Úvod do teórie kódovania

Daniel Olejár
Martin Stanek

2. októbra 2002

Obsah

Úvod

Úvod do teórie kódovania je prvou zo série kníh Vybrané kapitoly z informatiky a aplikovanej matematiky, ktorú plánuje vydávať DevínLab, spoločné pracovisko Katedry informatiky FMFI UK a firmy ArtInApples, s.r.o. Séria je určená predovšetkým poslucháčom univerzitného štúdia informatiky a informatikom z praxe. Jej cieľom je aspoň čiastočne zaplniť medzeru v ponuke odbornej literatúry a prístupným spôsobom oboznámiť čitateľov so základmi dôležitých a zaujímavých informatických a matematických disciplín a možnosťami ich použitia.

Samotná teória kódovania je pekným príkladom využitia abstraktnej matematiky na riešenie praktických problémov vznikajúcich pri spracovaní informácie. Pri prenose údajov pomocou komunikačného kanála a pri uchovávaní údajov na pamäťových médiách vznikajú chyby, ktoré môžu zostať neodhalené a pri automatickom spracovávaní údajov spôsobovať vážne problémy. Teória kódovania ponúka riešenie v podobe samoopravných kódov, schopných odhaľovať a opravovať chyby spôsobené náhodnými vplyvmi. Druhým aktuálnym problémom je efektívnosť zápisu informácie. Často je potrebné prenášať alebo uchovávať také množstvá údajov, ktoré presahujú existujúce komunikačné alebo archivačné kapacity. Sú známe metódy kódovania, ktoré využívajú napr. štatistické zákonitosti v údajoch na ich transformáciu do „stručnejšej“ podoby.

Kniha „Úvod do teórie kódovania“ je venovaná základom klasickej teórie kódovania: efektívnym a samoopravným kódom.

Kapitola 1

Základné pojmy a označenia

Pojmy ako „kód“, „kódovanie“, „informácia“, „údaje“, „prenos“, „prenosový kanál“ a ďalšie sa v informatike považujú za všeobecne známe a možno aj preto sa často používajú v rozličných významoch. Aby sme sa vyhli nedorozumeniam spôsobeným rozličnou interpretáciou základných pojmov, vybudujeme si potrebný pojmový aparát exaktne. Začneme uvedením potrebných pojmov teórie formálnych jazykov.

1.1 Abecedy, slová a jazyky

Abeceda je ľubovoľná konečná neprázdna množina. Prvky abecedy budeme nazývať *znakmi* alebo *symbolami*. Abecedu budeme označovať symbolom Σ; ak bude potrebné rozlišovať rozličné abecedy, budeme symbol Σ indexovať. 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 štandardnom tvare: $abeceda$. Nech je w slovo nad abecedou Σ, potom počet znakov slova w nazveme „dĺžkou slova w“ a budeme ju 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 symbolom λ. Pre dĺžku prázdneho slova platí $l(\lambda) = 0$. Teraz definujeme operácie nad slovami, pomocou ktorých bude možné 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 λ je obojstranným neutrálnym prvkom vzhľadom na operáciu zreťazenia slov: pre ľubovoľné slovo w platí $w\lambda = \lambda w = w$). 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. $w^0 = \lambda$,
2. $w^{k+1} = w^k w$.

Nech $u = a_1 \dots a_n$ je ľubovoľné slovo, 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 λ 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 $z = a_1 \dots a_k$, $0 < k \leq n$ nazveme počiatočným podslovom (prefixom) slova u a slovo $x = a_i a_{i+1} \dots a_n$; $1 \leq i = n$ 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 $a_n \dots a_1$

Slová môžu byť prvkami množín. Ľubovoľnú množinu slov nad abecedou Σ nazveme *jazykom nad abecedou* Σ. Okrem bežných množinových operácií s jazykmi zavedieme aj operácie odvodené od zreťazovania 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: $\mathcal{L} = \{uv \mid u \in \mathcal{L}_1, v \in \mathcal{L}_2\}$. Jazyk možno zreťazovať so sebou samým; pre ľubovoľný jazyk $\mathcal{L}$ a ľubovoľné číslo $k \in \mathcal{N}$ definujeme:

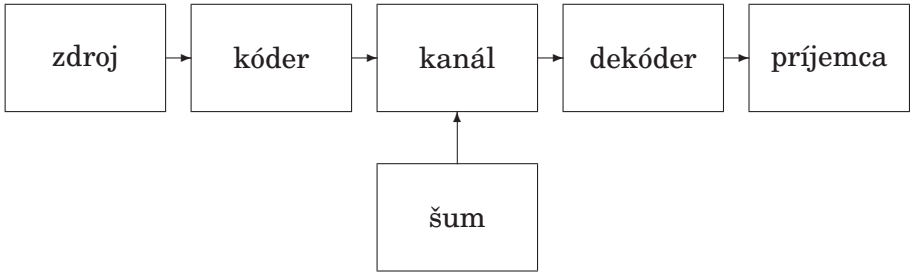
1. $\mathcal{L}^0 = \{\lambda\}$,
2. $\mathcal{L}^{k+1} = \mathcal{L}^k\mathcal{L}$.

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. Potom Σ^* označuje množinu všetkých slov nad abecedou Σ.

Ilustrujeme teraz zavedené pojmy na príkladoch.

1.2 Model prenosového kanála

Využijeme pojmový aparát, ktorý sme vybudovali v predchádzajúcej časti a zavedieme základné pojmy z teórie informácie a teórie kódovania. Budeme predpokladať, že prenášané alebo uchovávané informácie majú podobu slova alebo slov nad nejakou abecedou. Transformácie, ktorými prechádza informácia pri prenose, resp. pri uchovávaní na pamäťovom médiu a čítaní z neho popíšeme pomocou modelu prenosového kanála uvedeného na obr. 1.1 Aby sme sa nemuseli zaoberať tým, odkiaľ informácia pochádza a ani zvláštnosťami jednotlivých zdrojov informácie, predpokladáme, že informácia je zapísaná v tvare postupnosti symbolov z abecedy Σ_S a vytvára ju nejaký *generátor–zdroj informácie.* Abecedu $\Sigma_S = \{s_1, \dots, s_q\}$ budeme nazývať *abecedou zdroja,* alebo *zdrojovou abecedou*; informáciu, ktorá sa prenáša z jedného miesta na druhé budeme nazývať *správou* a informáciu, ktorá sa uchováva na pamäťovom médiu, *údajmi*. Existuje viacero matematických modelov zdroja informácie (generátora), ktorými sa podrobnejšie budeme zaoberať v kapitole 2. Správa, ktorú vytvoril zdroj informácií je určená pre



Obrázok 1.1: Model prenosového kanála

vzdialeného príjemcu. Na prenos správy od zdroja informácie k príjemcovi/adresátovi slúži *prenosový kanál*. Prenosový kanál má vlastnú abecedu, kanálovú abecedu Σ_C, ktorá je vo všeobecnosti rôzna od abecedy zdroja Σ_S. Prenosový kanál z matematického hľadiska predstavuje transformáciu, ktorá vstupnú hodnotu (správu m) zobrazí na výstupnú hodnotu (správu m'). V ideálnom prípade je transformácia realizovaná prenosovým kanálom identická, t.j. $m = m'$, v reálnom živote však činnosť prenosového kanálu komplikuje šum. Šum spôsobuje v správach prenášaných prenosovým kanálom zmeny (chyby) dvojakého typu:

1. nahradenie jedného symbolu prenášanej správy iným symbolom (z abecedy Σ_C),
2. výpadok symbolu, doplnenie nového symbolu (z abecedy Σ_C) do prenášanej správy (poruchy synchronizácie).

V tejto knihe sa budeme zaoberať kódmi, odhaľujúcimi, resp. opravujúcimi chyby prvého typu. Poruchy synchronizácie ??? Výskyt chýb v prenesenej správe závisí od šumu. Šum je však výsledkom celého radu rozličných vplyvov, ktoré nie je možné deterministicky popísať. Budeme preto predpokladať, že chyby v jednotlivých symboloch prenášanej správy vznikajú nezávisle na sebe s rovnakými pravdepodobnosťami.

Ak neplatí vzťah $\Sigma_S \subseteq \Sigma_C$, správu zapísanú v zdrojovej abecede bude pred vstupom do prenosového kanálu potrebné zapísať pomocou symbolov z abecedy Σ_C. Táto transformácia je čiastočným zobrazením[1], ktoré sa nazýva kódovanie správy ($\mathsf{ENC} : \Sigma_S^* \to \Sigma_C^*$) a realizuje ju *kóder*. Výsledkom kódovania správy je *kódovaná* alebo *zakódovaná správa*. Zakódovaná správa vstupuje do prenosového kanála a po prenesení sa spätne transformuje v *dekóderi* na pôvodnú správu nad abecedou Σ_S. Transformácia, ktorú realizuje dekóder je čiastočné zobrazenie ($\mathsf{DEC} : \Sigma_C^* \to \Sigma_S^*$), ktoré sa nazýva dekódovanie (správ) a je inverznou transformáciou ku kódovaniu. Transformácie $\mathsf{ENC}, \mathsf{DEC}$ by mali spĺňať požiadavku jednoznačnosti dekódovania:

$$\forall m \in \Sigma_S^* : \mathsf{DEC}(\mathsf{ENC}(m)) = m.$$

Všimneme si, že hoci sme v modeli prenosového kanála hovorili o prenose správ, možno tento model priamo použiť aj na popis uchovávania a opätovného čítania údajov.

[1] kódovanie môže byť definované na množine správ, ktorá sa nemusí zhodovať s množinou Σ_S^*. Na druhej strane, keďže ENC nemusí byť surjekcia, DEC nemusí byť všade definované zobrazenie.

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. Vzhľadom na túto podobnosť sa v ďalšom budeme zaoberať problémami vznikajúcimi pri prenose informácie v priestore.

Poznámka. V modeli prenosového kanála sme abstrahovali od technickej realizácie jednotlivých častí systému. V reálnych systémoch sa však kanálom neprenášajú nejaké abstraktné symboly

1.3 Kódovanie

Vráťme sa k modelu prenosového kanála z predchádzajúcej časti. Predpokladajme, že prenosový kanál je odolný voči šumu; t.j. že realizuje identickú transformáciu a prenáša správy bez zmeny. Aj v tomto prípade však zostáva vyriešiť, ako zapisovať správy nad abecedou Σ_S pomocou kanálovej abecedy Σ_C. Existuje viacero riešení tohto problému. Začneme tým najjednoduchším; kódovaním znakov zdrojovej abecedy. Nech $\Sigma_S = \{s_0, \ldots, s_{m-1}\}$ je zdrojová abeceda a $\Sigma_C = \{b_0, \ldots, b_r\}$ je abeceda prenosového kanálu (v ďalšom ju budeme nazývať *kódovou abecedou*) 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 na rozdiel od kódovacej transformácie z predchádzajúcej kapitoly, je kódovanie po písmenách totálnym (všade definovaným) zobrazením a keďže zdroj S generuje len postupnosti znakov nad abecedou Σ_S, každá správ 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 dekódovaném 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 kódy, pre ktoré sa nie každá postupnosť kódových symbolov dá rozdeliť na kódové slová jednoznačne.

Príklad 1.3.1. *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*

desiatkovej číslici jej binárny zápis:

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

Binárna postupnosť 001011 *sa dá interpretovať viacerými spôsobmi, ako binárny zápis desiatkových postupností* $001011, 00103, 00211, 0023$.

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

Definícia 1.3.1. *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. Jednoduchým riešením problému rozdeliteľnosti sú *blokové* alebo *rovnomerné kódy*. Blokový kód sa vyznačuje tým, že všetky jeho kódové slová majú rovnakú dĺžku.

Príklad 1.3.2. *Rozšírime predchádzajúci príklad a uvedieme dva spôsoby kódovania desiatkových číslic.*

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 1.3.3. *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 majú problémy kódy pozostávajúce zo slov nerovnakej dĺžky; tzv. *nerovnomerné kódy*. Aj medzi nerovnomernými kódmi existujú rozdeliteľné kódy. Tieto kódy umožňujú zapisovať informáciu častokrát úspornejšie ako blokové kódy a používajú sa najmä na kompresiu údajov. Skôr, ako sa nimi budeme zaoberať podrobnejšie, zovšeobecníme pojem kódovania znakovo zdrojovej abecedy.

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 pokrýva aj kódovanie znakov zdrojovej abecedy; stačí položiť $U = \Sigma_S$.

Príklad 1.3.4. *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, 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 $\mathcal{U}$: $|\mathcal{U}| = 2^{100}$.*

Kapitola 2

Nerovnomerné kódy

Základná motivácia pre používanie nerovnomerných kódov spočíva v tom, že sa prvky množiny ktorú kódujeme, v správach nevyskytujú rovnako často. To umožňuje priradiť často sa vyskytujúcim prvkom kratšie kódové slová a tak dosiahnuť, že kódovaná správa je v priemernom prípade kratšia, ako keby sa na kódovanie používali napríklad blokové kódy.

V tejto kapitole sa budeme zaoberať rôznymi rozdeliteľnými nerovnomernými kódami. Najprv zavedieme prefixové kódy, potom dokážeme Kraftovu-McMillanovu nerovnosť, ktorá poskytuje kritérium na rozdelenie dĺžok kódových slov rozdeliteľného kódu. Nakoniec zavedieme pojem ceny kódu, skonštruujeme dolný odhad na cenu kódu a budeme sa zaoberať konštrukciami optimálnych a kvázioptimálnych kódov. Kvôli zjednodušeniu budeme v priebehu tejto kapitoly predpokladať, že kódová abeceda je binárna.

2.1 Rozdeliteľné kódy

2.1.1 Prefixové kódy

Prefixové kódy predstavujú rozsiahlu triedu nerovnomerných kódov s dobrými vlastosťami, ktoré budeme často využívať v ďalších konštrukciách.[1]

Definícia 2.1.1. *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.

Kód z príkladu 1.3.1 nebol prefixový; kódové slovo 1 bolo prefixom kódového slova 10. Dokážeme, že prefixové kódy sú rozdeliteľné.

Veta 2.1.1. *Ak je kód* $V = v_0, \ldots, v_{m-1}$ *prefixový, potom je rozdeliteľný kód.*

[1] z hľadiska praktického použitia je zaujímavé aj to, že pre prefixové kódy existujú efektívne metódy dekódovania.

Dôkaz. Predpokladajme, že V je prefixový, ale nie je 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 :

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

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:

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

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 spĺňajúca podmienku nemôže existovať a kód V je rozdeliteľný. □

Prirodzenou otázkou je, či existujú aj iné rozdeliteľné kódy okrem prefixových. Uvedieme príklad rozdeliteľného kódu, ktorý nie je prefixový.

Príklad 2.1.1. *Kód* $V = \{0, 01, 11\}$ *nie je prefixový, ale napriek tomu to je rozdeliteľný kód.*

Kód z predchádzajúceho príkladu je tzv. sufixový kód, ktorý je charakteristický 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 budeme rozdeľovať na kódové slová „odzadu“. Príkladom takejto postupnosti, ktorá sa nedá rozdeliť na postupnosť kódových slov, kým sa nedočíta do konca, je postupnosť:

$$0111 \dots 1$$

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: $01(11)^k$.

- kódové stromy
- silne rozdeliteľné kódy
- automatové dekódovanie

2.1.2 Kraftova - McMillanova nerovnosť

Kvôli zjednodušeniu výkladu prijmeme niektoré predpoklady: budeme predpokladať, že zdrojová abeceda Σ_S obsahuje aspoň dva symboly, t.j. $m \geq 2$, že kódová abeceda je binárna a že sa v rozdelení pravdepodobností P nevyskytujú nulové pravdepodobnosti.

Veta 2.1.2 (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*

$$\sum_{i=0}^{m-1} 2^{-l_i} \leq 1. \tag{2.1}$$

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):

$$h_V(x) = \sum_{i=0}^{m-1} x^{-l(v_i)}. \tag{2.2}$$

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

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

Príklad 2.1.2. *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. Matematickou indukciou možno dokázať, že

$$h_{V^n}(x) = \left(\sum_{i=0}^{m-1} x^{-l(v_i)} \right)^n. \tag{2.3}$$

Predpokladajme teraz, že $V = \{v_0, \ldots, v_{m-1}\}$ je rozdeliteľný kód a že $l_i = l(v_i)$, $i = 0, \ldots, m-1$. Symbolom M_i označíme počet slov dĺžky i v kóde V^n a namiesto toho, aby sme v sume ???? sčítavali cez jednotlivé slová, budeme sčítavať cez dĺžky slov (pozri príklad 2.1.2). Maximálnu dĺžku slova v kóde V označíme symbolom l_{max}. Dosadíme namiesto premennej x hodnotu 2 a dostávame:

$$h_{V^n}(2) = \sum_{i=1}^{n \cdot l_{max}} 2^{-i} M_i. \tag{2.4}$$

Je zrejmé, že ak V neobsahuje slovo λ, 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 $M_1 = M_2 = \cdots = M_{n \cdot l_{min}-1} = 0$. Teraz využijeme 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. Ale to 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 ???? a po jednoduchých úpravách dostávame:

$$h_{V^n}(2) = \sum_{i=1}^{n \cdot l_{max}} 2^{-i} M_i \leq \sum_{i=1}^{n \cdot l_{max}} 2^{-i} 2^i = n \cdot l_{max}. \tag{2.5}$$

Na druhej strane, zo vzťahov xxx xxx vyplýva

$$h_{V^n}(2) = \left(\sum_{i=0}^{m-1} x^{-l(v_i)} \right)^n \leq n \cdot l_{max}. \tag{2.6}$$

Posledná nerovnosť platí pre ľubovoľné n. Ak by teda

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

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

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

čo je v spore so vzťahom xxx . To znamená, že

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

Dostatočnosť. Predpokladajme, že $l_0 \leq l_1 \cdots < l_{m-1}$; dĺžky kódových slov sú usporiadané vzostupne, a že 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. Nájdeme také slovo v_k dĺžky l_k, pre ktoré žiadne zo slov $v_0, \ldots, v_{k-1}$ nie je 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 $l_0, \ldots, l_{k-1}$ je

$$\sum_{i=0}^{k-1} 2^{l_k - l_i}. \tag{2.7}$$

Z Kraftovej-McMillanovej nerovnosti však vyplýva, že

$$\sum_{i=0}^{m-1} 2^{-l_i} = \sum_{i=0}^{k-1} 2^{-l_i} + 2^{-l_k} + \cdots + 2^{-l_{m-1}} \leq 1, \tag{2.8}$$

resp.

$$\sum_{i=0}^{k-1} 2^{-l_i} + 2^{-l_k}. \tag{2.9}$$

Vynásobíme rovnosť 2.9 hodnotou 2^{l_k} a upravíme

$$\sum_{i=0}^{k-1} 2^{l_k - l_i} \leq 2^{l_k} - 1. \tag{2.10}$$

Zo vzťahu 2.10 vyplýva, že, existuje aspoň jeden binárny vektor dĺžky l_k, ktorého prefixom nie je žiadne zo slov $\nu_0, \ldots, \nu_{k-1}$. Vyberieme tento vektor ako kódové slovo ν_k.

Predchádzajúci dôkaz mal skôr existenčný ako konštruktívny charakter. Dokážeme ešte raz dostatočnosť Kraftovej-McMillanovej nerovnosti pomocou Shannonovskej konštrukcie.

2. konštrukcia [6]. Rovnako ako v predchádzajúcom dôkaze budeme predpokladať, že $l_0 \leq l_1 \cdots < l_{m-1}$; dĺžky kódových slov sú usporiadané vzostupne, a že platí Kraftova-McMillanova nerovnosť. Zavedieme čísla $q_i,\ i = 0, \ldots, m-1$ nasledovne:

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

Z Kraftovaej-McMillanovej nerovnosti vyplýva, že $0 \leq q_k < 1$. Zapíšeme q_k v binárnom tvare. Zo spôsobu vytvárania q_k a skutočnosti, že $l_0 \leq l_1 \cdots < l_{m-1}$, vyplýva, že q_k sa dá zapísať ako $q_k = (0.b_{k,1} \ldots b_{k,l_{k-1}})_2$, kde $b_{k,i} \in \{0, 1\}$. Kód $V = \{\nu_0, \ldots, \nu_{m-1}\}$ vytvoríme z binárne zapísaných čísel q_k nasledujúcim spôsobom:

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

t.j. slovo ν_i pozostáva z prvých l_i binárnych číslic nasledujúcich po rádovej čiarke v rozvoji čísla q_i. Tvrdíme, že takto zostrojený kód je prefixový, a teda aj rozdeliteľný. Predpokladajme opak, t.j. 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 $\nu_{i+1}, \ldots, \nu_{h-1}, \nu_h$. 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 $q_i,\ q_{i+1}$.

$$q_i = 0.\overbrace{\underbrace{b_{i,1}\dots b_{i,l_{i-1}}}_{l_{i-1}}0\dots0}^{l_i}$$

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

Môžu nastať dve možnosti:

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

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

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

Z uvedeného dôsledku vyplýva, že ak nám nezáleží na konkrétnej podobe kódových slov, môžeme rozdeliteľný kód nahradiť prefixovým kódom s rovnakými dĺžkami kódových slov. Túto možnosť budeme v ďalších úvahách využívať. Skôr ako budeme pokračovať v skúmaní vlastností nerovnomerných kódov, uvedieme príklad Shannonovho kódu, ktorý sme použili v dôkaze Kraftovej-McMillanovej nerovnosti.

Príklad 2.1.3. *Uvažujme nasledujúce dĺžky kódových slov: 2,3,3,4,5,5,6,6. Keďže* $2^{-2} + 2^{-3} + 2^{-3} + 2^{-4} + 2^{-5} + 2^{-5} + 2^{-6} + 2^{-6} < 1$*, z vety 2.1.2 vyplýva, že existuje prefixový kód s týmito dĺžkami kódových slov. Vypočítame hodnoty* q_i *a vyjadríme príslušné kódové slová.*

$q_0 = 0.00$	$l_0 = 2$	$v_0 = 00$
$q_1 = 0.01$	$l_1 = 3$	$v_1 = 010$
$q_2 = 0.011$	$l_2 = 3$	$v_2 = 011$
$q_3 = 0.1$	$l_3 = 4$	$v_3 = 1000$
$q_4 = 0.1001$	$l_4 = 5$	$v_4 = 100010$
$q_5 = 0.10011$	$l_5 = 5$	$v_5 = 100011$
$q_6 = 0.101$	$l_6 = 6$	$v_6 = 101000$
$q_7 = 0.101001$	$l_7 = 6$	$v_7 = 101001$

2.1.3 Úplné kódy

Kód z príkladu 2.1.3 je prefixový, ale má jeden 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 2.1.2. *Binárny rozdeliteľný kód* V *sa nazýva úplným kódom práve vtedy, 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 2.1.4. *Uvažujme blokový kód* $V = \{00, 01, 10, 11\}$*. Keďže kód* V *obsahuje všetky binárne slová dĺžky 2, spĺňa podmienky definície 2.1.2 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 2.1.2, by nemuselo byť jednoduché. Našťastie úplnosť kódu úzko súvisí s Kraftovou-McMillanovou nerovnosťou a prefixovými kódmi.

Veta 2.1.3. *Binárny rozdeliteľný kód* $V = \{v_0, \ldots, v_{m-1}\}$ *je úplný práve vtedy, ak je prefixový a platí*

$$\sum_{i=0}^{m-1} 2^{-l_i} = 1. \tag{2.11}$$

Dôkaz. Predpokladajme, že $V = \{v_0, \ldots, v_{m-1}\}$ je prefixový kód, pre ktorý platí rovnosť (2.11), ale 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 prefixový kód $V' = V \cup \{\beta\}$, pre ktorý platí

$$\sum_{i=0}^{m-1} 2^{-l_i} + 2^{-l(\beta)} = 1 + 2^{-l(\beta)} > 1. \tag{2.12}$$

Ale nerovnosť (2.12) je v rozpore s Kraftovou-McMillanovou nerovnosťou (2.9). 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 preň neplatí rovnosť (2.11). Keďže z rozdeliteľnosti kódu V vyplýva platnosť Kraftovej-McMillanovej nerovnosti (2.9), znamená to, že pre kód V platí

$$\sum_{i=0}^{m-1} 2^{-l_i} < 1. \tag{2.13}$$

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:

$$\sum_{i=0}^{m-1} 2^{n-l_i} \geq 2^n. \tag{2.14}$$

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

$$\sum_{i=0}^{m-1} 2^{n-l_i} = 2^n, \tag{2.15}$$

a

$$\sum_{i=0}^{m-1} 2^{-l_i} = 1, \tag{2.16}$$

čo je v spore s predpokladom (2.13) 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

$$\sum_{i=0}^{m-1} 2^{n-l_i} > 2^n, \tag{2.17}$$

lebo postupnosti začínajúce slovom v_j sú už zarátané v sume (2.17) ako postupnosti začínajúce slovom v_i. Na druhej strane nemôže platiť nerovnosť

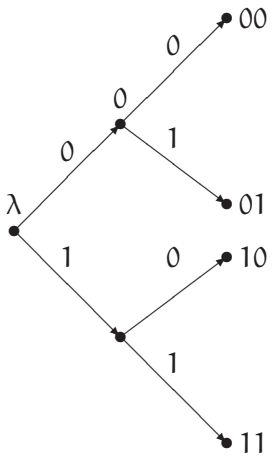
$$\sum_{k=0}^{m-1} 2^{n-l_k} - 2^{l(v_j)} < 2^n, \tag{2.18}$$

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 (2.17). Platnosť nerovnosti (2.17) je však v rozpore s tvrdením vety 2.1.2. Dostávame spor, ktorý dokazuje platnosť nášho tvrdenia. □

2.1.4 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 λ a vrcholu v priradíme postupnosť binárnych hodnôt, ktoré boli priradené hranám ležiacim na ceste, spájajúcej koreň s vrcholom v.[2] 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

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



Obrázok 2.1: Ohodnotený binárny strom

strom binárneho kódu V; $\mathcal{T}(\mathcal{V})$ 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 ododnotením zodpovedajúcim niektorému kódovému slovu kódu V. Na obr. 2.1 je zobrazený binárny (ohodnotený) strom hĺbky 2.

Binárny kódový strom kódu V z príkladu 2.1.3 je zobrazený na obr. 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 2.1.4. *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 2.1.3 nie je úplný; problémy spôsobujú postupnosti 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 $\mathcal{T}(\mathcal{V}'')$ 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[3] prefixový kód, ktorý nie je úplný, možno týmto spôsobom transformovať na

[3]Ako uvidíme neskôr, mnohé z vlastností nerovnomerných kódov nezávisia od počtu znakov kódovej

úplný kód.

2.1.5 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 2.1.1, 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 2.1.3. *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 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. 2.2. 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 kruku 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.

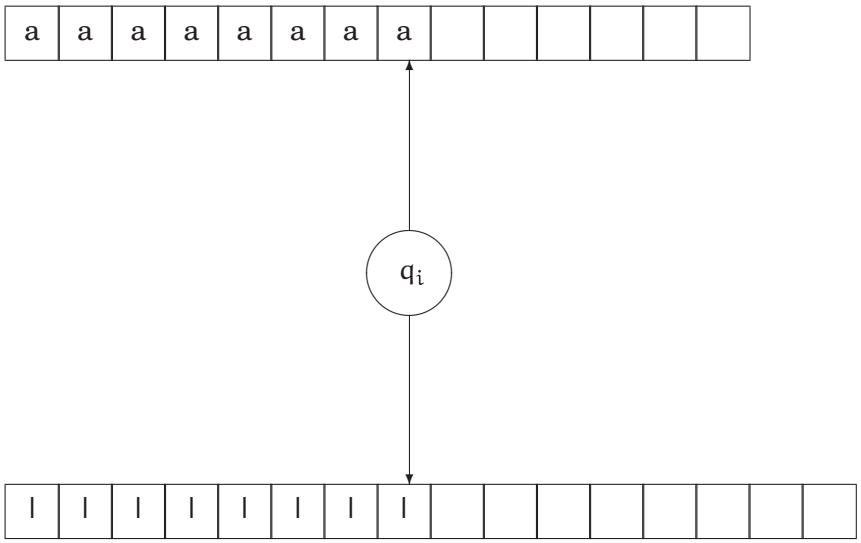
Pri dekódovaní binárnych prefixových kódov pomocou konečného automatu bude vstupná abeceda $\Sigma_i = \{0, 1\}$, výstupná abeceda sa bude zhodovať so zdrojovou abecedou; $\Sigma_o = \Sigma_S$ a prechodovú a výstupnú funkciu definujeme pomocou tabuľky. Ilustrujeme dekódovanie binárnárneho prefixového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

Príklad 2.1.5. *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*

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.



Obrázok 2.2: Konečný automat

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 A dekóduje. Kvôli jednoduchosti budeme pozíciu čítacej hlavy na vstupnej páske a stav automatu A zapisovať tak, že stav automatu zapíšeme pred symbol, ktorý v danom kroku automat A číta. Symboly, ktoré by sa zapisovali na výstupnej páske budeme zapisovať pod dekódované slová binárnej správy.*

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

2.2 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 blokové napríklad 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 sdroj 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 2.2.1. *Nech je* $P = \{p_0, \ldots, p_{m-1}\}$ *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, t.j. 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.

2.3 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 2.3.1. *Nech je* $P = \{p_0, \dots, p_{m-1}\}$ *ľubovoľné rozdelenie pravdepodobností,* $p_0 \geq p_1 \geq \dots \geq p_{m-1} > 0$. *Potom platí*

$$\sum_{i=0}^{m-1} p_i \cdot \log_2 \frac{1}{p_i} \leq \mathcal{L}(P) \leq \sum_{i=0}^{m-1} p_i \cdot \log_2 \frac{1}{p_i} + 1.$$

Rovnosť

$$\sum_{i=0}^{m-1} p_i \cdot \log_2 \frac{1}{p_i} = \mathcal{L}(P) \tag{2.19}$$

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

Dôkaz. Dolný odhad. Predpokladajme, že $V = \{v_0, \dots, v_{m-1}\}$ je ľubovoľný prefixový kód s dĺžkami kódových slov $l(v_i) = l_i,\ i = 0, \dots, m-1$. Porovnáme cenu kódu V s entropiou zdroja $H_2(\mathcal{P}) = \sum_{i=0}^{m-1} p_i \cdot \log_2 \frac{1}{p_i}$:

$$\sum_{i=0}^{m-1} p_i \cdot \log_2 \frac{1}{p_i} - \sum_{i=0}^{m-1} l_i p_i = \sum_{i=0}^{m-1} p_i \cdot \left[\log_2 \frac{1}{p_i} - \log_2 2^{l_i}\right] = \sum_{i=0}^{m-1} p_i \cdot \log_2 \frac{2^{-l_i}}{p_i}.$$

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

$$\sum_{i=0}^{m-1} p_i \cdot \log_2 \frac{2^{-l_i}}{p_i} = \frac{1}{\ln 2} \sum_{i=0}^{m-1} p_i \cdot \ln_2 \frac{2^{-l_i}}{p_i} \leq \frac{1}{\ln 2} \sum_{i=0}^{m-1} p_i \cdot \left[\frac{2^{-l_i}}{p_i} - 1\right] = \tag{2.20}$$

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

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 $V = \{v_0, \dots, v_{m-1}\}$ 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 $l_i = \lceil \log_2 \frac{1}{p_i} \rceil$. Potom platí:

$$\sum_{i=0}^{m-1} 2^{-l_i} = \sum_{i=0}^{m-1} 2^{\lceil \log_2 \frac{1}{p_i} \rceil} \leq \sum_{i=0}^{m-1} 2^{-\log_2 \frac{1}{p_i}} = \sum_{i=0}^{m-1} p_i = 1;$$

a teda existuje prefixový kód s dĺžkami kódových slov $l_i = \lceil \log_2 \frac{1}{p_i} \rceil$, $i = 0, \ldots, m-1$. Cena tohto kódu je

$$\mathcal{L}(P, V) = \sum_{i=0}^{m-1} p_i l_i = \sum_{i=0}^{m-1} p_i \lceil \log_2 \frac{1}{p_i} \rceil \leq \sum_{i=0}^{m-1} p_i \left[\log_2 \frac{1}{p_i} + 1 \right] = \sum_{i=0}^{m-1} p_i \log_2 \frac{1}{p_i} + 1.$$

Vráťme sa ešte k dôkazu rovnosti (2.19). Ak $P = \{p_i = 2^{-l_i},\ l_i \in N,\ i = 0, \ldots, m-1\}$ je rozdelenie pravdepodobností, potom podľa vety 2.1.2 existuje prefixový kód s dĺžkami kódových slov

$$\lceil \log_2 \frac{1}{p_i} \rceil = \lceil \log_2 \frac{1}{2^{-l_i}} \rceil = \lceil \log_2 2^{l_i} \rceil = \lceil l_i \rceil = l_i,$$

ktorý má cenu

$$\sum_{i=0}^{m-1} p_i l_i = \sum_{i=0}^{m-1} p_i \log_2 \frac{1}{p_i}.$$

Na druhej strane, ak

$$\sum_{i=0}^{m-1} p_i l_i = \sum_{i=0}^{m-1} p_i \log_2 \frac{1}{p_i}$$

to znamená, že v odvodení 2.20 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. 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.*)

2.3.1 Fanov kód

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

1. usporiadame pravdepodobnosti zostupne $p_0 \geq p_1 \geq \cdots \geq p_{m-1} > 0$ a zapíšeme do 1. stĺpca tabuľky. Jednotlivým pravdepodobnostiam (zastupujúcim symboly zdrojovej abecedy) priradíme prázdne slová λ.

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 2.3.1.

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	v_i
0.25	0.0	2	00
0.20	0.010	3	010
0.13	0.011	3	011
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 Sahnnonov* 3.22 *a entropia je* 2.822. *Shannonov kód ešte možno upraviť (skrátiť). Keďže slovo* 100 *je prefixom jediného kódového slova, možno toto slovo* 1001 *nahradiť slovom* 100*; podobne 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.

2.3.2 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 budem potrebovať nasledujúcu lemu.

Veta 2.3.2. *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* $V = \{v_0, \dots, v_{m-1}\}$ *s dĺžkami kódových slov* $l_i = l(v_i)$, $i = 0, \dots, m-1$ *optimálny pre rozdelenie pravdepodobností* P*, taký, že minimálnym pravdepodobnostiam* p_{m-2}, p_{m-1} *zodpovedajú slová* v_{m-2}, v_{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 Lemy. □

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

Veta 2.3.3. 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 2.3.2. 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, ináč 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 2.3.2.

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 uznačené 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.

2.3.3 Rozšírenie kódu

Príklad 2.3.3. *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* $\mathcal{L}(\mathcal{P}', \mathcal{V}_\in) = 1.29/2 = 0.645$. *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

$\mathcal{L}(\mathcal{P}'', \mathcal{V}_\ni) = 1.599/3 = 0.533$. *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 vypožčítanú 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 2.3.4.

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
0.625	0.125*							0001	00010
0.625									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
0.625	0.125							011	110
0.625									111

Huffmanov kód V′.

ľahko sa 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.

2.3.4 Chyby v pravdepodobnostiach 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 [12]. 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 $P(e = e_i) = \frac{1}{m}$, $m = 0, \ldots, m-1$. Potom stredná hodnota chyby je

$$E(e) = \sigma^2 = \sum_{i=0}^{m-1} e_i \frac{1}{m} = \frac{1}{m} \sum_{i=0}^{m-1} e_i = 0. \tag{2.21}$$

Vyjadríme disperziu chýb:

$$Var(e) = \sum_{i=0}^{m-1} e_i^2 \frac{1}{m} = \frac{1}{m} \sum_{i=0}^{m-1} e_i^2. \tag{2.22}$$

Využijeme 2.21 a 2.22 a zostrojíme Lagrangeovu funkciu pre veličinu $\sum_{i=0}^{m-1} l_i e_i$ (λ, μ sú Lagrangeove neurčité koeficienty):

$$\mathcal{F} = \frac{1}{m}\sum_{i=0}^{m-1} l_i e_i + \lambda\left(\frac{1}{m}\sum_{i=0}^{m-1} e_i\right) + \mu\left(\frac{1}{m}\sum_{i=0}^{m-1} e_i^2 - \sigma^2\right). \tag{2.23}$$

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

$$\frac{\partial \mathcal{F}}{\partial e_i} = \frac{1}{m}(l_i - \lambda - 2\mu e_i) = 0; \quad i = 0, \ldots, m-1. \tag{2.24}$$

Sčítame rovnice 2.24 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

$$\lambda = \frac{1}{m}\sum_{i=0}^{m-1} l_i. \tag{2.25}$$

Vrátime sa k sústave rovníc 2.24. 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:

$$\mu = \frac{1}{2m\sigma^2}\sum_{i=0}^{m-1} l_i e_i. \tag{2.26}$$

Napokon vynásobíme jednotlivé rovnice sústavy 2.24 príslušnými hodnotami l_i a výsledky násobenia sčítame cez všetky i:

$$\frac{1}{m}\sum_{i=0}^{m-1}\left(l_i^2 - \lambda l_i - 2\mu l_i e_i\right) = \frac{1}{m}\sum_{i=0}^{m-1} l_i^2 - \frac{\lambda}{m}\sum_{i=0}^{m-1} l_i - \frac{2\mu}{m}\sum_{i=0}^{m-1} l_i e_i = 0. \tag{2.27}$$

Dosadíme hodnoty konštánt μ, λ do 2.27 a upravíme

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

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

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. Inák 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 2.3.5. *Ilustrujeme predchádzajúce úvahy na Huffmanových kódoch z príkladu 2.3.4. 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.

2.3.5 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, $t = 0, \ldots$,[4] ktorá spĺňa nasledujúcu podmienku: pre ľubovoľné prirodzené číslo n a ľubovoľné čísla $i_0, \ldots, i_{n+1} \in \{0, \ldots, q-1\}$ platí

$$P(S_{n+1} = s_{i_{n+1}} | S_0 = s_{i_0}, \ldots, S_n = s_{i_n}) = P(S_{n+1} = s_{i_{n+1}} | S_n = s_{i_n}). \tag{2.28}$$

Podmienka 2.28 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 2.28 sa nazýva *Markovovský reťazec*. Analogicky, zdroj S ktorý spĺňa podmienku 2.28, 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 hlavičkých 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

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

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; M. [13].) Ilustrujeme uvedené pojmy na príklade.

Príklad 2.3.6. *Uvažujeme Markovovský zdroj S so štvorprvkovou abecedou $\Sigma_S = \{a, b, c, d\}$. Vzťahy medzi symbolami sú popísané pomocou nasledujúcej matice pravdepodobností*

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

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 že rozdelenie pravdepodobností symbolov v 16-tom kroku nezávisí od toho, aký symbol bol na výstupe zdroja v nultom kroku. Markovovský zdroj popísaný v príklade 2.3.6 je zvláštnym prípadom tzv. *ergodického Markovovského zdroja.* Definujeme ergodický Markovovský zdroj formálne.

Definícia 2.3.1. *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:

$$(p_0, \ldots, p_{q-1}) \times M = (p_0, \ldots, p_{q-1}). \tag{2.29}$$

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

Veta 2.3.4. *(Markovova, [13]) Nech je* S *Markovovský zdroj s abecedou* $\Sigma_S = \{s_0, \ldots, s_{q-1}\}$ *a* $p_{j,k}^{(m)}$ *je pravdepodobnosť prechodu* $s_j \to s_k$ *po* m *krokoch. Ak existujú také prirodzené čísla* $s > 0,\ k_0 \geq 0$, *že* $\forall j,\ j = 0, \ldots, q-1$ *platí* $p_{j,k_0}^{(s)}$; *t.j. v matici* M^s *existuje aspoň jeden stĺpec, ktorý má všetky prvky kladné, tak potom je Markovovský zdroj* S *ergodický, t.j. 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

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

T.j. 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 (2.29) nájdeme stacionárne limitné rozdelenie pravdepodobností. Pripomíname, že matica M nie je regulárna, a preto sústavu (2.29) treba riešiť za predpokladu

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

Príklad 2.3.7. *Nájdeme stacionárne limitné rozdelenie pravdepodobností ergodického Markovovského zdroja* S *z príkladu 2.3.6. (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 $s_{i_0} s_{i_1} \dots s_{i_m}$ vytvorenú Markovovským zdrojom S budeme kódovať nasledovne:

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

Pri dekódovaní najprv dekódujeme prvý symbol,s_{i_0}, zakódovaný pomocou kódu V_{s_0}; na jeho základe 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 2.3.8. *Nech postupnosť* abababcadadca *vytvoril Markovovský zdroj z príkladu 2.3.6. 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 *sme potrebovali* 17*-bitový reťazec.*

Cena kódu Markovovského zdroja závisí od cien čiastkových kódov V_{s_i} a limitného rozdelenia pravdepodobností a dá sa vypočítať na základe nasledujúceho vzťahu:

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

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

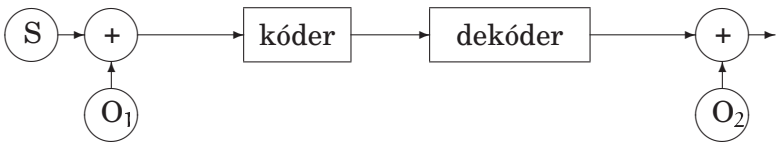
Využitím závislostí medzi jednotlivými symbolmi sme dostali cenu kódu, ktorá je výrazne nižšia ako entropia limitného rozdelenia pravdepodoboností.[5]

2.4 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 metódu kódovania, tzv. *kódovanie s predpoveďou*, ktorá zahŕňa viacero špeciálnych prípadov. Podstata tejto metódy spočíva v nasledujúcom (Obr. 2.3): 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 omocou 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

[5] ktorá závislosti medzi jednotlivými symbolmi nezohľadňuje.



Obrázok 2.3: Kódovanie s predpoveďou

alebo dlhšie postupnosti núl ukončené jednotkami:

$$\alpha \oplus \beta = 0000010011000000000000100001000000100100101000011...$$

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^n1 kde $n \geq 2^k$ sa už pomocou jedného kódového slova nedajú zakódovať. Označíme symbolom C kódovú transformáciu realizovanú kóderom, potom C možno definovať nasledovne:

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

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 parametra 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

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

Na zaklade výrazu $\frac{k}{1-q^m}$ nájdeme optimálnu hodnotu dĺžky bloku k. Keďže analytický výpočet minima výrazu $\frac{k}{1-q^m}$ by mohol byť príliš zložitý,

$$\left(1-p^{2^k-1}\right)^{-1} + \frac{kp^{2^k-1}2^k\ln(2)\ln(p)}{\left(1-p^{2^k-1}\right)^2} = 0$$

vypočítame hodnoty výrazu $\frac{k}{1-q^m}$ pre rozličné p a k a určíme optimálne k:

p	0.5	0.6	0.7	0.8	0.9	0.95	0.99	0.999
$k=1$	2.0	2.5	3.3	5.0	10	20	100	1000
$k=2$	2.29	2.55	3.04	4.09	7.38	14.02	67.33	667.33
$k=3$	3.02	3.09	3.27	3.80	5.75	9.94	44.16	429.86
$k=4$	4.00	4.00	4.02	4.15	5.04	7.45	28.58	268.54
$k=5$	5.00	5.00	5.00	5.00	5.20	6.28	18.68	163.72
$k=6$	6.00	6.00	6.00	6.00	6.00	6.24	12.79	98.22
$k=7$	7.00	7.00	7.00	7.00	7.00	7.01	9.71	58.66
$k=8$	8.00	8.00	8.00	8.00	8.00	8.00	8.66	35.52
$k=9$	9.00	9.00	9.00	9.00	9.00	9.00	9.05	22.48
$k=10$	10.00	10.00	10.00	10.00	10.00	10.00	10.00	15.61
$k=11$								12.63

Kapitola 3

Metódy kompresie údajov

3.1 Slovníkové metódy kompresie dát

V prednáške si ukážeme príklady slovníkových metód kompresie dát. Tieto metódy sa snažia využiť na kompresiu skutočnosť, že v dátach sa častokrát opakujú rovnaké postupnosti znakov.

3.2 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.

3.2.1 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 $i \in \{p-w, \ldots, p-k\}$:

$$T[i, \ldots, i+k-1] = T[p, \ldots, p+k-1]. \tag{3.1}$$

Postup pri kompresii je nasledujúci:

1. $p = 1$
2. pokiaľ je $p < n$ opakujeme:
   (a) nájdeme i a k podľa (3.1)
   (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	$\langle 6, 2, b\rangle$

3.2.2 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.

3.2.3 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.

3.2.4 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 porovnateľný-

mi 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.

3.3 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

3.3.1 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

3.3.2 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

3.3.3 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.

3.4 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.

3.4.1 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:

$$I = \langle d, h) \xmapsto{c_k} \langle d + p'_{k-1}|I|, d + p'_k|I|). \tag{3.2}$$

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 (3.2)
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 $0,0001000000 1000001101$. 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:

3.4.2 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 (3.2). 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.

3.4.3 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.

3.4.4 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 3.5).

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).

3.5 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 3.5.3) a štatistickým kóderom nultého rádu tak, ako je to zobrazené na obrázku. Prirodzene, možné sú aj ďalšie modifikácie.

3.5.1 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`.

3.5.2 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.

3.5.3 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.

3.5.4 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 4

Samoopravné kódy

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] Budeme konštruovať 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.

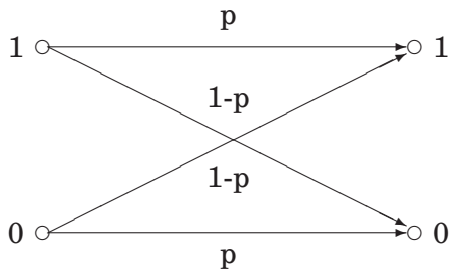
4.1 Princíp samoopravných kódov

4.1.1 Binárny symetrický kanál bez pamäte

Najprv upresníme predstavu o tom, aké chyby môžu vznikať pri prenose správ. Budeme predpokladať, že pri prenose správ

- 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.

[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.



Obrázok 4.1: Binárny symetrický kanál bez pamäte

Na popis prenosového kanála 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 4.1.

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 4.1.1. *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*

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

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

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.*

4.1.2 Geometrická interpretácia samoopravného kódu

Skôr ako pristúpime ku 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 4.1.1. *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* $\operatorname{wt} u$*, definované nasledovne:*

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

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

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

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 $\mathbf{v_0}$:

$$\begin{array}{ll} \mathbf{v_0} & 000\dots0 \\ \mathbf{v_1} & 100\dots0 \\ \mathbf{e_1} & 100\dots0 \\ \mathbf{v_1} \oplus \mathbf{e_1} & 000\dots0 \quad = \mathbf{v_0} \end{array}$$

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 $\mathbf{v_1} = (1, 1, 0, \dots, 0)$. Žiadna chyba váhy 1 nemôže transformovať slovo $\mathbf{v_1}$ na slovo $\mathbf{v_0}$. Čo sa však stane, ak sme prijali slovo

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

$0100\ldots0$? Existujú dve rovnako pravdepodobné možnosti (a množstvo menej pravdepodobných iných):

$$\begin{array}{lll}
\mathbf{v_0} & 000\ldots0 & \\
\mathbf{v_1} & 110\ldots0 & \\
\mathbf{e_1} & 100\ldots0 & \\
\mathbf{e_2} & 010\ldots0 & \\
\mathbf{v_0}\oplus\mathbf{e_1} = & 100\ldots0 & = \mathbf{v_1}\oplus\mathbf{e_2} \\
\mathbf{v_i}\oplus\mathbf{e_j} = & 100\ldots0 & \mathbf{e_j} = \mathbf{v_i}\oplus 100\ldots0
\end{array}$$

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:

$$\begin{array}{lll}
\mathbf{v_0} & 0000\ldots0 & \\
\mathbf{v_1} & 1110\ldots0 & \\
\mathbf{e_1} & 1000\ldots0 & \\
\mathbf{e_2} & 0100\ldots0 & \\
\mathbf{e_3} & 0110\ldots0 & \\
\mathbf{v_0}\oplus\mathbf{e_1} = & 1000\ldots0 & \mathrm{wt}\, v_0 \oplus e_1 = 1 \\
\mathbf{v_1}\oplus\mathbf{e_3} = & 1000\ldots0 & \\
\mathbf{v_1}\oplus\mathbf{e_2} = & 1010\ldots0 & \mathrm{wt}\, v_1 \oplus e_2 = 2
\end{array}$$

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

$$P(1000\ldots0|\mathbf{v_0}) = 0.00868745812768978 > 0.0000877521022998968 = P(1000\ldots0|\mathbf{v_1}),$$

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 $S(\mathbf{v}, r)$ množinu vektorov;

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

ktorú budeme nazývať sférou s polomerom r a stredom $\mathbf{v}$. Je zrejmé, že platí

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

a mohutnosť $S(\mathbf{v}, r)$ je

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

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 sférami 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 sféry s polomerom t vo vektorovom priestore $\{0,\dots,q-1\}$ 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

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

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.

4.2 Jednoduché kódy odhaľujúce/opravujúce chyby

4.2.1 Testovanie parity.

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

$$\begin{array}{ll}0100001110000101 0 & 0\\ 0101101001010101 1 & 1\\ \multicolumn{2}{c}{\dots}\end{array}$$

Doplnený bit sa nazýva *paritným bitom.* Ak v kódovom slove vznikne pri prenose chyba nepárnej váhy $(1,3,5,\dots)$ 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 nepárnej váhy $(2,4,\dots)$, 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.

4.2.2 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.

0110101010	1
1110000111	0
1010101010	1
0010000111	0

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 väššj 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 kotrolnej 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>110101010<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 4.2.1. *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 budem zaoberať kódmi, ktoré majú vysoké prenosové rýchlosti a zároveň dobré opravné schopnosti. Začneme zaujímavým kódom opravujúcim jednu chybu.

4.2.3 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.

[4] v takomto prípade hovoríme o štvorcovom 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 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 $\mathbf{u}\&\mathbf{v} = (u_1 v_1, \ldots, u_n v_n)$. 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 $\mathrm{wt}\,\sigma(i, 4)$ 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 4.2.1. *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:

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

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 ($n = 2^m - 1,\ m \geq 3$)

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

Hammingove kódy však nie sú jedinými kódmi, pre ktoré je mohutnosť sféry mocninou dvojky. Pre $n = 23$ a $t = 3$ platí:

$$\binom{23}{0} + \binom{23}{1} + \binom{23}{2} + \binom{23}{3} = 2^{11};$$

čím je splnená nutná podmienka pre existenciu binárneho samoopravného kódu dĺžky 23 opravujúceho chyby váhy 3. Taký kód skutočne existuje, je to Golayov (23,12)-kód, ktorým sa budeme zaoberať v časti xxx.

4.3 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 uskutočniť 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. Prvé čo nás 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 4.3.1. *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* $(-u_1, \ldots, -u_n) \in G^n$*, kde* $-u_i$ *je opačný prvok k prvku* $u_i,\ i = 1, \ldots, n$.[5]

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. Lineárnym kódom

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

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

Príklad 4.3.2. *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*[6] *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* $7/8$.

Linerny 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 4.3.1. *Nech je* C *lineárny kód. Potom pre minimálnu vzdialenosť* d^* *kódu* C *platí nasledujúci vzťah*

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

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 $\operatorname{wt}\mathbf{u}-\mathbf{v} = d(\mathbf{u},\mathbf{v}) = d^*$. Na druhej strane, ak by v kóde C existovalo nenulové kódové slovo $\mathbf{w}$ s váhou $\operatorname{wt}\mathbf{w} < d^*$, potom by aj $d(\mathbf{w},\mathbf{0}) < d^*$ 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

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

[6] 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 efektívnejšia, 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 [7], 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 riadkov-vektorov 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 podrobnejšia zaoberať vplyvom výberu generujúcej matice na vlastnosti lineárneho kódu.

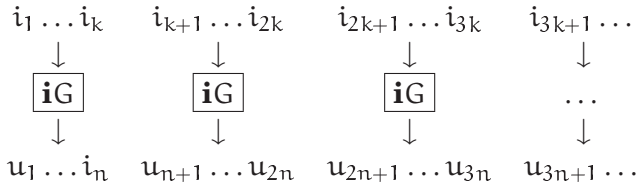
Ako vyzerá kódovanie správy[8] pomocou lineárneho (n, k) kódu? Postupnosť znakov správy sa rozdelí na bloky dĺžky k a tie sa postupne vynásobia generujúcou maticou G a tak sa transformujú na postupnosť kódových slov, obr. 4.2

Ilustrujeme kódovanie správy pomocou lineárneho kódu na nasledujúcom jednoduchom príklade.

Príklad 4.3.3. *Hammingove kódy opravujúce chyby váhy 1 sú lineárne kódy. Kvôli zjednodušeniu výpočtov zoberieme kratší Hammingov kód, ako sme skonštruovali v časti*

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

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



Obrázok 4.2: Kódovanie správy pomocou lineárneho kódu s generujúcou maticou G

4.2.3; 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 ("+" označuje súčet modulo 2):*

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

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.*

$$\mathbf{u} = (u_3 + u_5 + u_7, u_3 + u_4 + u_7, u_3, u_5 + u_6 + u_7, u_5, u_6, u_7),$$

$$\mathbf{v} = (v_3 + v_5 + v_7, v_3 + v_4 + v_7, v_3, v_5 + v_6 + v_7, v_5, v_6, v_7).$$

Potom súčet vektorov

$$\begin{aligned} \mathbf{u} + \mathbf{v} \;=\; & (u_3 + u_5 + u_7 + v_3 + v_5 + v_7, u_3 + u_4 + u_7 + v_3 + v_4 + v_7, u_3 + v_3, \\ & u_5 + u_6 + u_7 + v_5 + v_6 + v_7, u_5 + v_5, u_6 + v_6, u_7 + v_7) = \\ & (u_3 + v_3 + u_5 + v_5 + u_7 + v_7, u_3 + v_3 + u_4 + v_4 + u_7 + v_7, u_3 + v_3, \\ & u_5 + v_5 + u_6 + v_6 + u_7 + v_7, u_5 + v_5, u_6 + v_6, u_7 + v_7) \end{aligned}$$

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 4.3.4. *Kontrolná matica Hammingovho* $(7,4)$ *kódu z príkladu 4.3.3 má tvar*

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

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

Veta 4.3.2. *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 záviských 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, matica 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 4.3.3 (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 lineárny G generujúca matica lineárneho (n,k)-kódu C. Riadky matice G tvoria bázu lineárneho podpriestoru C. Z lineárnej algebry je známe (pozri napr. [5]), 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* [3, 1].

Poznámka. Jacobus van Lint [14] 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 4.3.4. *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.

4.3.1 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 riadu 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[9] kódu pomocou tabuľky dekódovania uvádza nasledujúca veta.

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

Veta 4.3.5. *Ak sa matica štandardného rozkladu používa ako matica dekódovania lineárneho kódu, tak sa prijatý wektor* $\mathbf{w}$ *dekóduje na odvysielaný vektor (kódové slovo)* 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 4.3.5 vyplýva, že nie je možné vylúčiť možnosť nesprávneho dekódovania lineárneho kódu. Predpokladajme, že prenosový kanál je q-nárny symetrický kanál[10] a potom vhodnou voľbou reprezentantov tried rozkladu môžeme minimalizovať pravdepodobnosť nesprávneho dekódovania. Využijeme nasledujúci dôsledok predchádzajúcej vety.

Dôsledok vety 4.3.5 (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 4.3.5. *[3] Lineárny (5,2)-kód* C *s generujúcou maticou*

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

a kontrolnou maticou

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

[10] 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 4.3.6. *[3] 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

4.3.2 Modifikácie lineárnych kódov

4.3.3 Reed-Mullerove kódy

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

$n = 2^m$	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 xxx Základné parametre Reed-Mullerových kódov

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

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

Aby sme mohli popísať konštrukciu generujúcej matice G kódu $\mathcal{R}(m, r)$, 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 $uv = (a_1b_1, \ldots, a_nb_n)$. (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, $1 \geq l \geq r$ 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 Reed-Mullerových kódov na príklade $\mathcal{R}(4, 2)$.

Príklad 4.3.7.

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

Kódovanie a dekódovanie Reed-Mullerových kódov. Nech je daný vektor informačných symbolov i dĺžky k. Kódové slovo vytvoríme tak, že vynásobíme informačný vektor generujúcou maticou. Pri dekódovaní prijatého slova by sme mohli použiť metódu dekódovania lineárnych kódov, vynásobiť prijaté slovo kontrolnou maticou, vypočítať syndróm chyby, na jeho základe určiť najpravdepodobnejšiu chybu a odčítať tento chybový vektor od prijatého slova. Ukážeme inú metódu dekódovania Reed-Mullerových kódov

4.4 Cyklické kódy

4.5 Bose-Chandhury-Hocquenghove kódy

4.6 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).

Počas celej prezenácie metódy budeme predpokladať, že K je cyklický (n, k) kód nad Z_2 opravujúci t chýb. Teda kódové slová v K majú dĺžku n a k informačných symbolov. Nech $g(x)$ je generujúci polynóm kódu K. Metóda 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 Z_2. Označme $\mathrm{wt}(f(x)) = \sum_i a_i$ 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 $\mathrm{wt}(e(x)) \leq t$. Navyše vieme, že $\mathrm{wt}(s(x)) \leq t$ (predpoklad lemy). Teda váha $e(x) - s(x)$ je najviac $2t$. Kód K 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 K, $\deg(g(x)) = n-k$. Preto $s'(x) = e'(x)$. Po zohľadnení predpokladu o počte chýb vo w dostávame $\mathrm{wt}(s'(x)) \leq t$. □

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 $\mathrm{wt}(s'(x)) \leq t$ (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 BCH 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 BCH 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 BCH $(15,5)$ 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 $5/\binom{15}{3} \sim 0,011$.

Kapitola 5

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 kapitole 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 [7, 8]. 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.

5.1 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. **??**.

5.1.1 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 5.1.1. *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 ubovoľný 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 5.1.2. *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* $1/a$.

[1] pojem opačný prvok sa zvykle používať v súvislosti s aditívnou operáciou a pojem inverzný prvok zasa v súvislosti s multiplikatívnou operáciou.

Ďalší pojem je natoľko dôležitý, že si zasluhuje explicitnú definíciu.

Definícia 5.1.1. *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.

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 5.1.3. *(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*. Yrejme najmenšou podgrupou grupy je jednoprvková množina obsahujúca neutrálny prvok grupy. Definujeme teraz pojem podgrupy exaktne.

Mech 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 5.1.1. *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 5.1.4. *1. Uvažujme množinu $Z_6 = \{0, 1, 2, 3, 4, 5\}$ s operáciou $+$ definovanou nasledovne:*

$+$	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. $a^{-m} = (-a)^m = \underbrace{-a - a - \cdots - a}_{m}$

Tam kde nebude potrebné odlišovať multiplikatívne a aditívne grupy, budeme používať terminológiu multiplikatívnych grúp (inverzný prvok, jednotkový prvok, mocnina prvku a pod.)

Definícia 5.1.2. *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* $j \in \mathbb{N}$ *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 5.1.5. *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 1.1.4 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 5.1.6. *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 najsledujúci klasický výsledok.

Veta 5.1.2 (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 (norm8lnej) podgrupy* H*)*.

Príklad 5.1.7. *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.

5.1.2 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 5.1.3. 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

[2] Posúdenie prípadov $m \leq 1$ ponechávame na čitateľa

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 5.1.4. *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 $b = 0$,*
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 5.1.5. *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 [7] hovorí, ako to možno urobiť tak, aby výsledná algebraická štruktúra zostala okruhom.

Veta 5.1.3. *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

$$d = a_0 + a_1 b + \cdots + a_m b^m.$$

Keďže $b \in C'$ 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 5.1.6. *Podokruh* J *okruhu* A *sa nazýva ideálom ak pre každé* $a \in J$ *a* $r \in A$ *platí* $ar \in J$ *a* $ra \in J$.

Pri štúdiu vlastností tzv. cyklických kódov budeme pracovať s algebraickou štruktúrou nazvanou okruhom hlavných ideálov.

Definícia 5.1.7. *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 5.1.8. *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 5.1.4. *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.

5.1.3 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 1.1.3 dajú vyjadriť v tvare

$$a_0 + a_1x + a_2x^2 \cdots + a_nx^n;\ a_k \in B,\ n \in N; \tag{5.1}$$

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 1.1 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 1.1. Vypočítame rozdiel dvoch rozličných reprezentácií prvku $a(x)$:

$$\begin{aligned} 0 &= a(x) - a(x) = a_0 + a_1x + a_2x^2 \cdots + a_nx^n - (b_0 + b_1x + b_2x^2 \cdots + b_mx^m) = \\ &= (a_0 - b_0) + (a_1 - b_1)x + (a_2 - b_2)x^2 \cdots + (a_m - b_m)x^m + a_{m+1}x^{m+1} + \\ &\quad \cdots + a_nx^n. \end{aligned} \tag{5.2}$$

Z rovnosti 1.2 vyplýva, že ľubovoľný prvok okruhu $B[x]$ je možné vyjadriť jednoznačne v tvare 1.1 práve vtedy, ak rovnosť

$$0 = c_0 + c_1x + \cdots c_nx^n$$

platí práve vtedy, ak

$$c_0 = c_1 \cdots = c_n = 0. \tag{5.3}$$

Ak totiž platí 1.3, 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 1.3, 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

[3] v ďalšom budeme najčastejšie predpokladať, že A je pole

$a(x),\ a_n = 1$, 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 5.1.8. *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 1.1.3 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*

$$a_0 + a_1e + \cdots + e_ne^n;\ n \in N,\ a_k \in Q.$$

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 5.1.9. *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 $f(x) = f_0 + f_1x + \cdots + f_nx^n$ je ľubovoľný polynóm okruhu $F[x]$. Pre ľubovoľný prvok $a \in F$ definujeme hodnotu $f(a) = f_0 + f_1a + \cdots + f_na^n$. 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 5.1.10. *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* $f(a) = 0$.

[4] stačil by aj okruh polynómov $Q[\sqrt{2}][x]$

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 5.1.5 (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*

$$f(x) = q(x)g(x) + r(x), \tag{5.4}$$

kde $\deg(r(x)) < \deg(g(x))$ *a polynómy* $q(x), r(x)$ *sú určené jednoznačne.*

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 1.4 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 1.1.5), vyplýva známy fakt, že každý ideál okruhu $F[x]$ je hlavný; t.j. že okruh $F[x]$ je okruhom hlavných ideálov.

Veta 5.1.6. *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 1.4: $f(x) = q(x)g(x) + r(x)$, 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 5.1.11. *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* $\operatorname{lcm}(f(x), g(x))$*), 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 5.1.7. *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:

$$\begin{aligned}
f(x) &= q_1(x)g(x) + r_1(x) \quad 0 \leq \deg(r_1(x)) < \deg(g(x)) \\
g(x) &= q_2(x)r_1(x) + r_2(x) \quad 0 \leq \deg(r_2(x)) < \deg(r_1(x)) \\
r_1(x) &= q_3(x)r_2(x) + r_3(x) \quad 0 \leq \deg(r_3(x)) < \deg(r_2(x)) \\
&\vdots \qquad\qquad\qquad\qquad \vdots \\
r_{s-2} &= q_s(x)r_{s-1}(x) + r_s(x) \quad 0 \leq \deg(r_s(x)) < \deg(r_{s-1}(x)) \\
r_{s-1} &= q_{s+1}(x)r_s(x).
\end{aligned}$$

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 5.1.8. *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 1.1.5 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 5.1.9. *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. □

Veta 5.1.10 (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*

$$f(x) = af_1(x)^{e_1} \cdots f_m(x)^{e_m}, \tag{5.5}$$

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 1.5, 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 1.5. Ukážeme, že aj polynóm stupňa n možno rozložiť na súčin ireducibilných polynómov v tvare 1.5. 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 1.5.

Ostáva ukázať jednoznačnosť rozkladu 1.5. Predpokladajme, že existujú dva rozličné rozklady polynómu $f(x)$; t.j.

$$f(x) = af_1(x)^{e_1} \cdots f_m(x)^{e_m} = bg_1(x)^{d_1} \cdots g_s(x)^{d_s}. \tag{5.6}$$

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 $f_1(x) = cg_k(x)$, kde $c \in F$. Oba polynómy $f_1(x), g_k(x)$ sú normované, a teda $f_1(x) = g_k(x)$. Vydelíme rovnosť 1.6 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. □

5.1.4 Konečné polia

5.1.5 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.

Definícia 5.1.12. *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 "vonkajšieho" ako aj "vnútorného" násobenia. Aby sme odlíšili vektory a skaláry, budeme vektory sádzať boldom.

Príklad 5.1.9. *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.

$$\forall \mathbf{v}(\mathbf{v} \in W \rightarrow \mathbf{v} \in [\mathbf{u_1}, \ldots, \mathbf{u_m}]).$$

Je zrejmé, že ten istý vektorový priestor možno generovať pomocou viacerých generujúcich množín vektorov. Budú nás zaujímať mohutnosti generujúcich množín vektorov vektorového priestoru.

Veta 5.1.11 (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 5.1.13. *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 5.1.14. *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 5.1.12. *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 5.1.15. *Dimenzia konečnorozmerného vektorového priestoru je počet prvkov niektorej z jeho báz. Dimenzia nulového vektorového priestoru je* 0. *Dimenzia nekonečnorozmerného vektorového priestoru je* ∞.

5.1.6 Lineárna algebra

Literatúra

[1] Adamek J. *Foundation of Coding*. John Wiley, Chichester, 1991.

[2] Birkhoff G. and MacLane S. *Prehľad modernej algebry*. Alfa, Bratislava, 1-st edition, 1979.

[3] Blahut R.E. *Theory and practice of error control codes*. Addison Wesley, 1984. ruský preklad, Moskva, Mir 1986.

[4] Hamming R.W. *Coding and Information Theory*. Prentice Hall, New Jersey, 1980.

[5] Havel V. and Holenda J. *Lineární algebra*. SNTL, Praha, 1-st edition, 1984.

[6] Jablonskij S.V. and Lupanov O.B. *Diskrétna matematika a matematické otázky kybernetiky*. Mir, 1974, Moskva. (V ruštine).

[7] Katriňák T. et al. *Algebra a teoretická aritmetika*, volume 1. SNTL a Alfa, Praha, Bratislava, 1-st edition, 1985.

[8] Katriňák T. et al. *Algebra a teoretická aritmetika*, volume 2. SNTL a Alfa, Praha, Bratislava, 1-st edition, 1986.

[9] Lidl R. and Niederreiter H. *Introduction to finite fields and their applications*. Cambridge University Press, Cambridge, revised edition, 1994.

[10] MacLane G. and Birkhoff G. *Algebra*. Alfa, Bratislava, 2-nd edition, 1974.

[11] Peterson W.W. and Weldon E.J. *Error Correctin Codes*. MIT Press, Cambridge, 2-nd edition, 1972.

[12] Rektorys K. et al. *Přehled užité matematiky*. SNTL, Praha, 4-th edition, 1981.

[13] Rényi A. *Teorie pravděpodobnosti*. Academie, Praha, 1972.

[14] van Lint J.H. *Introduction to Coding Theory*. Springer Verlag, Berlin, 3-rd edition, 1999.