$\beta = a_{m-1}\alpha^{m-1} + \cdots + a_1\alpha + a_0, \tag{15.7}$$

kde $a_0 \ldots, a_{m-1} \in GF(q)$.

Na konštrukciu konečného poľa nám stačil ľubovoľný ireducibilný polynóm. V príklade, ktorý sme uviedli, bol koreň ireducibilného polynómu použitého na vytvorenie poľa zároveň aj primitívnym prvkom zostrojeného poľa. Takýto polynóm sa nazýva primitívnym polynómom.

Definícia 15.4.1. *Nech je dané konečné pole* $GF(q)$ *a ireducibilný polynóm* $f(x)$ *nad týmto poľom. Polynóm* $f(x)$ *sa nazýva primitívnym polynómom nad poľom* $GF(q)$*, ak koreňomj polynómu* $f(x)$ *v rozšírení poľa* $GF(q)$*, poli* $GF(q)[x]/f(x)$ *je primitívny prvok poľa* $GF(q)[x]/f(x)$.

Pripomíname, že nie každý ireducibilný polynóm $f(x)$ nad poľom $GF(q)$ je zároveň primitívnym polynómom. Na druhej strane, pre ľubovoľné konečné pole $GF(q)$ existujú primitívne polynómy (nad poľom $GF(q)$) ľubovoľného stupňa.

Príklad. Uvažujme ireducibilný polynóm $f(x) = x^4 + x^3 + x^2 + x + 1$ nad poľom $GF(2)$. Tento polynóm má v poli $GF(2)[x]/x^4+x^3+x^2+x+1$ korene $(x) = \beta; \beta^2, \beta^3, \beta^4$. Využijeme vzťah $\beta^4 + \beta^3 + \beta^2 + \beta + 1 = 0$ a vyjadríme β^4 pomocou lineárnej kombinácie nižších mocnín prvku β: $\beta^4 = \beta^3 + \beta^2 + \beta + 1$. Jednotlivé mocniny sú uvedené v tabuľke 15.5.

0	β^0	0001				1
1	β^1	0010			β	
2	β^2	0100		β^2		
3	β^3	1000	β^3			
4	β^4	1111	β^3	$+\beta^2$	$+\beta$	$+1$
5	β^5	0001				1

Tabuľka 15.5: Mocniny prvku β poľa $GF(2^4)$

Prvok β má teda rád 5. (Pomocou tabuľky 15.5 ľahko dokážeme, že rád 5 majú aj ostatné korene polynómu $f(x)$, prvky $\beta^2, \beta^3, \beta^4$). Ireducibilný polynóm $x^4 + x^3 + x^2 + x + 1$ nad poľom $GF(2)$ teda nie je primitívny.

V ďalšej časti budeme pokračovať v skúmaní základných vlastností konečných polí. Vieme, že charakteristika konečného poľa je prvočíslo. V akom vzťahu je pole $GF(p^m)$ s charakteristikou p s konečným poľom $GF(p)$?

Veta 15.4.6. *Nech je* $GF(q)$ *ľubovoľné konečné pole, potom* $GF(q)$ *obsahuje jediné konečné pole, ktorého rád je charakteristikou poľa* $GF(q)$.

Dôkaz. Nech je 1 jednotka konečného poľa $GF(q)$. Sumu $\underbrace{1 + 1 + \cdots + 1}_{n}$ označíme symbolom n a zostrojíme postupnosť

$$0, 1, 2, 3 \ldots, p - 1, p = 0.$$

Táto postupnosť je konečná (lebo $GF(q)$ je konečné pole) jej prvky tvoria aditívnu cyklickú grupu G. Sčítanie prvkov je v grupe G definované ako súčet celých čísel $\mod p$. Využijeme to, že v poli $GF(q)$ platí distributívny zákon a zavedieme násobenie v grupe G pomocou sčítania $\mod p$ nasledovne ($\alpha, \beta \in G$):

1. $\alpha \cdot 1 = \alpha$,
2. $\alpha \cdot (\beta + 1) = \alpha \cdot \beta + \alpha$.

dokončiť podľa Niederreitera

□

V tabuľkách 15.1 a 15.2 sme uviedli ireducibilné polynómy stupňa $2, 3, 4$ nad poľom $GF(2)$. V poli $GF(2^4)$ sa však tieto ireducibiné polynómy dali rozložiť na súčin lineárnych činiteľov 15.6. Tento príklad ilustruje skutočnosť, že ireducibilné polynómy nad poľom $GF(q)$ môžu byť reducibilné nad vhodným rozšírením poľa $GF(q)$. Pri konštrukcii cyklických kódov budeme potrebovať zostrojiť polynóm nad nejakým konečným poľom s predpísanými koreňmi; pozrieme sa preto na takéto polynómy podrobnejšie. Zavedieme najprv jeden dôležitý pojem.

Definícia 15.4.2. *Nech je* $GF(q)$ *konečné pole a* $GF(Q)$ *je jeho rozšírenie; nech* $\alpha \in GF(Q)$. *Normovaný polynóm* $m_\alpha(x)$ *nad* $GF(q)$ *budeme nazývať minimálnym polynómom prvku* α *nad poľom* $GF(q)$*, ak platí*

1. $m_\alpha(\alpha) = 0$,
2. *ak existuje polynóm* $a(x)$ *nad* $GF(q)$ *taký, že* $a(\alpha) = 0$*, tak potom* $m_\alpha(x) | a(x)$.

Minimálny polynóm prvku α nad poľom $GF(q)$ je teda normovaný polynóm najmenšieho stupňa nad poľom $GF(q)$, ktorého koreňom je prvok α. Minimálny polynóm prvku α vždy existuje a je daný jednoznačne.

Príklad. Minimálne polynómy prvkov poľa $GF(2^4)$ nad poľom $GF(2)$ sú uvedené v tabuľke 15.6

Poznámka. Časť venovaná konečným poliam bola spracovaná na základe [2] a [10]. Čitateľovi, zaujímajúcemu sa o teóriu konečných polí odporúčame do pozornosti najmä prácu [10]. Zaujímavý pohľad na konečné polia a ich aplikácie v kryptológii ponúka aj práca [**?**].

15.5 Vektorové priestory

Zrejme najznámym príkladom vektorového priestoru je trojrozmerný Euklidovský priestor, ktorý vystupuje v mnohých úlohách stredoškolskej matematiky a fyziky. Euklidovský priestor možno zovšeobecniť na n-rozmerný vektorový priestor nad poľom reálnych čísel, ktorý taktiež nachádza uplatnenie v mnohých aplikáciách. V teórii kódovania nebudeme pracovať s vektorovými priestormi nad reálnymi číslami, ale budeme využívať trocha abstraktnejšie vektorové priestory nad konečnými poľami. Tieto vektorové priestory sú základom pre konštrukciu veľmi dôležitých samoopravných kódov, pre tzv. lineárne kódy. Zavedieme najprv základné pojmy a potom preskúmame vlastnosti vektorových priestorov, ktoré budeme potrebovať (napríklad) pre konštrukciu, kódovanie a dekódovanie lineárnych kódov.

prvok	minimálny polynóm
0	x
α^0	$x+1$
α^1	x^4+x+1
α^2	x^4+x+1
α^3	$x^4+x^3+x^2+x+1$
α^4	x^4+x+1
α^5	x^2+x+1
α^6	$x^4+x^3+x^2+x+1$
α^7	x^4+x^3+1
α^8	x^4+x+1
α^9	$x^4+x^3+x^2+x+1$
α^{10}	x^2+x+1
α^{11}	x^4+x^3+1
α^{12}	$x^4+x^3+x^2+x+1$
α^{13}	x^4+x^3+1
α^{14}	x^4+x^3+1

Tabuľka 15.6: Minimálne polynómy prvkov poľa $GF(2^4)$

Definícia 15.5.1. *Nech je* F *ľubovoľné pole. Nech* V *je množina, na ktorej je definovaná binárna operácia* $+$*, a nech pre každé* $a \in F$ *a* $\mathbf{v} \in V$ *existuje prvok* $a \cdot \mathbf{v} \in V$*, pričom pre aditívne a multiplikatívne operácie platia nasledujúce podmienky:*

1. $(V,+)$ *je abelovská grupa; pre ľubovoľné* $\mathbf{u},\mathbf{v} \in V$ *a ľubovoľné* $a,b \in F$
2. $a \cdot (\mathbf{u}+\mathbf{v}) = a \cdot \mathbf{u} + a \cdot \mathbf{v}$*;*
3. $(a+b) \cdot \mathbf{v} = a \cdot \mathbf{v} + b \cdot \mathbf{v}$*;*
4. $(a \cdot b) \cdot \mathbf{u} = a(b \cdot \mathbf{u})$*;*
5. $1 \cdot \mathbf{u} = \mathbf{u}$,

kde 1 *je jednotkový prvok poľa* F*. Potom* V *je vektorový priestor nad poľom* F*. Prvky množiny* V *sa nazývajú vektory a prvky poľa* F *skaláry.*

Poznámka. Všimnite si, že v definícii vektorového priestoru nad poľom F vystupujú dve rôzne aditívne operácie (sčítanie v poli F a sčítanie v grupe $(V,+)$) a dve takisto rozličné multiplikatívne operácie ("vnútorné násobenie prvkov poľa a "vonkajšie násobenie vektora skalárom.) Z kontextu bude spravidla jasné, o akú operáciu sa jedná, a preto na označenie oboch aditívnych operácií budeme používať symbol "+". Budeme sa pridŕžať zaužívaného označenia a operátor "$\cdot$"budeme vynechávať tak pri označovaní "vonkajšiehoäko aj "vnútorného"násobenia. Aby sme odlíšili vektory a skaláry, budeme vektory sádzať boldom.

Príklad. 1. Nech je F ľubovoľné pole a $n > 1$ je ľubovoľné prirodzené číslo. Potom symbolom F^n označíme množinu všetkých usporiadaných n-tíc prvkov poľa F. Definujeme

operácie sčítania n-tíc a násobenia n-tíc prvkom poľa nasledovne: pre ľubovoľné prvky (n-tice) $\mathbf{u}, \mathbf{v} \in F^n$; $\mathbf{u} = (u_1, \ldots, u_n), \mathbf{v} = (v_1, \ldots, v_n)$ a ľubovoľný prvok $c \in F$ platí

$$\mathbf{u} + \mathbf{v} = (u_1 + v_1, \ldots, u_n + v_n), \quad c\mathbf{u} = (cu_1, \ldots, cu_n).$$

Dá sa ľahko overiť, že F^n s takto definovanými operáciami je vektorový priestor.

2. Trocha netradičným príkladom vektorového priestoru je faktorový okruh polynómov $F[x]/(x^n - 1)$, pozostávajúci z tried reprezentovaných polynómami nad poľom F stupňa menšieho než n.

Nech je daný vektorový priestor V nad poľom F, nech sú $\mathbf{u_1}, \ldots, \mathbf{u_m} \in V$ ľubovoľné vektory a nech sú $a_1, \ldots, a_m \in F$ ľubovoľné skaláry. Vektor

$$\mathbf{v} = a_1\mathbf{u_1} + \cdots + a_m\mathbf{u_m}$$

budeme nazývať *lineárnou kombináciou* vektorov $\mathbf{u_1}, \ldots, \mathbf{u_m}$. Množina vektorov $\mathbf{u_1}, \ldots, \mathbf{u_m}$ sa nazýva *lineárne závislou*, ak existuje množina skalárov $a_1, \ldots, a_m \in F$, z ktorých je aspoň jeden nenulový a

$$\mathbf{0} = a_1\mathbf{u_1} + \cdots + a_m\mathbf{u_m}.$$

Ak množina vektorov $\mathbf{u_1}, \ldots, \mathbf{u_m} \in V$ nie je lineárne závislá, budeme o nej hovoriť, že je *lineárne nezávislá*. Je zrejmé, že ak má byť nejaká množina vektorov lineárne nezávislá, nesmie obsahovať nulový vektor a žiaden z jej vektorov sa nesmie dať vyjadriť v podobe lineárnej kombinácie ostatných vektorov. Množinu všetkých lineárnych kombinácií vektorov $\mathbf{u_1}, \ldots, \mathbf{u_m} \in V$ $\{a_1\mathbf{u_1} + \cdots + a_m\mathbf{u_m}, a_1 \ldots, a_m \in F\}$ budeme označovať symbolom $[\mathbf{u_1}, \ldots, \mathbf{u_m}]$. Budeme hovoriť, že množina vektorov $\mathbf{u_1}, \ldots, \mathbf{u_m}$ generuje vektorový priestor W, ak sa každý vektor z W dá vyjadriť v podobe lineárnej kombinácie vektorov $\mathbf{u_1}, \ldots, \mathbf{u_m}$; t.j.

$$\forall \mathbf{v}(\mathbf{v} \in W \to \mathbf{v} \in [\mathbf{u_1}, \ldots, \mathbf{u_m}]).$$

Je zrejmé, že ten istý vektorový priestor možno generovať pomocou viacerých generujúcich množín vektorov. Budú nás zaujímať mohutnosti generujúcich množín vektorov vektorového priestoru.

Veta 15.5.1 (Steinitzova veta.)**.** *Nech je vektorový priestor V nad poľom F generovaný množinou lineárne nezávislých vektorov $\mathbf{u_1}, \ldots, \mathbf{u_n}$a nech sú vektory $\mathbf{v_1}, \ldots, \mathbf{v_k} \in V$ lineárne nezávislé. Potom $k \leq n$ a existuje $n-k$ vektorov u_i takých, že $[\mathbf{v_1}, \ldots, \mathbf{v_k}, \mathbf{u_{i_1}}, \ldots, \mathbf{u_{i_{n-k}}}] = V$.*

Dôkaz. Budeme postupne nahrádzať vektory $\mathbf{u_i}$ vektormi $\mathbf{v_j}$ v množine generujúcej vektorový priestor V. Dôkaz budeme potom robiť matematickou indukciou vzhľadom na počet vektorov $\mathbf{v_i}$ v množine vektorov generujúcich vektorový priestor V.
Množina $\mathbf{u_1}, \ldots, \mathbf{u_n}$ generuje V; t.j. $[\mathbf{u_1}, \ldots, \mathbf{u_n}] = V$. Pridajme do generujúcej množiny vektor $\mathbf{v_1}$. Keďže $\mathbf{v_1} \in V$ a $\mathbf{v_1} \neq \mathbf{0}$ existuje lineárna kombinácia

$$\mathbf{v_1} = a_1\mathbf{u_1} + \cdots + a_n\mathbf{u_n},$$

taká, že medzi koeficientami $a_1 \ldots, a_n$ je aspoň jeden nenulový. Bez ujmy na všeobecnosti môžeme predpokladať, že $a_1 \neq 0$. Potom môžeme vyjadriť vektor $\mathbf{u_1}$ pomocou lineárnej kombinácie

$$\mathbf{u_1} = \mathbf{v_1} + a_1^{-1}a_2\mathbf{u_2} \cdots + a_1^{-1}a_n\mathbf{u_n}.$$

Z toho vyplýva, že množina $\mathbf{v_1}, \mathbf{u_2}, \ldots, \mathbf{u_n}$ generuje vektorový priestor V.
Predpokladajme, že množina vektorov $\mathbf{v_1}, \ldots, \mathbf{v_{s-1}}, \mathbf{u_s}, \ldots, \mathbf{u_n}$ generuje vektorový priestor V,

Definícia 15.5.2. *Vektorový priestor V nad poľom F sa nazýva konečnorozmerný, ak existujú vektory $\mathbf{u_1}, \ldots, \mathbf{u_n} \in V$ také, že $[\mathbf{u_1}, \ldots, \mathbf{u_n}] = V$. Ak vektorový priestor nie je konečnorozmerný, nazývame ho nekonečnorozmerným vektorovým priestorom.*

Definícia 15.5.3. *Nech je vektorový priestor V nad poľom F konečnorozmerný. Vektory $\mathbf{u_1}, \ldots, \mathbf{u_n} \in V$ nazývame bázou vektorového priestoru V, ak*

1. $[\mathbf{u_1}, \ldots, \mathbf{u_n}] = V$,
2. *vektory $\mathbf{u_1}, \ldots, \mathbf{u_n}$ sú lineárne nezávislé.*

Jeden a ten istý (konečnorozmerný) vektorový priestor môže mať viacero rozličných báz. Podstatné je, že všetky budú mať rovnaký počet prvkov.

Veta 15.5.2. *Nech je V konečnorozmerný vektorový priestor nad poľom F. Potom všetky bázy vektorového priestoru V majú rovnaký počet prvkov.*

Počet prvkov bázy (konečnorozmerného) vektorového priestoru teda nezávisí od výberu bázy. Zavedieme na jeho označenie špeciálny pojem.

Definícia 15.5.4. *Dimenzia konečnorozmerného vektorového priestoru je počet prvkov niektorej z jeho báz. Dimenzia nulového vektorového priestoru je 0. Dimenzia nekonečnorozmerného vektorového priestoru je ∞.*

15.6 Lineárna algebra

V tejto časti zavedieme matice, základné operácie s maticami a ich vlastnosti; pojem determinantu, vlastnosti determinantov a použitie determinantov pri zisťovaní vlastností matíc a na riešenie sústav lineárnych rovníc. Pôjde o základné poznatky nevyhnutné najmä pre konštrukciu, kódovanie, dekódovanie a dokazovanie vlastností lineárnych, cyklických a BCH kódov. Pri písaní tejto časti sme čerpali najmä z prác [4], [8] a [13].

15.6.1 Matice

Definícia 15.6.1. *Maticou $\mathbf{A}$ typu $m \times n$ nazývame mn prvkov $a_{1,1}, a_{1,2}, \ldots, a_{1,n}, a_{2,1}, \ldots, a_{m,n}$ poľa F[5] usporiadaných v m riadkoch a n stĺpcoch:*

$$\mathbf{A} = \begin{pmatrix} a_{1,1}, & a_{1,2}, & a_{1,3}, & \ldots, & a_{1,n} \\ a_{2,1}, & a_{2,2}, & a_{2,3}, & \ldots, & a_{2,n} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ a_{m,1}, & a_{m,2}, & a_{m,3}, & \ldots, & a_{m,n} \end{pmatrix}$$

[5] niektoré vlastnosti matíc platia aj pre slabšie algebraické štruktúry, napr. okruhy, ale väčšina matíc, s ktorými pracujeme v teórii kódovania (najmä pri skúmaní samoopravných kódov) je definovaných ako matice nad nejakými konečnými poľami, a preto sme aj my postavili definíciu matíc na poliach.

Maticu $\mathbf{A}$ typu $m \times n$ budeme označovať $\mathbf{A} = (a_{i,j})_{m,n}$, alebo typ matice vypisovať explicitne; $\mathbf{A} = (a_{i,j})$ je matica typu $m \times n$. Usporiadaná dvojica (i,j) $1 \leq i \leq m, 1 \leq j \leq n$ sa nazýva *miestom matice* , číslo i riadkovým indexom a číslo j stĺpcovým indexom matice, prvok poľa F priradené jednotlivým miestam matice $\mathbf{A}$ budeme nazývať prvkami matice $\mathbf{A}$. Ak $m = b$, maticu $\mathbf{A}$ budeme nazývať štvorcovou maticou rádu n. Prvky $a_{1,1}, a_{2,2}, \ldots, a_{n,n}$ tvoria *hlavnú diagonálu* štvorcovej matice $\mathbf{A}$ (rádu n) a prvky $a_{1,n}, a_{2,n-1}, \ldots, a_{n,1}$ *vedľajšiu diagonálu* matice $\mathbf{A}$. *Nulovou maticou* sa nazýva matica, ktorej všetky prvky sú nulové (rovné neutrálnemu prvku poľa F vzhľadom na sčítanie). Štvorcová matica (rádu n) sa nazýva *diagonálna*, ak $a_{i,j} = 0,\ i \neq j;\ 1 \leq i,j \leq n$ t.j. matica môže mať nenulové prvky len na hlavnej diagonále a všetky jej ostatné prvky sú nulové. Štvorcová matica (rádu n) sa nazýva *jednotková* (alebo aj *identická*), ak je diagonálna a všetky prvky na jej hlavnej diagonále nadobúdajú hodnotu 1, t.j. neutrálneho prvku poľa F vzhľadom na násobenie. Jednotková matica rádu m sa označuje symbolom $\mathbf{I}_m$.

Definícia 15.6.2. *Hodnosťou matice sa nazýva maximálny počet lineárne nezávislých vektorov, tvorených riadkami matice.*

Veta 15.6.1. *Nech je* $\mathbf{A}$ *matica typu* $m \times n$. *Potom pre hodnosť* h *matice* $\mathbf{A}$ *platí*

$$h \leq \min(m, n).$$

Definícia 15.6.3. *Matica*

$$\mathbf{A}^\top = \begin{pmatrix} a_{1,1}, & a_{2,1}, & a_{3,1}, & \ldots, & a_{m,1} \\ a_{1,2}, & a_{2,2}, & a_{3,2}, & \ldots, & a_{m,2} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ a_{1,n}, & a_{2,n}, & a_{3,n}, & \ldots, & a_{m,n} \end{pmatrix}$$

ktorá vznikne výmenou riadkov matice $\mathbf{A}$ *za stĺpce (alebo preklopením okolo hlavnej diagonály) sa nazýva* transponovaná matica *k matici* $\mathbf{A}$.

Veta 15.6.2. *[13] Hodnosť matice* $\mathbf{A}$ *sa rovná hodnosti transponovanej matice* $\mathbf{A}^\top$.

Nasledujúce dve vety prevzaté z [13] poskytujú návod na výpočet hodnosti matice.

Veta 15.6.3. *Hodnosť matice* $\mathbf{A}$ *nad poľom* F *sa nezmení, ak*

1. *zmeníme poradie riadkov v matici,*
2. *vynásobíme jeden riadok matice nenulovým prvkom poľa* F,
3. *pripočítame k jednému riadku matice lineárnu kombináciu ostatných riadkov matice,*
4. *vynecháme v matici riadok, ktorý je lineárnou kombináciou ostatných riadkov matice.*

Poznámka. Využijúc vetu 15.6.2 môžeme úpravy uvádzané vo vete 15.6.3 robiť aj nad stĺpcami matice $\mathbf{A}$.

Veta 15.6.4. *Matica*

$$\mathbf{A} = \begin{pmatrix} a_{1,1}, & a_{1,2}, & a_{1,3}, & a_{1,4}, & \dots, & a_{1,n} \\ 0, & a_{2,2}, & a_{2,3}, & a_{2,4}, & \dots, & a_{2,n} \\ 0, & 0, & a_{3,3}, & a_{3,4}, & \dots, & a_{3,n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0, & \dots, & 0, & a_{m,m}, & \dots & a_{m,n} \end{pmatrix} \tag{15.8}$$

kde sú prvky ležiace na diagonále $a_{1,1}, \dots, a_{m,m}$ *nenulové a prvky pod diagonálou rovné* 0 *má hodnosť* m.

Príklad 15.1. *Využijeme operácie uvádzané vo vete 15.6.3, upravíme maticu na tvar uvedený vo vete 15.6.4 a určíme jej hodnosť. Uvažujme celočíselnú maticu* $\mathbf{A}$

$$\begin{pmatrix} 2 & 0 & -3 & 5 & 1 \\ 6 & -2 & 1 & 3 & 4 \\ 8 & -2 & -2 & 8 & 5 \\ 4 & 1 & 2 & 0 & -1 \end{pmatrix}$$

Tretí riadok je lineárnou kombináciou prvých dvoch, a preto ho možno vynechať

$$\begin{pmatrix} 2 & 0 & -3 & 5 & 1 \\ 6 & -2 & 1 & 3 & 4 \\ 4 & 1 & 2 & 0 & -1 \end{pmatrix}.$$

Teraz preusporiadame stĺpce matice

$$\begin{pmatrix} 1 & 2 & 0 & -3 & 5 \\ 4 & 6 & -2 & 1 & 3 \\ -1 & 4 & 1 & 2 & 0 \end{pmatrix}.$$

Od druhého riadku odčítame štvornásobok prvého a k tretiemu riadku pripočítame prvý riadok:

$$\begin{pmatrix} 1 & 2 & 0 & -3 & 5 \\ 0 & -2 & -2 & 13 & -17 \\ 0 & 6 & 1 & -1 & 5 \end{pmatrix}.$$

A nakoniec k tretiemu riadku pripočítame trojnásobok druhého

$$\begin{pmatrix} 1 & 2 & 0 & -3 & 5 \\ 0 & -2 & -2 & 13 & -17 \\ 0 & 0 & -5 & 38 & -46 \end{pmatrix}.$$

Výsledná matica spĺňa podmienky vety 15.6.4, a preto má matica $\mathbf{A}$ *hodnosť* 3.

Teraz definujeme dôležitý pojem podmatice, ktorý budeme potrebovať pri výpočte determinantov. Budeme vychádzať z matice $\mathbf{A}$ typu $m \times n$. Z množiny $1, \dots, m$ vyberieme k-prvkovú podmnožinu $i_1, \dots, i_k$ a podobne z množiny $1, \dots, n$ vyberieme l-prvkovú podmnožinu $j_1, \dots, j_l$. Podmaticou $\mathbf{A}'$ typu $k \times l$ nazveme maticu

$$\mathbf{A}' = \begin{pmatrix} a_{i_1,j_1}, & a_{i_1,j_2}, & a_{i_1,j_3}, & \ldots, & a_{i_1,j_l} \\ a_{i_2,j_1}, & a_{i_2,j_2}, & a_{i_2,j_3}, & \ldots, & a_{i_2,j_l} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{i_k,j_1}, & a_{i_k,j_2}, & a_{i_k,j_3}, & \ldots, & a_{i_k,j_l} \end{pmatrix}$$

Hoci matice vyzerajú zložito, môžu byť samy prvkami ešte zložitejších štruktúr. Uvažujeme množinu matíc typu $m \times n$ s prvkami z poľa F (nad poľom F). Najprv formalizujeme zdanlivo triviálny, ale mimoriadne dôležitý pojem, *rovnosti matíc*, ktorý potrebujeme na korektné zavedenie operácií nad maticami.

Definícia 15.6.4. *Matica* $\mathbf{A}$ *sa rovná matici* $\mathbf{B}$*, ak sú rovnakého typu a*

$$a_{i,j} = b_{i,j} \quad \forall i, j.$$

Pre tieto, resp. pre matice rovnakého typu nad tým istým poľom môžeme definovať operáciu sčítania.

Definícia 15.6.5. *Súčtom dvoch matíc* $\mathbf{A}$ *a* $\mathbf{B}$ *typu* $m \times n$ *je matica* $\mathbf{B}$ *typu* $m \times n$*, taká že*

$$c_{i,j} = a_{i,j} + b_{i,j}, \quad 1 \leq i \leq m, 1 \leq j \leq n.$$

Keďže do súčtu matíc sa premietajú vlastnosti súčtu prvkov poľa, súčet matíc je asociatívny, komutatívny, v množine všetkých matíc toho istého typu nad poľom F existuje nulový prvok (nulová matica $\mathbf{0}$) a pre každú maticu $\mathbf{A}$ existuje opačná matica $-\mathbf{A}$ taká, že

$$\mathbf{A} + -\mathbf{A} = \mathbf{0}.$$

Množina všetkých matíc rovnakého typu nad tým istým poľom s operáciu sčítania teda tvorí komutatívnu grupu.

Matice môžeme podobne ako vektory násobiť prvkom poľa F: pre maticu $\mathbf{A} = (a_{i,j})$ typu $m \times n$ a prvok $\alpha \in F$ definujeme $\alpha \cdot \mathbf{A} = (\alpha \cdot a_{i,j})$.

Ako to je s násobením matíc? Aby výsledok násobenia matíc nad poľom F bol maticou nad poľom F, využijeme pri násobení matíc skalárny súčin vektorov. Potrebujeme ešte, aby matica $\mathbf{A}$ mala toľko stĺpcov, koľko má matica $\mathbf{B}$ riadkov. Ak je táto podmienka splnená, môžeme definovať súčin matíc

Definícia 15.6.6. *Matica* $\mathbf{C}$ *typu* $m \times n$*, taká že*

$$c_{i,j} = a_{i,1}b_{1,j} + a_{i,2}b_{2,j} + \cdots + a_{i,p}b_{p,j} = \sum_{k=1}^{p} a_{i,k} \cdot b_{k,j},$$

je súčinom matíc $\mathbf{A}$ *typu* $m \times p$ *a* $\mathbf{B}$ *typu* $p \times n$.

Príklad 15.2. *Ukážeme, že súčin matíc nie je komutatívna operácia. Uvažujme dve štvorcové matice nad poľom* $\mathbb{R}$*,*

$$\mathbf{A} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \qquad \mathbf{B} = \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix}.$$

Potom

$$\mathbf{AB} = \begin{pmatrix} 19 & 24 \\ 43 & 50 \end{pmatrix} \qquad \mathbf{BA} = \begin{pmatrix} 23 & 34 \\ 31 & 46 \end{pmatrix}$$

a teda

$$\mathbf{AB} \neq \mathbf{BA}.$$

Vlastnosti súčinu matíc zhrnieme v nasledujúcej vete [4].

Veta 15.6.5. *Nech sú matice* $\mathbf{A}$ *typu* $m \times p$, $\mathbf{B}$ *typu* $p \times r$ *a* $\mathbf{C}$ *typu* $r \times n$ *ľubovoľné matice nad poľom* F, *nech sú* α, β *ľubovoľné prvky poľa* F. *Potom platí*

1. $(\mathbf{AB})\mathbf{C} = \mathbf{A}(\mathbf{BC})$;
2. $(\mathbf{A} + \mathbf{B})\mathbf{C} = (\mathbf{AC} + \mathbf{BC})$;
3. $\mathbf{A}(\mathbf{B} + \mathbf{C}) = \mathbf{AC} + \mathbf{BC}$;
4. $\alpha \cdot (\beta \cdot \mathbf{A} = (\alpha\beta) \cdot \mathbf{A}$;
5. $\alpha \cdot (\mathbf{AB}) = (\alpha \cdot \mathbf{A})\mathbf{B}$;
6. $\mathbf{I}_m\mathbf{A} = \mathbf{A}, \quad \mathbf{A}\mathbf{I}_p = \mathbf{A}$.

15.6.2 Determinanty

Uvažujme štvorcovú maticu

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix}$$

s prvkami z poľa F. Determinantom matice A budeme nazývať výraz

$$\begin{vmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{vmatrix} = a_{1,1}a_{2,2} - a_{1,2}a_{2,1}. \tag{15.9}$$

Determinant matice A budeme označovať symbolom $\det(A)$. Zovšeobecníme definíciu determinantu na prípad štvorcovej matice typu $n \times n$, kde $n \geq 2$. *Permutáciou množiny* $M = \{1, 2, \ldots, n\}$ budeme nazývať ľubovoľnú bijekciu $\phi : M \to M$. Množinu všetkých permutácií množiny M budeme označovať symbolom $P(M)$. Permutáciu ϕ možno jednoznačne zadať pomocou tabuľky:

i	1	2	$\ldots$	n
$\phi(i)$	$\phi(1)$	$\phi(2)$	$\ldots$	$\phi(n)$

resp. ak fixujeme poradie prvkov definičného oboru (prvý riadok tabuľky), tak permutáciu ϕ môžeme jednoznačne zadať pomocou druhého riadka tabuľky zapísaného v podobe vektora: $(\phi(1), \phi(2), \ldots, \phi(n))$. *Inverziou permutácie* ϕ nazveme dvojicu $(1 \leq i < j \leq n)$ takú, že $\phi(i) > \phi(j)$. Napríklad identická permutácia $(1, 2, 3, 4, 5)$ nemá žiadnu inverziu, permutácia $(2, 1, 3, 4, 5)$ má jednu inverziu $(1, 2)$, permutácia $(3, 5, 4, 2, 1)$ má 8 inverzií: $(1, 4); (1, 5); (2, 3); (2, 4); (2, 5); (3, 4); (3, 5); (4, 5)$. Pre zovšeobecnenie pojmu determinantu je popri pojme permutácie dôležitý aj počet inverzií permutácie; počet inverzií permutácie ϕ budeme označovať symbolom $i(\phi)$.

Definícia 15.6.7. *Nech* A *je štvorcová matica typu* $n \times n$ *nad nejakým poľom* F, $M = \{1, 2, \ldots, n\}$;

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \ldots & a_{1,n} \\ a_{2,1} & a_{2,2} & \ldots & a_{2,n} \\ \vdots & & & \ldots \\ a_{n,1} & a_{n,2} & \vdots & a_{n,n} \end{pmatrix}.$$

Determinanom matice A *budeme nazvať výraz*

$$\det(\mathbf{A}) = \sum_{\phi \in P(M)} (-1)^{i(\phi)} \times a_{1,\phi(1)} a_{2,\phi(2)} \ldots a_{n,\phi(n)}. \tag{15.10}$$

Všimneme si dva prípady, $n = 2, 3$. V prvom prípade existujú dve permutácie množiny $\{1, 2\}$; identicka permutácia ϕ_1 a permuácia $\phi_2 = (2, 1)$. Identická permutácia nemá žiadnu inverziu, permutácia ϕ_2 má práve jednu inverziu; $(1, 2)$. Determinant matice A typu 2×2 vyjadrený podľa vzťahu 15.10 sa zhoduje so špecifickým prípadom definovaným vzťahom 15.9. Pozrime sa teraz na prípad matice typu 3×3. Existuje $3! = 6$ rôznych permutácií trojprvkovej množiny, ktorým zodpovedajú nasledujúce členy sumy v 15.10:

ϕ_i	permutácia	$i(\phi_i)$	
ϕ_1	$(1, 2, 3)$	0	$a_{1,1}a_{2,2}a_{3,3}$
ϕ_2	$(1, 3, 2)$	1	$-a_{1,1}a_{2,3}a_{3,2}$
ϕ_3	$(2, 3, 1)$	2	$a_{1,2}a_{2,3}a_{3,1}$
ϕ_4	$(2, 1, 3)$	1	$-a_{1,2}a_{2,1}a_{3,3}$
ϕ_5	$(3, 1, 2)$	2	$a_{1,3}a_{2,1}a_{3,2}$
ϕ_6	$(3, 2, 1)$	3	$-a_{1,3}a_{2,2}a_{3,1}$

Determinant matice A typu 3×3 sa dá vypočítať podľa tzv. Sarusovho pravidla takto: k matici sa pripíšu prvé dva riadky a determinant sa vyjadrí ako súčet súčinov členov ležiacich na diagonálach smerujúcich zľava doprava, od ktorých sa odčítajú súčiny členov ležiacich na diagonálach smerujúcich sprava doľava:

	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	
	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	
$-a_{1,3}a_{2,2}a_{3,1} \leftarrow$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$	$\rightarrow a_{1,1}a_{2,2}a_{3,3}$
$-a_{2,3}a_{3,2}a_{1,1} \leftarrow$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$\rightarrow a_{2,1}a_{3,2}a_{1,3}$
$-a_{3,3}a_{1,2}a_{2,1} \leftarrow$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$\rightarrow a_{3,1}a_{1,2}a_{2,3}$

Uvedieme niektoré zakladné vlastnosti determinantov, ktoré využijeme napr. na skúmanie opravných schopností lineárnych kódov a dekódovanie BCH kódov. Budeme vychádzať z prác [4] a [8]. Nasledujúca veta uvádza, ako závisí determinat matice od vlastností jej riadkov, resp. ako sa zmení pri rozličných operáciách nad riadkami matice. Analogické tvrdenia platia pre stĺce matice.

Veta 15.6.6. *Nech je* $\mathbf{A}$ *štvorcová matica typu* $n \times n$ *nad poľom* F, *s determinantom* $\det(\mathbf{A})$. *Potom platia nasledujúce tvrdenia*

1. *Ak je niektorý riadok matice* $\mathbf{A}$ *nulový, tak potom* $\det(\mathbf{A}) = 0$.

2. *Determinant matice* $\mathbf{A}$ *sa nezmení, ak k niektorému riadku pripočítame ľubovoľný násobok iného riadku matice* $\mathbf{A}$.
3. *Ak matica* $\mathbf{A}$ *obsahuje dva rovnaké riadky, tak potom* $\det(\mathbf{A}) = 0$.
4. *Ak je jeden z riadkov matice* $\mathbf{A}$ *lineárnou kombináciou ostatných riadkov matice* $\mathbf{A}$, *tak* $\det(\mathbf{A}) = 0$.
5. *Ak matica* B *vznikne z matice* A *tak, že sa* i*-ty riadok matice* A *vynásobí konštantou* c, *tak* $\det(B) = c \cdot \det(\mathbf{A})$.
6. *Ak matica* B *vznikne z matice* A *tak, že sa v matici* A *vymenia dva riadky, tak* $\det(B) = -\det(\mathbf{A})$.
7. *Matica* A *je regulárna práve vtedy, ak* $\det(\mathbf{A}) \neq 0$.
8. $\det(\mathbf{A}) = \det(\mathbf{A}^\top)$.

Dôkaz. Dôkazy vyššie uvedených tvrdení možno nájsť napríklad v [8]. □

Pre determinant súčinu matíc platí nasledujúce tvrdenie, ktorého dôkaz možno taktiež nájsť v [8].

Veta 15.6.7. *Nech sú* $\mathbf{A}, \mathbf{B}$ *štvorcové matice typu* $n \times n$ *nad poľom* F. *Potom platí*

$$\det(\mathbf{A} \cdot \mathbf{B}) = \det(\mathbf{A}) \cdot \det(\mathbf{B}).$$

Vráťme sa teraz k pojmu podmatice. Nech je $\mathbf{A}$ matica typu $m \times n$ a $\mathbf{A}'$ je jej štvorcová podmatica typu $k \times k$ $\quad k \leq \min(m, n)$. Determinant matice $\mathbf{A}'$ sa nazýva *subdeterminant* k-teho stupňa matice $\mathbf{A}$. Zatiaľ vieme prakticky vypočítať len determinanty matíc typu nanajvýš 3×3. Ukážeme, že determinanty „väčších“ matíc sa dajú vyjadriť pomocou subdeterminantov nižších stupňov. Uvažujem kvôli jednoduchosti prvok $a_{1,j}$ prvého riadku[6] matice A a všetky členy determinantu $\det(\mathbf{A})$, ktoré $a_{1,j}$ obsahujú. Túto časť determinantu môžeme vyjadriť v tvare $a_{1,j} \cdot \mathbf{A}_{1,j}$, kde výraz $\mathbf{A}_{1,j}$ sa nazýva algebraickým doplnkom prvku $a_{1,j}$. Keďže každý sčítanec determinantu $\det(\mathbf{A})$ obsahuje práve jeden prvok prvého (vo všeobecnosti i-teho) riadku, determinant matice $\mathbf{A}$ možno vyjadriť v tvare

$$\det(\mathbf{A}) = a_{1,1} \cdot \mathbf{A}_{1,1} + a_{1,2} \cdot \mathbf{A}_{1,2} + \cdots + a_{1,n} \cdot \mathbf{A}_{1,n}.$$

Ostáva ešte určiť, ako sa vypočítajú hodnoty $\mathbf{A}_{1,j}$.

Definícia 15.6.8. *Determinant*

$$\mathbf{A}_{ij} = \det \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,j-1} & a_{1,j+1} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,j-1} & a_{2,j+1} & \dots & a_{2,n} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ a_{i-1,1} & a_{i-1,2} & \dots & a_{i-1,j-1} & a_{i-1,j+1} & \dots & a_{i-1,n} \\ a_{i+1,1} & a_{i+1,2} & \dots & a_{i+1,j-1} & a_{i+1,j+1} & \dots & a_{i+1,n} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ a_{n,1} & a_{n,2} & \dots & a_{n,j-1} & a_{n,j+1} & \dots & a_{n,n} \end{pmatrix}$$

[6] úvahy a tvrdenia platia aj pre ľubovoľný prvok tabuľky

podmatice ktorá vznikne vynechaním i*-teho riadka a* j*-teho stĺpca matice* $\mathbf{A}$ *sa nazýva subdeterminant* $(n-1)$*-ho stupňa determinantu* $\det \mathbf{A}$ *prislúchajceho k prvku* $a_{i,j}$. *Doplnkom* $\mathcal{A}_{i,j}$ *prvku* $a_{i,j}$ *v determinante* $\det \mathbf{A}$ *nazývame subdeterminant* $\mathbf{A}_{ij}$ *vynásobený znamienkom* $(-1)^{i+j}$*;*

$$\mathcal{A}_{i,j} = (-1)^{i+j} \cdot \mathbf{A}_{ij}.$$

Teraz môžeme dokončiť úvahy o vyjadrení determinantu pomocou subdeterminantov nižších stupňov.

Veta 15.6.8. *(Rozvoj determinantu podľa* i*-teho riadku) Nech je* $\mathbf{A}$ *štvorcová matica typu* $n \times n$. *Potom platí*

$$\begin{aligned} \det(\mathbf{A}) &= a_{i,1} \cdot \mathcal{A}_{i,1} + a_{i,2} \cdot \mathcal{A}_{i,2} + \cdots + a_{i,n} \cdot \mathcal{A}_{i,n} = \\ &= (-1)^{i+1} \cdot a_{i,1} \cdot \mathbf{A}_{i,1} + (-1)^{i+2} \cdot a_{i,2} \cdot \mathbf{A}_{i,2} + \cdots + (-1)^{i+n} \cdot a_{i,n} \cdot \mathbf{A}_{i,n} \end{aligned}$$

Vetu 15.6.8 využijeme pri výpočte Vandermondovho eterminantu, ktorý zohráva mimoriadne dôležitú úlohu pri určovaní konštrukčnej minimálnej vzdialenosti BCH kódov.

Veta 15.6.9. *Nech je daná štvorcová matica*

$$\mathbf{A} = \det \begin{pmatrix} 1 & x_1 & x_1^2 & x_1^3 & \dots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & x_2^3 & \dots & x_2^{n-1} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & x_n & x_n^2 & x_n^3 & \dots & x_n^{n-1} \end{pmatrix}. \tag{15.11}$$

Potom

$$\det(\mathbf{A}) = \prod_{1 \le i < j \le n} (x_j - x_i). \tag{15.12}$$

Dôkaz. Matematickou indukciou.

1. $n = 2$

$$\det \mathbf{A} = \begin{vmatrix} 1 & x_1 \\ 1 & x_2 \end{vmatrix} = x_2 - x_1. \tag{15.13}$$

2. Predpokladajme, že tvrdenie vety platí pre $m < n$ a dokážeme jeho platnosť pre n. Upravíme maticu 15.11 tak, že od každého riadka druhým počínajúc odčítame prvý riadok:

$$\det \mathbf{A} = \begin{vmatrix} 1 & x_1 & x_1^2 & x_1^3 & \dots & x_1^{n-1} \\ 0 & x_2 - x_1 & x_2^2 - x_1^2 & x_2^3 - x_1^3 & \dots & x_2^{n-1} - x_1^{n-1} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & x_n - x_1 & x_n^2 - x_1^2 & x_n^3 - x_1^3 & \dots & x_n^{n-1} - x_1^{n-1} \end{vmatrix} \tag{15.14}$$

Rozvinieme determinant matice $\mathbf{A}$ podľa 1. stĺpca. Podľa vety 15.6.8 z 15.14 dostávame

$$\det(\mathbf{A}) = \begin{vmatrix} x_2 - x_1 & x_2^2 - x_1^2 & x_2^3 - x_1^3 & \dots & x_2^{n-1} - x_1^{n-1} \\ x_3 - x_1 & x_3^2 - x_1^2 & x_3^3 - x_1^3 & \dots & x_3^{n-1} - x_1^{n-1} \\ \dots & \dots & \dots & \dots & \dots \\ x_n - x_1 & x_n^2 - x_1^2 & x_n^3 - x_1^3 & \dots & x_n^{n-1} - x_1^{n-1} \end{vmatrix} \tag{15.15}$$

Vyjmeme z prvého riadku 15.15 $x_2 - x_1$, druhého $x_3 - x_1$ až $n-1$-ho $x_n - x_1$ a dostávame

$$\det(\mathbf{A}) = \prod_{k=2}^{n}(x_k - x_1) \times \begin{vmatrix} 1 & x_2 + x_1 & x_2^2 + x_2x_1 + x_1^2 & \dots & \sum_{l=0}^{n-2} x_2^{n-2-l} \cdot x_1^l \\ 1 & x_3 + x_1 & x_3^2 + x_3x_1 + x_1^2 & \dots & \sum_{l=0}^{n-2} x_3^{n-2-l} \cdot x_1^l \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_n + x_1 & x_n^2 + x_nx_1 + x_1^2 & \dots & \sum_{l=0}^{n-2} x_n^{n-2-l} \cdot x_1^l \end{vmatrix}$$

Podľa indukčného predpokladu má Vandermondova matica typu $(n-1) \times (n-1)$ determinant

$$\prod_{2 \le j < k \le n} (x_k - x_1) - (x_j - x_1) = \prod_{2 \le j < k \le n} (x_k - x_j).$$

□

Poznámka. Aby sme rozptýlili prípadné pochybnosti o tom, či je posledný determinant v predchádzajúcom dôkaze Vandermondov, spravíme ešte dva kroky dôkazu. Najprv odčítame prvý riadok od všetkých ostatných. V prvom stĺpci máme v prvom riadku jednotky a v ostatných riadkoch nuly. Zasa spravíme rozklad determinantu podľa prvého stĺpca. Kvôli zjednodušeniu výkladu označíme maticu po poslednej úprave $\mathbf{B} = (b_{i,j})_{n-1,n-1}$. Pozrieme sa na $k-1$-vý riadok, $3 < k \le n$. Zrejme $b_{k,0} = 0$. Prvok v druhom stĺpci nadobúda hodnotu

$$b_{k,1} = (x_k + x_1) - (x_2 + x_1) = x_k - x_2,$$

v treťom stĺpci

$$b_{k,2} = (x_k^2 + x_kx_1 + x_1^2) - (x_2^2 + x_2x_1 + x_1^2) = (x_k^2 - x_2^2) + x_1(x_k - x_2),$$

až napokon

$$b_{k,n-1} = \sum_{l=0}^{n-2} x_1^l(x_k^{n-2-l} - x_2^{n-2-l}).$$

Každý prvok $k-1$-ho riadka je deliteľný $(x_k - x_2)$, a to znamená, že z každého prvku v $k-1$-vom riadku môžeme vyňať činiteľ $(x_k - x_2)$. Eliminovali sme ďalší riadok a stĺpec matice a k hodnote determinantu príspevok $\prod_{k=3}^{n}(x_k - x_2)$.

15.6.3 Sústavy lineárnych rovníc

Kapitola 16

Entropia a množstvo informácie

Pri riešení niektorých problémov potrebujeme určiť množstvo informácie obsiahnuté v údajoch. Predpokladajme, že na začiatku nemáme o údajoch žiadnu informáciu, t.j. údaje môžu byť ktorýmkoľvek prvkom nejakej množiny textov (údaje môžu byť napríklad v šifrovej podobe a my nepoznáme ani použitý šifrovací algoritmus a nemáme k dispozícii dešifrovací kľúč). Môžeme nanajvýš odhadnúť potenciálnu množinu otvorených textov, ktorých zašifrovaním vznikli naše údaje. Potom získame nejakú informáciu (napr. o dĺžke a formáte údajov). Táto informácia redukuje počiatočnú neurčitosť - množina možných textov vyhovujúcich získanej informácii je menšia ako pôvodná množina textov. Takto budeme postupovať až do okamihu, keď jednoznačne určíme údaje a neredukujeme neurčitosť na nulovú hodnotu. Kvantitatívna miera informácie obsiahnutej v údajoch sa teda dala určiť pomocou miery neurčitosti. Na meranie neurčitosti sa používa *entropia*. Zavedieme najprv entropiu a pomocou nej aj kvantitatívnu mieru informácie.

Nech je daný zdroj S, s abecedou $\Sigma_S = \{s_0, \ldots, s_{m-1}\}$ a rozdelením pravdepodobností $P = \{p_0, \ldots, p_{m-1}\}$. Na začiatku budeme predpokladať, že je rozdelenie pravdepodobností rovnomerné, t.j. že sa všetky symboly zdrojovej abecedy vyskytujú v textoch rovnako často. Funkcia, označme ju pracovne symbolom f, ktorá má merať neurčitosť zdroja, musí spĺňať nasledujúce prirodzené podmienky:

1. jej hodnota nesmie závisieť od symbolov[1] ale len od rozdelenia pravdepodobností P,
2. funkcia je monotónne rastúca vzhľadom na počet symbolov zdrojovej abecedy [2]
3. aditívnosť: ak sú S_1, S_2 dva nezávislé zdroje, tak potom

$$f(S_1, S_2) = f(S_1) + f(S_2).$$

Vyššie uvedené požiadavky spĺňa logaritmická funkcia. R.V.L. Hartley (1928) definoval logaritmickú mieru informácie (mieru neurčitosti) zdroja S s m-prvkovou abecedou a

[1] symboly môžu byť reprezentované číslami

[2] pripomíname, že rozdelenie pravdepodobností je rovnomerné

rovnomerným rozdelením pravdepodobnosti nasledovne:

$$H(S) = \log(m).$$

Základ logaritmov neovplyvňuje podstatne hodnotu entropie, vzhľadom na to, že pre logaritmy o základoch a, b platí

$$\log_a x = (\log_a b) \log_b x.$$

Jednotkou miery informácie (neurčitosti) je v závislosti od použitého základu logaritmov bit (binárne logaritmy) nat (prirodzené logaritmy) a Hartley (dekadické logaritmy). Množstvo informácie sa najčastejšie vyjadruje v bitoch alebo jednotkách od nich dovedených.

Akú informáciu nesie jeden symbol v prípade, keď rozdelenie pravdepodobností zdroja nie je rovnomerné? Informačný obsah budeme tak ako v predchádzajúcom prípade merať znížením neurčitosti, ktorá závisí od pravdepodobnosti výskytu symbolu. Predpokladajme napríklad, že zdroj má generovať jednu z množiny možných správ, možné správy máme usporiadané lexikograficky. Na výstupe zdroja sa objaví symbol s_{i_0}. Pozrieme sa na dve krajné možnosti: ak sa všetky správy začínajú symbolom s_{i_0}, jeho objavenie neredukovalo množinu možných správ a teda symbol s_{i_0} nenesie žiadnu informáciu. Druhá krajná možnosť - existuje jediná správa, začínajúca symbolom s_{i_0}; t.j. v tomto prípade je množstvo informácie obsiahnuté v s_{i_0} maximálne (rovné logaritmu počtu možných správ). Definujeme množstvo informácie v s_{i_0} ako

$$\log 1/p_{i_0},$$

kde $p_{i_0} = p(s_{i_0})$. V prípade rovnomerného rozdelenia pravdepodobností niesol každý symbol rovnaké množstvo informácie. V prípade nerovnomerného rozdelenia pravdepodobností symbolov tomu tak nie je a má zmysel sa zaoberať strednou hodnotou množstva informácie, t.j. hodnotou

$$H_r(S) = \sum_{i=0}^{m-1} p_i \log_r 1/p_i.$$

Funkcia $H_r(S)$ sa nazýva *entropiou zdroja* S. Entropiu zdroja ako prvý definoval C. Shannon. Pozrieme sa teraz na vlastnosti entropie. Začneme skúmaním funkcie $p \lg 1/p$.

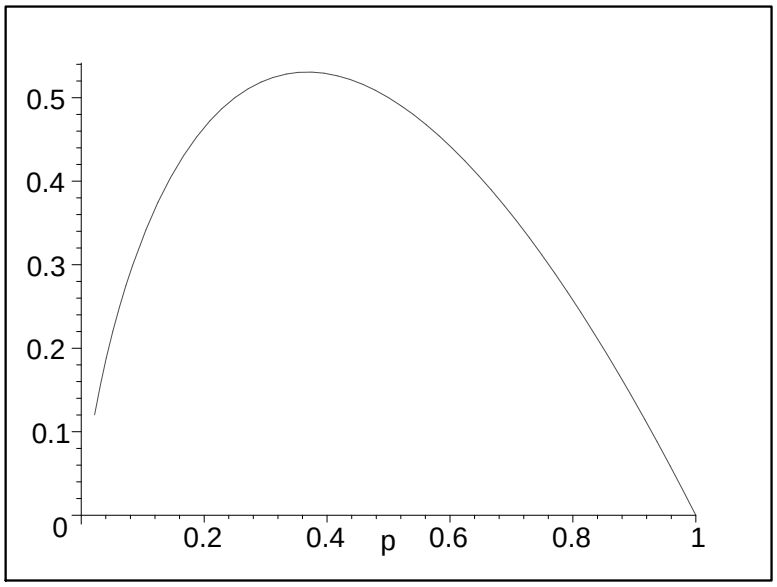
Vypočítame deriváciu funkcie $p \lg 1/p$ a určíme jej extrémy (obr. 1):

$$\frac{d(p \lg 1/p)}{dp} = \lg 1/p - \lg e = 0,$$

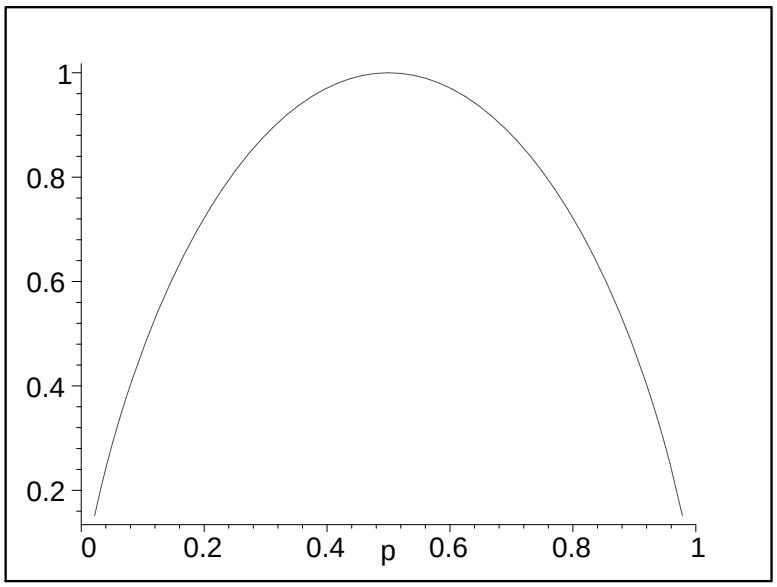
keďže funkcia $p \lg 1/p$ je rastúca na intervale $\langle 0, 1/e)$ a klesajúca na intervale $(1/e, 1\rangle$, nadobúda v bode $p = 1/e$ maximum (≈ 0.5307378454). V bode 0 nadobúda funkcia $p \lg 1/p$ nadobúda hodnotu 0 (L´Hospitalovo pravidlo). Často budeme pracovať so zdrojom, ktorý má binárnu abecedu a rozdelenie pravdepodobností $p, 1-p$. Entropia sa v tomto prípade dá vyjadriť formulou:

$$H_2(p) = p \lg 1/p + (1-p) \lg 1/(1-p).$$

Funkcia $H_2(p)$ (obr. 2) dosahuje maximálnu hodnotu (=1) pre $p = (1-p) = 1/2$. Zhrnieme stručne podstatné vlastnosti entropie.



Obr. 16.1: Graf funkcie $p \cdot \lg 1/p$



Obr. 16.2: Graf funkcie $H_2(p)$

Veta 16.0.10. *Nech je daný zdroj* S*, s abecedou* $\Sigma_S = \{s_0, \ldots, s_{m-1}\}$ *a rozdelením pravdepodobností* $P = \{p_0, \ldots, p_{m-1}\}, m \geq 1$*; entropia zdroja* $H_2(S)$ *je definovaná formulou*

$$H_2(S) = \sum_{i=0}^{m-1} p_i \lg 1/p_i.$$

Potom pre $H_2(S)$ *platí:*

1. $H_2(S)$ *je reálna nezáporná funkcia,*
2. $H_2(S) = 0$ *práve vtedy, ak existuje také* i*, že* $p_i = 1$ *a* $p_j = 0$ *pre* $i \neq j$*,*
3. $H_2(S) \leq m$*.*

Dôkaz. Prvé dve tvrdenia sú očividné a ich dôkazy prenechávame čitateľovi. Dokážeme tretie tvrdenie. Použijeme Lagrangeovu metódu neurčitých koeficientov. Nech

$$f(p_0, p_1, \ldots, p_{m-1}) = \frac{1}{\ln 2} \cdot \sum_{i=0}^{m-1} p_i \ln(1/p_i) + \lambda \cdot \left(\sum_{i=0}^{m-1} p_i - 1 \right)$$

Vypočítame parciálne derivácie funkcie $f(p_0, p_1, \ldots, p_{m-1})$ a položíme ich rovné nule:

$$\frac{\partial f}{\partial p_i} = \frac{1}{\ln 2} [\ln(1/p_i) - 1] + \lambda = 0, \quad i = 0, \ldots m-1.$$

Z poslednej rovnosti vyplýva

$$\lambda = \frac{1}{\ln 2} [1 - \ln(1/p_i)] \quad i = 0, \ldots m-1.$$

Keďže λ je konštanta, to znamená, že pre ľubovoľné i, j platí $p_i = p_j$ a teda $p_i = 1/m$ pre $i = 0, \ldots, m-1$. To znamená, že entropia dosahuje maximálnu hodnotu $H_2(S) = m$ pre rovnomerné rozdelenie pravdepodobností. □

Zoznam obrázkov

Zoznam tabuliek

Register

Literatúra

[1] Adamek J. *Foundation of Coding*. John Wiley, Chichester, 1991.

[2] Blahut R.E. *Theory and practice of error control codes*. Addison Wesley, 1984. ruský preklad, Moskva, Mir 1986.

[3] Hamming R.W. *Coding and Information Theory*. Prentice Hall, New Jersey, 1980.

[4] Havel V. and Holenda J. *Lineární algebra*. SNTL, Praha, 1-st edition, 1984.

[5] Hoffner V. *Úvod do teorie signálú*. SNTL, Praha, 1-st edition, 1979.

[6] Jablonskij S.V. and Lupanov O.B. *Diskrétna matematika a matematické otázky kybernetiky*. Mir, 1974, Moskva. (V ruštine).

[7] Hall J.I. Notes on coding theory. www.math.msu.edu/~jhall, 2003. prednášky.

[8] Katriňák T. et al. *Algebra a teoretická aritmetika*, volume 1. SNTL a Alfa, Praha, Bratislava, 1-st edition, 1985.

[9] Katriňák T. et al. *Algebra a teoretická aritmetika*, volume 2. SNTL a Alfa, Praha, Bratislava, 1-st edition, 1986.

[10] Lidl R. and Niederreiter H. *Introduction to finite fields and their applications*. Cambridge University Press, Cambridge, revised edition, 1994.

[11] MacKay D.J.C. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, Cambridge, 4-th edition, 2005.

[12] Peterson W.W. and Weldon E.J. *Error Correctin Codes*. MIT Press, Cambridge, 2-nd edition, 1972.

[13] Rektorys K. et al. *Přehled užité matematiky*. SNTL, Praha, 4-th edition, 1981.

[14] Rényi A. *Teorie pravděpodobnosti*. Academie, Praha, 1972.

[15] van Lint J.H. *Introduction to Coding Theory*. Springer Verlag, Berlin, 3-rd edition, 1999.

[16] X.Xxx. Error-correcting codes. www.xxx.edu, 2002. rukopis prenášok.