Uvod do kodovania

T. K.

May 5, 2009

Uvod (1. lekcia)

Teoria kodovania sa zaobera konstrukciou kodov zameranych hlavne na schopnost opravovat chyby, tzv. *samoopravne kody*, pripadne na zrychlenie prenosu dat. Existuje aj taka cast teorie kodovania, ktora sa zaobera konstrukciou kodov, ktore sluzia hlavne na utajovanie dat. Tato vedecka disciplina sa vola **kryptologia**.

My sa budeme zaoberat len matematickou castou teorie. (Nebude nas zaujimat technicka realizacia.) Pojde vacsinou o aplikaciu algebry v teorii kodovania.

Teoria kodovania vznikla r. 1948. Suviselo to z rozvojom elektronickych pocitacov, elektronickych automatov a s uspechmi raketovej techniky. Prve dolezite vysledky sa objavili v USA (Shannon, Hamming, Golay, Reed, Muller), ZSSR (Kolmogorov), ale aj v Holandsku a Finsku.

Dnes uz existuje viacej literatury zaoberajucej sa teoriou kodovania. Uvediem tri zdroje, ktore budeme aj pouzivat: [1] J. Adamek, Kodovani, SNTL Praha 1989.

[2] E. R. Berlekamp, Algebraic Coding Theory, NY McGraw-Hill, 1968.

[3] J. H. van Lint, Introduction to Coding Theory, Springer Verlag 1999.

Historia

Pozrime sa, ako sme sa dostali ku dnesnemu ponimaniu kodovania. Suvisi to s vyvojom prenosu dat. Vsetko zaviselo od technologickeho pokroku. Davali sa oznamy opticky (ohen, dym) alebo akusticky (bubnovanie a pod.). Napr. medzi lodami sa komunikovalo pomocou mavania zastavkami, resp. v XIX. storoci sa pouzivali svetelne signaly na zaklade Morseovej abecedy (kodu). Koncom 19. stor. sa objavili nove moznosti s objavom telegrafu a telefonu, a hlavne bezdrotoveho telegrafu (Marconi a nas Murgas). Slo vlastne o pociatky radioveho vysielania. Tu sa velmi uspesne uplatnil Morseov kod, ktory sa pouzival este donedavna (koniec minuleho storocia). K Morseovej abecede (kodu) sa este vratime.

Pri dalsom vyvoji zohrala dolezitu ulohu *elektronka*, zvlastny elektricky spinac (rele), ktory ovladal tok elektrickeho prudu v nejakom obvode. Presnejsie, tieto spinace prepustali alebo neprepustali elektricky prud cez urcite obvody pocitaca. Tymto sposobom sa dosiahli dva dolezite stavy: elektricky prud prechadza obvodom, alebo, ze elektricky prud okruhom neprechadza. Matematicky sme prvu situaciu oznacili ako 1 a druhu situaciu, t.j. ze prud obvodom neprechadza, znakom 0. Takto sme mohli na pocitaci realizovat konecne postupnosti z 0 a 1. Spociatku, t.j. este pred elektronkami, bol cely proces pomaly: 10 operacii za sekundu. Elektronky to urychlili: 10.000 operacii za sekundu. Po elektronkach prisla revolucia: objavili sa polovodice (novy material) a z nich sa vyrabali tranzistory (r. 1948). Presnejsie, tranzistory boli vlastne kremikove krystaly. Tie boli omnoho mensie ako elektronky, lacnejsie, vykonnejsie a spolahlivejsie. Tranzistor vykonal za sekundu uz 1,000,000 operacii. Dalsi posun dopredu nastal po roku 1970, ked sa objavili integrovane obvody, t.j. niekolko tranzistorov umiestnenych na malinkych kremikovych platnickach vzajomne pospajanych vodivymi drahami vo viacerych rovinach. Obycajne im hovorime *chipy*. Uz v r. 1980 taky chip mal velkost 5×5 mm a na nom pracovalo az 150.000 tranzistorov. Samozrejme, ze technologia sa stale vylepsuje a s nou sa znizuje aj spotreba elektrickeho prudu. Takto to bezi az podnes.

Tato technika nam poskytuje moznost "komunikovat" so vzdialenymi druzicami, raketami a inymi utvarmi, ktore su vybavene patricnym zariadenim. T.j. my mozeme na dialku ovladat tieto pristroje. Samozrejme musime vyvinut rec, pomocou ktorej budeme komunikovat. Nasa rec bude pozostavat z istych konecnych postupnosti 0 a 1. To budu iste elektricke impulzy, ktore sa prenasaju pomocou elektromagnetickych vln ku prijimacu. Obratene, od prijimaca mozeme zase my prijimat spravy, ktore nam posle vzdialeny prijimac. Vsetko vyzera byt pekne a zda sa, ze nic nestoji v ceste, aby sa tak aj stalo. Pri prevadzkovani tohto systemu vsak zbadame, ze sa objavuju poruchy (volame ich *sumami*), ktore zavinili vonkajsie javy. Vychodiskom bude taka konstrukcia kodov, ktora nam iste chyby dokaze nielenze objavit, ale aj opravit. Konstrukcia vhodnych kodov je vlastne uloha matematikov, ktori pracuju na priprave komunikacie. Ako sa dnes ukazuje, navrhy vhodnych kodov sa nevyskytuje len v tych prikladoch, ktore sme spominali, ale su potrebne aj v mnohych "prizemnych" cinnostiach, ako napr. vyroba hudobnych diskiet, pri praci pocitacov a pod.

Kodovanie bez sumu (2. lekcia)

Pojde nam v prvom rade o pojmy *kodovanie* a *dekodovanie*. Spominali sme, ze spravy sa zvyknu posielat pomocou znakov 0 a 1. Dnes je uz situacia lepsia a mozeme si dovolit pouzivat aj bohatsiu abecedu. Teda, budeme pracovat s konecnymi a neprazdnymi mnozinami. Vezmime napr. mnozinu B. Prvky mnoziny volame *abecedou*. Nad touto abecedou vytvarame konecne postupnosti

$$b_1 \cdot b_2 \cdots b_k = b_1 b_2 \cdots b_k,$$

ktore volame *slovami* nad B. Ak $B = \{0, 1\}$, tak hovorime o *binarnych* slovach. Cislo k nam urcuje *dlzku* slova. Slova dlzky 1 stotoznujeme s prvkami mnoziny, t.j. s abecedou. Mnozinu vsetkych moznych slov nad B oznacujeme ako B^*. Do B^* zaradujeme aj tzv. prazdne slovo. Vsetky slova dlzky k budeme oznacovat ako B^k. Upozornenie: Mnozina B^k je konecna a plati, ze $\mid B^k \mid = \mid B \mid^k$. Na druhej strane, mnozina B^* je nekonecna!

Na mnozine B^* mozeme definovat binarnu operaciu $\mid$ *skladania* slov nasledovne: Nech $\mathbf{b} = b_1 \cdots b_k$, $\mathbf{c} = c_1 \cdots c_n$ su z B^*, tak

$$\mathbf{b} \mid \mathbf{c} = b_1 \cdots b_k c_1 \cdots c_n.$$

D.U. Ukazte, ze $(B^*; \mid)$ tvori *monoid*, t.j. pologrupu s jednotkou.

Pripominame este, ze pre slova **a** a **c** nad abecedou A plati $\mathbf{a} = \mathbf{c}$ prave vtedy, ked maju rovnaku dlzku a rovnaju sa znak po znaku.

Priklad 1.. Pomocou binarnej abecedy mozeme napisat cislice 0,1, ... 8, 9 takto: 0:=0000, 1:=0001, 2:=0010, 3:=0011, 4:= 0100, 5:=1011, 6:= 1100, 7:=1101, 8:=1110, 9:=1111. Pri tejto dohode mozeme napisat cislo 1984 napisane v dekadickom zapise ako: 0001111111100100. Vsimnime si, ze jednotlive binarne slova neoddelujeme medzerami, lebo to by bol uz dalsi znak. (Neskorsie uvidime, ako mozeme medzery pouzivat.)

Definicia 1. Nech A a B su konecne neprazdne mnoziny. Proste zobrazenie

$$\varphi : A \to B^*$$

volame *kodovanim*. Prvky mnoziny A volame *zdrojovou abecedou* alebo *zdrojovymi znakmi*. Prvky mnoziny B volame *kodovou abecedou* alebo *kodovymi znakmi*. Mnozina vsetkych *kodovych slov* $\{\varphi(a) : a \in A\}$ sa nazyva strucne *kod*.

Najdolezitejsi je pripad $B = \{0, 1\}$ s binarnymi znakmi. Vtedy hovorime o *binarnom kode*.

Priklad 2. Mame binarne zakodovat informacie o teplote vody. Zaujimaju nas len 4 moznosti: ladova, studena, vlazna, tepla. Teda zdrojova abeceda ma 4 prvky. Mozeme skusit nasledovne kodovanie: $\varphi : A \to B^*$, kde $A = \{ladova, studena, vlazna, tepla\}$, B je binarna abeceda a plati: ladova$\mapsto 0$, studena$\mapsto 01$, vlazna$\mapsto 011$, tepla$\mapsto 111$. Preco sme volili kodove slova rozlicnej dlzky? Ma to do cinenia s moznym poctom vyskytov patricneho "znaku". Zrejme "ladova" sa nevyskytuje tak casto ako "tepla". Spravu:"ladova, ladova, studena, ladova" zakodujeme ako 00010. (Este k tomu prideme.)

Definicia 2. Kodovanie zdrojovych znakov $\varphi : A \to B^*$ sa da rozsirit na kodovanie zdrojovych sprav $\varphi^* : A^* \to B^*$, pricom

$$\varphi^*(a_1 \cdots a_k) = \varphi(a_1) \mid \varphi(a_2) \mid \cdots \mid \varphi(a_k),$$

t.j. spravu kodujeme znak po znaku.

Definicia 3. Hovorime, ze kodovanie $\varphi : A \to B^*$ je *jednoznacne dekodovatelne*, ak $\varphi^* : A^* \to B^*$ je prostym zobrazenim.

Zrejme plati $B^n \subseteq B^*$.

Definicia 4. Proste zobrazenie $\varphi : A \to B^n$ volame *blokovym kodovanim*.

D.U. Ukazte, ze blokove kodovanie je jednoznacne dekodovatelne.

Ak mame slovo $\mathbf{a} = a_1 a_2 \cdots a_k$, tak podslova tvaru $a_1, a_1 a_2, \cdots, a_1 \cdots a_{k-1}, a_1 \cdots a_{k-1} a_k$ nazyvame *prefixami* slova **a**. Podobne sa definuje *sufix*.

Definicia 5. Kodovanie $\varphi : A \to B^*$ sa nazyva *prefixovym* (*sufixovym*), ak je splnena Fanova podmienka: ziadne kodove slovo nie je prefixom (sufixom) ineho kodoveho slova.

Veta 1. Prefixove (sufixove) kodovanie je jednoznacne dekodovatelne.

Dokaz. Potrebujeme dokazat, ze kodovanie zdrojovych sprav $\varphi^* : A^* \to B^*$ je prostym zobrazenim. Zoberme dve slova $\mathbf{a} = a_1 \cdots a_k$ a $\mathbf{c} = c_1 \cdots c_n$ nad zdrojovou abecedou A. Predpokladajme, ze $\varphi^*(\mathbf{a}) = \varphi^*(\mathbf{c})$, t.j. tieto slova sa rovnaju nad abecedou B. Potrebujeme dokazat, ze $\mathbf{a} = \mathbf{c}$. Prepisme hornu rovnost na

$$\varphi(a_1) \mid \cdots \mid \varphi(a_k) = \varphi(c_1) \mid \cdots \mid \varphi(c_n).$$

Nech l_i je dlzka slova $\varphi(a_i)$ (nad abecedou B) pre $i = 1, \cdots, k$. Podobne, nech t_j je dlzka slova $\varphi(c_j)$ pre $j = 1, \cdots, n$. Z $\varphi^*(\mathbf{a}) = \varphi^*(\mathbf{c})$ vidime, ze zakodovane slova (spravy) su rovnako dlhe, t.j.

$$l_1 + \cdots + l_k = t_1 + \cdots + t_n.$$

Mame dve moznosti: bud $l_1 = t_1$ alebo $l_1 \neq t_1$. Ukazeme, ze len prva moznost moze nastat. Predpokladajme, ze nastal druhy pripad. Znova mame dve moznosti: bud $l_1 < t_1$

alebo $l_1 > t_1$. Zaoberajme sa pripadom $l_1 < t_1$. Tvrdime, ze $\varphi(a_1)$ je prefixom $\varphi(c_1)$, t.j.

$$\varphi(c_1) = \varphi(a_1) \mid \mathbf{d},$$

kde $\mathbf{d} \in B^*$. To je v spore s Fanovou podmienkou. Tento pripad nemoze nastat. Podobne neprichadza do uvahy ani $l_1 > t_1$. (Preco?) Teda ostava $l_1 = t_1$. Z toho vyplyva $\varphi(a_1) = \varphi(c_1)$. Lenze φ je kodovanie, t.j. proste zobrazenie a preto plati $a_1 = c_1$.

Celu doterajsiu uvahu mozeme zopakovat pre podslova **a'** $= a_2 \cdots a_k$ a **c'** $= c_2 \cdots c_n$, pretoze sme slova $\varphi^*(\mathbf{a})$ a $\varphi^*(\mathbf{c})$ vykratili prefixami $\varphi(a_1)$ a $\varphi(c_1)$. Takto dostaneme $a_2 = c_2$ atd. Z toho hned vyplyva, ze $k = n$ a samozrejme $\mathbf{a} = \mathbf{c}$, co sme potrebovali dokazat. Dokaz pre sufixove kodovanie je podobny.

Dosledok. Blokove kodovanie je prefixove (a sucasne aj sufixove).

Priklad 3. Uvedieme priklad prefixoveho kodu, ktory nie je blokovym kodom. Polozme $A = \{a, b, c, d, e, f\}$ a nech $B = \{0, 1, 2\}$. Definujme kodovanie $\varphi : A \to B^*$ nasledovne: $a \mapsto 0$, $b \mapsto 1$, $c \mapsto 20$, $d \mapsto 21$, $e \mapsto 220$ a $f \mapsto 221$.

Definicia 6. Nech $\varphi : A \to B^*$ je jednoznacne dekodovatelny kod s mnozinou kodovych slov $\varphi(A) \subseteq B^*$. Pod *dekodovanim* tohto kodu rozumieme (parcialne) zobrazenie

$$\delta : B^* \to \varphi(A)$$

s vlastnostou, ze pre $\mathbf{b} \in \varphi(A)$ plati: $\delta(\mathbf{b}) = \mathbf{b}$.

Poznamka 1. Dekodovanie je vlastne isty algoritmus, pomocou ktoreho slovam nad kodovou abecedou B priradzujeme kodove slova. V pripade prefixoveho kodu je tento algoritmus jednoduchy: Slovo nad abecedou B citame zlava doprava po jednotlivych znakoch. Akonahle natrafime na prefix, ktory je kodovym slovom, tak tomu prefixu priradime ziskane kodove slovo. V pripade sufixoveho kodu postupujeme zase sprava dolava. Ak narazime na prvy sufix, co je kodovym slovom, tak danemu sufixu priradime ziskane kodove slovo. Pozrime sa na predchadzajuce priklady. Pozrime sa na priklad 2, kde sme kodovali teplotu vody. Ked sa pozrieme teraz na dany kod, tak vidime, ze nie prefixovy, ale sufixovy. Zoberme binarne slovo 01111111. Kedze sa jedna o sufixovy kod, tak postupujeme sprava dolava. Najblizsi sufix, ktory je kodovym slovom je 111, co odpoveda znaku "tepla". Odstrihneme tento sufix a na zvysku znova postupujeme sprava dolava. Znova sa objavi sufix 111, co je zase "tepla". Znova odstranime tento sufix a zostane podslovo 01. Odpoveda to znaku "studena". Teda povodna sprava znela: "studena", "tepla", "tepla". Vsimnime si, ze postup zlava doprava neda vysledok. Ak sa este pozrieme na kod z prikladu 3, tak mozeme dekodovat spravu: 1122112211.

Je to prefixovy kod, tak postupujeme zlava doprava. Po dekodovani dostaneme: bbfbfb.

Poznamka 2. Prefixove kody maju vyhodu oproti sufixovym kodom. Totiz, pri ich dekodovani nemusime vyckat, kym pride cela sprava. Mozeme hned zacat s dekodovanim. U sufixoveho kodu musime najprv vyckat koniec spravy a az potom mozeme zacat s dekodovanim.

Morseov kod (3. lekcia)

Uvedieme si jeden prakticky kod, ktory sa donedavna pouzival na postach, u zeleznice, armady, v namornictve a vobec vo verejnom zivote. Bude to binarny kod. Mnozina A, t,j, zdrojova abeceda obsahuje beznu abecedu a cislice. Mozu tam byt aj dalsie znaky. My uvedieme ten najjednoduchsi pripad.
$a \mapsto .-$ (01); $b \mapsto -...$ (1000);
$c \mapsto -.-.$ (1010); $d \mapsto -..$ (100);

$ch \mapsto ---- $ (1111); $e \mapsto .$ (0);
$f \mapsto ..-.$ (0010); $g \mapsto --.$ (110); $h \mapsto ....$ (0000);
$i \mapsto ..$ (00); $j \mapsto .---$ (0111);
$k \mapsto -.-$ (101); $l \mapsto .-..$ (0100);
$m \mapsto --$ (11); $n \mapsto -.$ (10); $o \mapsto ---$ (111);
$p \mapsto .--.$ (0110); $q \mapsto --.-$ (1101);
$r \mapsto .-.$ (010); $s \mapsto ...$ (000); $t \mapsto -$ (1);
$u \mapsto ..-$ (001); $v \mapsto ...-$ (0001);
$w \mapsto .--$ (011); $x \mapsto -..-$ (1001);
$y \mapsto -.--$ (1011); $z \mapsto --..$ (1100).
1 $\mapsto .----$ (01111); 2 $\mapsto ..---$ (00111);
3 $\mapsto ...--$ (00011); 4 $\mapsto ....-$ (00001);
5 $\mapsto .....$ (00000); 6 $\mapsto -....$ (10000);
7 $\mapsto --...$ (11000); 8 $\mapsto ---..$ (11100);
9 $\mapsto ----.$ (11110); 0 $\mapsto -----$ (11111).

Nie je to prefixovy kod! Totiz nesplnuje Fanovu podmienku. Napr. kodove slovo ku "e" je prefixom kodoveho slova ku "a". Mozeme to vsak jednoducho prerobit na prefixovy kod. Ak za kazdym kodovym slovom bude pauza, tak to uz bude prefixovy kod. Presnejsie povedane, Morseov kod je zobrazenie $\varphi : A \to B^*$, kde

$$A = \{abeceda\} \cup \{0, \cdots, 9\} \quad \text{a} \quad B = \{0, 1\}.$$

Zoberme $B_1 = B \cup \{2\}$. Definujme nove kodove zobrazenie

$$\varphi_1 : A \to B_1^*,$$

tak, ze plati $\varphi_1(x) = \varphi(x) \mid 2$ pre lubovolne pismeno $x \in A$. (V praxi namiesto "2" zaznamename "prazdne miesto".) Odkial vieme, ze φ_1 je uz prefixovy kod? Totiz, keby to nebola pravda, tak jedno kodove slovo by bolo prefixom ineho kodoveho slova (Fano). Teda "2" by sa nachadzalo vo vnutri toho druheho kodoveho slova, co nie je pravdou, lebo kodove slovo ma len jeden znak "2" a ten je na konci slova. Ako priklad uvedieme jednu vetu: -.. -. . ... .— . ... - .-. . -.. .- (Inac napisane: 100 10 0 000 0111 0 000 1 010 0 100 01 = dnes je streda.) Vsimnime si este raz, ze medzery su dolezite, t.j. maju vyznam zvlastneho znaku. Ak by sme nasu spravu poslali bez medzier, tak by to nebolo jednoznacne dekodovatelne. Totiz, dostali by sme bud

$$100 \mid 10 \mid 0 \mid 0 \mid 0 \mid 0 \mid 0 \mid$$

$$11100 \mid 001 \mid 010 \mid 01 \mid 000 \mid 1$$

alebo

$$10 \mid 01 \mid 000 \mid 000 \mid$$

$$11 \mid 10 \mid 000 \mid 101 \mid 001 \mid 00 \mid 01$$

alebo este mnohe dalsie spravy.

Konstrukcia prefixovych kodov (4. lekcia)

Zacneme malym prikladom. Predpokladajme, ze chceme zostrojit binarny prefixovy kod cislic $A = \{0, 1, \cdots, 9\}$. Pritom vieme, ze cislice 0 a 1 sa vyskytuju castejsie a zase 8, 9 zriedkavo. Ako to urobit? Chceme vsak usporny kod φ :

$A \to B^*$ s pomerne kratkymi slovami. Slova $\varphi(0)$ a $\varphi(1)$ nemozu mat dlzku 1. (Preco?) Zacneme, povedzme s $\varphi(0) = 00$ a $\varphi(1) = 01$. Zvysne cifry $2, \cdots, 9$ uz takto kodovat nemozeme, pretoze vyrobime len 4 slova dlzky 2. Takisto to nejde zo slovami dlzky 3, pretoze ich je 8 a chceme prefixovy kod. Skusme zo slovami dlzky 4. Teraz uz mozeme napisat: $0 \mapsto 00$, $1 \mapsto 01$, $2 \mapsto 1000$, $3 \mapsto 1100$, $4 \mapsto 1010$, $5 \mapsto 1001$, $6 \mapsto 1110$, $7 \mapsto 1101$, $8 \mapsto 1011$, $9 \mapsto 1111$. Rychle sa presvedcime, ze sa jedna o prefixovy kod.

Pred dalsou vetou sa dohovorime na tom,
ze mame zdrojovu abecedu $A = \{a_1, \cdots, a_r\}$, pricom dlzky odpovedajucich kodovych slov su $d_1, \cdots, d_r$ a plati

$$d_1 \leq d_2 \leq \cdots \leq d_r.$$

Veta 2. Nech A a B su dve konecne mnoziny, pricom $A = \{a_1, \cdots, a_r\}$ a $\mid B \mid = n \geq 2$. Predpokladajme, ze $d_1 \leq \cdots \leq d_r$ je postupnost prirodzenych cisel. Potom sa da zostrojit prefixovy kod $\varphi : A \to B^*$ s vlastnostou $\mid \varphi(a_i) \mid = d_i$ pre kazde $i = 1, \cdots, r$ prave vtedy, ked plati tzv. **Kraftova** nerovnost

$$n^{-d_1} + n^{-d_2} + \cdots + n^{-d_r} \leq 1.$$

Dokaz. Predpokladajme, ze mame prefixovy kod $\varphi : A \to B^*$ s vlastnostou $\mid \varphi(a_i) \mid = d_i$ pre kazde $i = 1, \cdots, r$. Prefixovost kodu hovori, ze $\varphi(a_1)$ nesmie byt prefixom slova $\varphi(a_2)$, ani $\varphi(a_3)$ atd. ani u slova $\varphi(a_r)$. Podobne je to zo slovom $\varphi(a_2)$. Nesmie byt prefixom slov $\varphi(a_3), \cdots, \varphi(a_r)$. (Samozrejme, ani u slova $\varphi(a_1)$. Lenze ono je kratsie alebo rovnako dlhe ako $\varphi(a_2)$. Kedze φ je kod, tak to automaticky plati.) Takto vidime, ze $\varphi(a_i)$ pre $1 \leq i \leq r$ nesmie byt prefixom slov $\varphi(a_{i+1}), \cdots, \varphi(a_r)$. To nas vedie k dolezitej definicii (dohode): Nech $Z(i, k)$ pre $1 \leq i < k \leq r$ znamena mnozinu vsetkych slov $\mathbf{a} \in B^*$ dlzky d_k tvaru

$$\mathbf{a} = \varphi(a_i) \mid \mathbf{d}$$

pre $\mathbf{d} \in B^*$. Platia nasledovne pomocne vety

Lemma A. $\mid B^j \mid = n^j$ pre kazde $1 \leq j$.

Dokaz lemmy A. Nech $\mathbf{b} = b_1 b_2 \cdots b_j \in B^j$. Podla predpokladu vieme, ze $\mid B \mid = n$. Teda v $\mathbf{b}$ na prvom mieste mame n-moznosti vyberu prvku b_1. Podobne je to na druhom atd. az na poslednom mieste. Na vyber slova $\mathbf{b}$ mame

$$n \cdot n \cdots n = n^j$$

moznosti.

Lemma B. $\mid Z(i,k) \mid = n^{d_k - d_i}$.

Dokaz lemmy B. Podla definicie,

$$\mathbf{a} = \varphi(a_i) \mid \mathbf{d}.$$

Vieme, ze $\mid \mathbf{a} \mid = d_k$. Predpokladame, ze $\mid \varphi(a_i) \mid = d_i$. Teda $\mid \mathbf{d} \mid = d_k - d_i$. Teraz sa ukaze podobne ako v predchadzajucej lemme, ze

$$\mid B^{d_k - d_i} \mid = \mid Z(i,k) \mid = n^{d_k - d_i}.$$

Lemma C. Nech $i \neq j$. Potom

(i) $Z(i,k) \cap Z(j,k) = \emptyset$;

(ii) $Z(i,r) \cap \{\varphi(a_r)\} = \emptyset$ pre $i \neq r$.

Dokaz lemmy C. (i) Nech $i < j$. Urobime nepriamy dokaz. Teda, nech **a** $\in Z(i,k) \cap Z(j,k)$. Potom

$$\mathbf{a} = \varphi(a_i) \mid \mathbf{d} = \varphi(a_j) \mid \mathbf{c}.$$

Z $i < j$ vychadza, ze $\varphi(a_i)$ je prefixom $\varphi(a_j)$, co je spor. Podobny spor dostaneme pre $j < i$. V pripade (ii) postupujeme tiez nepriamo. Totiz,

$$\mathbf{a} \in Z(i,r) \cap \{\varphi(a_r)\}$$

znamena, ze $\varphi(a_i)$ je prefixom $\varphi(a_r)$, co je spor.

Vratime sa k dokazu vety. Teraz vidime, ze

$Z(1,r) \cup Z(2,r) \cup \cdots \cup Z(r-1,r) \cup \{\varphi(a_r)\} \subseteq B^{d_r}$.

Z toho vyplyva (lemmy A - C)

$$| Z(1,r) \cup \cdots \cup Z(r-1,r) \cup \{\varphi(a_r)\} | =$$

$$= | Z(1,r) | + \cdots + | Z(r-1,r) | + 1 =$$

$$= n^{d_r - d_1} + n^{d_r - d_2} + \cdots + n^{d_r - d_{r-1}} + 1 \leq n^{d_r}.$$

Poslednu nerovnost vydelime s n_r^d, co je to iste, ako keby sme to vynasobili s (kladnym) cislom n^{-d_r}. Tak dostaneme

$$n^{-d_1} + n^{-d_2} + \cdots + n^{-d_r} \leq 1$$

a to je uz Kraftova nerovnost.

Obratene, predpokladajme, ze plati Kraftova nerovnost. Ukazeme, ze existuje prefixovy kod s predpisanymi parametrami. Presnejsie, dokaz bude konstruktivny, t.j. ukazeme ako sa taky kod da vyrobit. Teda, ukazeme, ako sa da vyrobit prefixovy kod $\varphi : A \to B^*$. Budeme postupovat pomocou matematickej indukcie vzhladom na usporiadanie prvkov mnoziny $A = \{a_1, a_2 \cdots, a_r\}$. Presnejsie, v prvom kroku urcime $\varphi(a_1)$. V druhom kroku budeme predpokladat, ze uz poznáme hodnoty $\varphi(a_1), \cdots, \varphi(a_k)$ pre $1 \leq k < r$. Na zaklade tejto informacie urcime $\varphi(a_{k+1})$. Zacneme s prvym krokom matematickej indukcie.

1° Z lemmy A vyplyva, ze $| B^{d_1} |= n^{d_1} \geq 2$. Preto existuje slovo $\mathbf{b} \in B^{d_1}$. Polozme

$$\varphi(a_1) = \mathbf{b} \in B^*.$$

2° (Indukcny predpoklad.) Predpokladajme, ze uz poznáme $\varphi(a_1), \cdots, \varphi(a_k)$ pre $1 \leq k < r$, pricom sa jedna o proste zobrazenie. Nech navyse plati, ze $\varphi(a_i)$ nie je prefixom $\varphi(a_j)$ pre akekolvek $1 \leq i < j \leq k$. Chceme ukazat, ze existuje taka hodnota $\varphi(a_{k+1}) \in B^*$ s vlastnostou: zobrazenie bude zase proste a $\varphi(a_i)$ nie je prefixom $\varphi(a_{k+1})$ pre ziadne $1 \leq i \leq k$. Zacneme Kraftovou nerovnostou

$$(n^{-d_1}+\cdots+n^{-d_{k+1}})+(n^{-d_{k+2}}+\cdots+n^{-d_r}) \leq 1.$$

Tvrdime, ze plati (preco?)

$$n^{-d_1}+\cdots+n^{-d_k}+n^{-d_{k+1}} \leq 1.$$

Prenasobme celu nerovnost (kladnym) cislom $n^{d_{k+1}}$. Dostaneme

$$n^{d_{k+1}-d_1}+\cdots+n^{d_{k+1}-d_k}+n^{d_{k+1}-d_{k+1}} \leq n^{d_{k+1}}.$$

Podla lemmy B mozeme nerovnost prepisat nasledovne:

$$\mid Z(1,k{+}1) \mid + \cdots + \mid Z(k,k{+}1) \mid +1 \leq \mid B \mid^{d_{k+1}}.$$

Ak pouzijeme este lemmu C, tak ziskame

$$Z(1,k+1) \cup \cdots \cup Z(k,k+1) \cup Y \subseteq B^{d_{k+1}},$$

kde $Y \neq \emptyset$ je dizjunktna so vsetkymi mnozinami $Z(1, k+1), \cdots, Z(k, k+1)$. Vyberme slovo **c** $\in Y$. Polozme teraz

$$\varphi(a_{k+1}) = \mathbf{c}.$$

Na zaklade indukcie sme ziskali zobrazenie $\varphi : A \to B^*$. Taktiez vidime, ze toto zobrazenie je proste. Totiz, ked nase zobrazenie je proste na podmnozine $\{a_1, \cdots, a_k)$, tak sa tato vlastnost zachova aj na najblizsej vacsej mnozine $\{a_1, \cdots, a_k, a_{k+1}\}$. (Pretoze **c** $\in Y$.) Prefixovost zobrazenia φ sa overi podobne. Vieme, ze staci ukazat: $\varphi(a_i)$ nie je prefixom $\varphi(a_j)$ pre akekolvek $i < j$. Znova z indukcie vidime, ze je to pravda, ak $1 \leq i \leq k$ a $j = k + 1$. Dokoncte dokaz!

Poznamka 3. Ak kod splnuje Kraftovu nerovnost, tak to este neznamena, ze je prefixovy! Staci sa pozriet na priklad 2 o vode. Kodove slova boli 0, 01, 110, 111. Dostaneme 1/2 +

$1/4 + 1/8 + 1/8$=1. Nerovnost je splnena, ale kod nie je prefixovy. Vsimnime si, ze aj sufixovy kod splnuje Kraftovu nerovnost. Pokuste sa vyslovit a dokazat analogicku vetu pre tieto kody.

McMillanova veta. (5. lekcia)

Veta 3.(McMillanova) Kazde jednoznacne dekodovatelne kodovanie splnuje Kraftovu nerovnost. (Pozri oznacenia z vety 2.)

Dokaz. Nech $\varphi : A \to B^*$ je jednoznacne dekodovatelny kod s dlzkami kodovych slov

$$1 \leq d_1 \leq \cdots \leq d_r.$$

Chceme dokazat, ze plati

$$c = n^{-d_1} + \cdots + n^{-d_r} \leq 1.$$

Najprv sa pozrieme na mocniny cisla c. Budeme pocitat mocniny $c^1, c^2, \cdots, c^k, \cdots$. (To bol ten genialny napad!)

Najprv uvedieme pomocne tvrdenie:

Lemma A. Nech R je komutativny okruh. Nech $a_1, \cdots, a_r \in R$. Potom plati

$$(a_1 + \cdots + a_r)^k = \sum a_{i_1} \cdots a_{i_k},$$

pre vsetky $(i_1, \cdots, i_k) \in \{1, \cdots, r\}^k$.

Urobte dokaz ako D.U. Staci pouzit matematicku indukciu vzhladom na k. Vsimnime si, ze na pravej strane mame sucet clenov, ktore vznikli nasledovne: z kazdej zatvorky (na lavej strane) sme vybrali jeden prvok a to sme znasobili. Presnejsie, z prvej zatvorky sme vybrali i_1-clen, z druhej i_2-clen atd. az z $k-$tej zatvorky vyberieme i_k-clen a tie znasobime.

Tvrdime, ze plati (ak pouzijeme Lemmu A)

$$c^k = (n^{-d_1} + \cdots + n^{-d_r})^k =$$

$$= \sum_{i_1, \cdots, i_k = 1}^{r} n^{-(d_{i_1} + \cdots + d_{i_k})}.$$

(Vsimnime si, ze suma, aj napriek inemu zapisu, sa vztahuje na vsetky k-tice $(i_1, \cdots, i_k)$ z mnoziny $\{1, \cdots, r\}^k$.) Po uprave dostaneme

$$c^k = s_1 n^{-1} + s_2 n^{-2} + \cdots + s_{kd_r} n^{-kd_r} + \cdots + s_t n^{-t},$$

kde $s_1, \cdots, s_t$ su nezaporne cele cisla. Potrebujeme blizsie informacie o tychto cislach. Z toho, ze $1 \leq d_1 \leq \cdots \leq d_r$, mame

$$s_1 = \cdots = s_j = 0$$

pre $j < kd_1$ a podobne, $s_j = 0$ pre $kd_r < j$. Dohodnime sa na dalsom oznaceni

$$S_j = \{(i_1, \cdots, i_k) \in \{1, \cdots, r\}^k : d_{i_1} + \cdots + d_{i_k} = j\}.$$

Zrejme plati

$$\mid S_j \mid = s_j,$$

pre patricne prirodzene cisla j. (Ak napr. $(i_1, \cdots, i_k) \in S_j$ a $i_1 \neq i_2$, tak $(j_1, \cdots, j_k) \in S_j$ pre $j_1 = i_2$, $j_2 = i_1$ a $j_3 = i_3, \cdots, j_k = i_k$.) Teraz tvrdime, ze plati

$$s_j \leq n^j$$

pre kazde j. Potrebujeme to overit len pre

$$kd_1 \leq j \leq kd_r.$$

Ak $s_j = 0$, tak je to trivialne pravda. Nech teda $s_j \neq 0$. Potom $S_j \neq \emptyset$. Mozeme zaviest zobrazenie

$$\tau : S_j \to B^j$$

definovane predpisom

$$\tau : (i_1, \cdots, i_k) \mapsto \varphi^*(a_{i_1} \cdots a_{i_k}).$$

Naozaj, τ zobrazi S_j do B^j, pretoze

$$\varphi^*(a_{i_1} \cdots a_{i_k}) = \varphi(a_{i_1}) \mid \cdots \mid \varphi(a_{i_k})$$

a

$$\mid \varphi^*(a_{i_1} \cdots a_{i_k}) \mid = d_{i_1} + \cdots + d_{i_k} = j.$$

Podla predpokladu je φ jednoznacne dekodovatelne, t.j. φ^* je proste zobrazenie. V lemme A vety 2 sme ukazali, ze $\mid B^j \mid = n^j$. Vzhladom na to, ze τ je proste zobrazenie, tak dostavame

$$\mid S_j \mid = s_j \leq n^j,$$

co sme chceli ukazat. Zaverom dostavame

$$c^k = s_1 n^{-1} + \cdots + s_{kd_r} n^{-kd_r} \leq$$

$$\leq (n^1 \cdot n^{-1}) + \cdots + (n^{kd_r} \cdot n^{-kd_r}) =$$

$$= 1 + 1 \cdots + 1 = kd_r.$$

Z posledneho vztahu vidime, ze

$$\frac{c^k}{k} \leq d_r.$$

Inac povedane, postupnost

$$c, \quad \frac{c^2}{2}, \ldots, \quad \frac{c^k}{k}, \ldots$$

je zhora ohranicena cislom d_r. Potrebujeme este jednu pomocnu vetu

Lemma B. (Marquis de l'Hospital) Nech realne funkcie $f(x)$ a $g(x)$ su definovane na intervale $[a, \infty)$ a nech

$$\lim_{x \to \infty} f(x) = \infty, \quad \lim_{x \to \infty} g(x) = \infty.$$

(i) Nech dalej existuju derivacie $f'(x)$ a $g'(x)$ na intervale $[a,\infty)$, pricom $g'(x) \neq 0$ pre vsetky body intervalu a

(ii) nech existuje (konecna alebo nekonecna)

$$\lim_{x\to\infty} \frac{f'(x)}{g'(x)} = K.$$

Potom

$$\lim_{x\to\infty} \frac{f(x)}{g(x)} = K.$$

Mozeme sa vratit k dokonceniu nasho dokazu. Postupnost

$$\left\{ \frac{c^k}{k} \right\}$$

je podpostupnostou funkcie $\frac{c^x}{x}$ na intervale $[a,\infty)$ pre lubovolne $0 < a < 1$. Ukazeme, ze plati $0 < c \leq 1$. Predpokladajme opak. Pretoze trivialne $0 < c$, tak ostava $1 < c$. Su splnene vsetky predpoklady de l'Hospitalovho kriteria. Zrejme $(c^x)' = c^x \cdot log(c)$. Dalej, $x' = 1$. Teda (v oznaceni lemmy B) mame $K = \infty$. Preto podla lemmy B plati

$$\lim_{x\to\infty} \frac{c^x}{x} = \infty$$

a tym skor to plati pre podpostupnost

$$\lim_{k\to\infty} \frac{c^k}{k} = \infty.$$

Dostali sme vsak spor, pretoze vieme, ze nasa postupnost je zhora ohranicena cislom d_r a preto nemoze mat za limitu ∞. Teda, $c \leq 1$, t.j. plati Kraftova nerovnost.

Priklady 4-6. Predpokladajme, ze $A = \{a, b, c, d, e, f\}$. Nech $\varphi_1(A)$ je po rade $\{00, 10, 001, 101, 011, 111\}$. Kraftova nerovnost: $2 \cdot 2^{-2} + 4 \cdot 2^{-3} = 1$. Na zaklade McMillanovej vety nevieme rozhodnut, ci φ_1 je jednoznacne dekodovatelne. Urcite nie je prefixovy, ale je sufixovy. To dava moznost φ_1 prerobit na prefixovy kod, co je uz jednoznacne dekodovatelne.

Nech v dalsom priklade je zdrojova abeceda A ta ista ako doteraz. Polozme

$$\varphi_2(A) = \{0, 02, 1, 12, 20, 21\}.$$

Pytame sa, ci kod je jednoznacne dekodovatelny. (Odpoved: Nie, nesplna Kraftovu nerovnost.)

V poslednom priklade ponechame A to iste, co doteraz. Nech

$$\varphi_3(A) = \{0, 1, 210, 211, 212, 1010\}.$$

Hoci φ_3 splna Kraftovu nerovnost, nejedna sa o jednoznacne dekodovatelny kod, pretoze mame

$$\varphi_3(f) = \varphi_3^*(baba).$$

Napriek tomu a vdaka Kraftovej nerovnosti mozeme φ_3 prerobit na prefixovy kod, co je uz jednoznacne dekodovatelne.

Bezpecnostne kody. (6. lekcia)

Predchadzajuce uvahy boli motivovane snahou skonstruovat "optimalny" kod z danych predpokladov. Existuju matematicke vety, ktore prinasaju na to odpovede. Nebudeme sa tym dalej zaoberat, uvedieme len jeden dolezity vysledok.

Predpokladajme, ze $A = \{a_1, \cdots, a_r\}$, $\mid B \mid = n \geq 2$ a nech p_i pre $1 \leq i \leq r$ znamena pravdepodobnost toho faktu, ze sa pismeno a_i vyskytne v nasej sprave. Podobne, nech d_i znamena dlzku kodoveho slova $\varphi(a_i)$. Mozeme zaviest novy pojem: *Stredna dlzka kodoveho slova* je cislo, ktore definujeme vztahom

$$d = d_1p_1 + d_2p_2 + \cdots + d_rp_r.$$

Ma vyznam hovorit o *najkratsom* n-znakovom kode $\varphi : A \to B^*$, ktory je prefixovy a ma najmensiu strednu dlzku kodoveho slova d. Plati

Veta 4. (D. Huffman). Za horeuvedenych predpokladov existuje najkratsi n-znakovy kod.

V dalsom budeme predpokladat, ze v nasom prenosovom kanali dochadza ku poruchам, hovorime o sume. Na zaciatku semestra sme hovorili o tom, ze kody potrebujeme na to, aby sme mohli prenasat "spravy" z jedneho miesta na druhe. Napr. mame na mysli televizny signal iduci od vysielaca cez eter ku satelitu a odtial ku prijimacu. Prenasa sa to elektromagentickymi vlnami, ktore podliehaju v eteri sumu (vplyv slnecneho ziarenia, pocasie a pod.). Podobne sa deje komunikacia pozemnej stanice so satelitom, alebo v pocitaci ukladanie dat v pamati.

Podla beznej predstavy funguje v praxi kodovanie nasledovne: Mame povedzme ulohu nastartovat motory vzdialeneho satelitu (rakety). Tento povel musime najprv zakodovat, t.j. prelozit do takej "reci", aby sa to mohlo poslat cez eter ku satelitu. Toto vykonaju inzinieri spolu s matematikmi. Ked je to hotove, tak zakodovanu spravu poslu pomocou elektromagnetickych vln smerom ku satelitu. Ten ich prijme a iste zariadenie prijatu spravu dekoduje, t.j. znova prelozi do reci pristrojov. Ked je to hotove, tak zapnu sa startery motorov. Kvoli prehladu, si to rozlozime na nasledovne body:

1. Priprava spravy s ulohou nastartovania motorov.

2. Zakodovanie spravy, aby sa mohla poslat cez eter ku satelitu. (Povedzme, ze sa jedna o binarny kod.)

3. Prijatie zakodovanej spravy satelitom.

4. Dekodovanie spravy, t.j. jej prelozenie do reci, ktoru "rozumie" satelit.

5. Vykonanie povelu.

Na jednom mieste dochadza casto ku porucham: V bode 3 ciha nebezpecie. Totiz nemame zarucene, ze satelit prijme tu istu zakodovanu spravu, ktoru sme mu poslali. V eteri dochadza ku roznym porucham. Jedna sa hlavne o dva druhy chyb: a) zamena vyslaneho znaku na iny znak, alebo b) vytvorenim znaku, ktory vobec nebol vyslany (porucha synchronizacie). O tychto chybach nebudeme tu hovorit. Pojde len o chyby prveho druhu.

Chceme poukazat na to, ze prijimac spravy moze chyby prveho druhu *objavit* a v priaznivom pripade aj *opravit*. Ako sa to urobi? Vsimnime si jeden trivialny priklad z praxe: Predpokladajme, ze sme vyslali slovenske slovo "opakovanie". Prijali sme vsak divne slovo "opakkvanie". To, co sme prijali nie je "kodovym" slovom, pretoze take slovo sa nenachadza v slovniku slovenskeho jazyka. Teda, objavili sme chybu. V tomto pripade vieme dokonca urobit viacej, totiz chybu aj napravit. Existuje len jedno slovenske slovo, z ktoreho zmenou jedneho pismena ziskame nase prijate slovo. Je to, ako uz vieme slovo "opakovanie". Ak by sme totiz boli prijali slovo "opakkvanir", tak sme zaregistrovali chybu, ale ju uz nevieme opravit. Preco? Pozrime sa blizsie na to.

Objavovanie chyb

Vo vseobecnosti mame dobry prehlad o kodovych slovach (to su tie slova v slovniku!).

Definicia 7. Nech $\varphi : A \to B^*$ je kod. Ak prijmeme nekodove slovo, tak hovorime, ze sme *objavili* chybu.

Samozrejme, ak sme prijali kodove slovo, tak bud nedoslo ku chybe, alebo doslo ku chybe, ale my ju nevieme ako objavit. Ku pochopeniu tejto skutocnosti nam pomozu nasledovne definicie

Definicia 8. Nech $t \geq 1$ je prirodzene cislo. Hovorime o t-*nasobnej chybe*, ak pocet chybnych (zmenenych) miest v prijatom slove je aspon 1 a nanajvys t. V pripade $t = 1$ hovorime o *jednoduchych* chybach.

Definicia 9. Hovorime, ze kod $\varphi : A \to B^*$ *objavuje* $t-$nasobne chyby, ak pri vyslani kodoveho slova a vzniku t-nasobnej chyby prijmeme **vzdy** *nekodove* slovo.

V dalsom sa budeme zaoberat len blokovymi kodmi.

Definicia 10. Zoberme dve slova $\mathbf{a} = a_1 \cdots a_n$, $\mathbf{b} = b_1 \cdots b_n \in A^n$. Potom definujeme *vzdialenost* medzi slovami ako

$$\rho(\mathbf{a}, \mathbf{b}) =| \{i : a_i \neq b_i\} | .$$

Budeme to volat *Hammingovou* vzdialenostou.

Veta 5. Hammingova vzdialenost je metrikou na mnozine slov A^n.

Dokaz. Nech $\mathbf{u}, \mathbf{v}, \mathbf{w} \in A^n$, pricom $\mathbf{u} = u_1 \cdots u_n$ atd. Treba dokazat, ze

(1) $\rho(\mathbf{u}, \mathbf{v}) \geq 0$ a $\rho(\mathbf{u}, \mathbf{v}) = 0$ prave vtedy, ked

$$\mathbf{u} = \mathbf{v},$$

(2) $\rho(\mathbf{u}, \mathbf{v}) = \rho(\mathbf{v}, \mathbf{u})$ a

(3) $\rho(\mathbf{u}, \mathbf{v}) \leq \rho(\mathbf{u}, \mathbf{w}) + \rho(\mathbf{w}, \mathbf{v})$.

Staci overit (3), lebo ostatne body su jasne. Vyplyva to z tejto uvahy: ak $u_i \neq v_i$, tak bud $u_i \neq w_i$ alebo $w_i \neq v_i$.

Definicia 11. Nech $\varphi : A \to B^n$ je blokovy kod. (Zrejme $\varphi(A) \subseteq B^n$, kde $\varphi(A)$ je mnozina kodovych slov.) Nech

$$d = d_\varphi = \min\{\rho(\mathbf{v}, \mathbf{w}) : \mathbf{v}, \mathbf{w} \in \varphi(A), \mathbf{v} \neq \mathbf{w}\}.$$

Cislo d_φ volame *minimalnou vzdialenostou* kodu φ. (Toto cislo existuje, lebo kodovych slov je konecne vela!)

Veta 6. Nech $\varphi : A \to B^n$ je blokovy kod o minimalnej vzdialenosti d_φ. Potom φ objavuje t-nasobne chyby pre $t < d_\varphi$ a nie je schopny uz objavit vsetky t-nasobne chyby pre $t \geq d_\varphi$.

Dokaz. Predpokladajme, ze sme vyslali kodove slovo $\mathbf{u} \in \varphi(A) \subseteq B^n$ a namiesto neho sme prijali slovo $\mathbf{w} \in B^n$, pricom doslo ku t-nasobnej chybe, kde $t < d_\varphi$. Pretoze

$$\rho(\mathbf{u}, \mathbf{w}) = t < d_\varphi,$$

tak $\mathbf{w}$ je nekodove slovo. Objavili sme chybu. Ak by $t = d_\varphi$, tak nemame zaruku, ze $\mathbf{w}$ je nekodovym slovom, pretoze **w** by mohlo byt kodovym slovom, ktore ma minimalnu vzdialenost od **u**. Podobne je to aj v pripadoch, ked $t \geq d\varphi$.

Priklad 7. *Kod celkovej kontroly parity*. Predpokladajme, ze mame binarny blokovy kod $\varphi : A \to B^n$. Zrejme $d = d_\varphi \geq 1$. Kod φ prerobime na kod ψ, u ktoreho jednoduchsie skontrolujeme vyskyt chyb. Nech novy kod ma tvar

$$\psi : A \to B^{n+1},$$

pricom nove kodove slova definujeme nasledovne: Ak $\varphi(a) = v_1 \cdots v_n \in \varphi(A)$, tak

$$\psi(a) = v_1 \cdots v_n v_{n+1},$$

kde $v_{n+1} = 1$, ak pocet cislic "1" medzi

$$v_1, v_2, \cdots, v_n$$

je neparny. V opacnom pripade polozime

$$v_{n+1} = 0.$$

(Hovorime, ze sme dodali kontrolny znak.) Ukazeme, ze $d_\psi \geq 2$. Urobme to nepriamo. Nech $\mathbf{v}, \mathbf{w} \in \psi(a)$. Predpokladajme, ze $\rho(\mathbf{v}, \mathbf{w}) = 1$. Bez ujmy na vseobecnosti mozeme predpokladat, ze $v_1 = 1$, $w_1 = 0$,

$$v_2 = \cdots = v_k = w_2 = \cdots = w_k = 1$$

a sucasne

$$v_{k+1} = \cdots = v_{n+1} = w_{k+1} = \cdots = w_{n+1} = 0.$$

Obe slova maju parny pocet "1". Preto k je parne cislo. Lenze $\mathbf{w}$ ma $k-1$ zloziek rovnych 1, co je neparne cislo. Dostali sme spor. Teda plati $\rho(\mathbf{v}, \mathbf{w}) \geq 2$.

Priklad 8. *Opakovaci kod.* Nech $n \geq 2$. Mozeme vytvorit nasledovny pozoruhodny kod $\varphi : B \to B^n$ definovany vztahom

$$\varphi(b) = b_1 \cdots b_n \in B^n,$$

kde $b = b_1 = \cdots = b_n$. Volame ho *opakovacim* kodom. Rychle sa presvedcime, ze $d_\varphi = n$.

(Vidime, ze pre kazde n mame kod s minimalnou vzdialenostou rovnou n.) Neskorsie si uvedieme sposob, ako mozeme pomocou opakovacieho kodu jednoduchym sposobom zvacsovat minimalnu vzdialenost kodov.

Opravovanie chyb. (7. lekcia)

Zacneme s dvoma vseobecnymi poznamkami, ktore by mali posluzit na lepsie pochopenie dalsieho textu.

Poznamka 1. U oboch poslednych kodov (priklady 7 a 8) mame znaky dvojakeho druhu: *informacne* a *kontrolne*. U kodu celkovej kontroly parity je $(n+1)$-vy znak kontrolny, ostatne su informacne. U opakovacieho kodu je jeden znak informacny a ostatne su kontrolne. Presnejsie, prvy znak je informacny a zvysne su kontrolne. Vseobecne sa tomuto postupu hovori, ze sme dodali ku informacii *redundanciu*. Pouzijeme to na odhalovanie, ci opravovanie chyb. (Aj v hovorovej reci sa u dlhsich slov lahsie odhalia, resp. opravia chyby.) Napr. kod celkovej kontroly parity sa pouzival pre $n = 8$ pri mnohych pocitacoch na komunikaciu medzi klavesnicou a pocitacom. Je to v ramci tzv. ASCII kodu na vytvaranie pismen, cislic a dalsich znakov, pricom posielame zvlastne binarne 8-tice (=bytes) s dodatocnym kontrolnym znakom. Ak taky "byte" spolu s kontrolnym znakom by presiel s neparnym poctom znakov "1", tak to pocitac neakceptuje, t.j. nezareaguje, pretoze objavil chybu. (Ukon sa musi zopakovat.) Stane sa tak na zaklade nasho kodu, ktory odhaluje jednoduche chyby. (Pripominame, ze ASCII znamena American Standard Code for Information Interchange.)

Poznamka 2. Samozrejme, ze okrem ASCII kodu existuju aj dalsie, podobne, napr. IBM kod. Povodny ASCII kod mal 128 (=2^7) (kodovych) slov. Uvedieme niekolko kodovych slov (bez kontrolneho znaku): 033: 00100001 (=!); 048: 00110000 (=0); 053: 00110101 (=5); 065: 01000001 (=A); 090: 01011010 (=Z); 097: 01100001 (=a); 098: 01100010 (=b); 122: 01111010 (=z); 127: 01111111 (=DEL).

Definicia 12. Nech $\varphi : A \to B^n$ je blokovy kod. Hovorime, ze φ *opravuje* $t - nasobne$ chyby, ak pri vyslani kodoveho slova $\mathbf{u} \in \varphi(A)$ a pri prijati slova $\mathbf{w} \in B^n$ plati:

(i) $\rho(\mathbf{u}, \mathbf{w}) \leq t$ a

(ii) $\rho(\mathbf{u}, \mathbf{w}) < \rho(\mathbf{x}, \mathbf{w})$

pre kazde $\mathbf{x} \in \varphi(A)$ a $\mathbf{u} \neq \mathbf{x}$.
(Inac povedane, pozadujeme, aby $\mathbf{u}$ bolo jedine

kodove slovo, ktore ma najmensiu vzdialenost ku prijatemu slovu **w**.)

Poznamka 3. Definicia 12 je vlastne aj navodom, ako mame dekodovat φ. Pojde o parcialne zobrazenie

$$\delta : B^n \to \varphi(A),$$

kde plati

(i) $\delta(\mathbf{u}) = \mathbf{u}$ pre $\mathbf{u} \in \varphi(A)$ a

(ii) $\delta(\mathbf{w}) = \mathbf{u}$, ak slova **u** a **w** splnuju poziadavky definicie 12. Inac, $\delta(\mathbf{w})$ nie je definovane.

Tento sposob dekodovania sa vola niekedy po anglicky ”maximum-likelihood-decoding”.

Veta 7. Blokovy kod φ minimalnej vzdialenosti $d = d_\varphi$ opravuje $t-$nasobne chyby pre vsetky

$$t < d/2.$$

Dokaz. Nech $\mathbf{u} \in \varphi(A) \subseteq B^n$ a nech $\mathbf{w} \in B^n$, pricom $\rho(\mathbf{u}, \mathbf{w}) \leq t < d/2$. Predpokladame, ze sme vyslali slovo $\mathbf{u}$ a prijali slovo $\mathbf{w}$.
Pre kazde kodove slovo $\mathbf{x} \in \varphi(A) - \{\mathbf{u}\}$ plati

$$d = d_\varphi \leq \rho(\mathbf{u}, \mathbf{x}).$$

Z trojuholnikovej nerovnosti

$$\rho(\mathbf{u}, \mathbf{x}) \leq \rho(\mathbf{u}, \mathbf{w}) + \rho(\mathbf{w}, \mathbf{x})$$

dostavame

$$d/2 \leq d - \rho(\mathbf{u}, \mathbf{w}) \leq \rho(\mathbf{u}, \mathbf{x}) - \rho(\mathbf{u}, \mathbf{w}) \leq \rho(\mathbf{w}, \mathbf{x}).$$

Vyuzijuc predpoklad mame

$$\rho(\mathbf{u}, \mathbf{w}) \leq t < d/2 \leq \rho(\mathbf{w}, \mathbf{x}) = \rho(\mathbf{x}, \mathbf{w}),$$

pre kazde $\mathbf{x} \in \varphi(A)$ a $\mathbf{x} \neq \mathbf{u}$.

Poznamka 4. Podobne ako v pripade vety 6 sa pytame, ci tvrdenie vety 7 neplati aj pre $t \geq d/2$. Ukazeme na "kontrapriklade", ze to uz nejde. Presnejsie, ukazeme len, ze $t = d/2$ vedie k sporu. (Pripad $t > d/2$ sa vybavi podobne.) Zoberme $\mathbf{u}, \mathbf{v} \in \varphi(A)$, pricom pozadujeme navyse, ze

$$d = d_\varphi = \rho(\mathbf{u}, \mathbf{v}).$$

Predpokladajme este, ze $\{i_1, \cdots, i_d\}$ su tie indexy i, pre ktore je $u_i \neq v_i$. Vyberieme slovo $\mathbf{w} = w_1 \cdots w_n \in B^n$ takto

$$w_i = \begin{cases} v_i & \text{ak } i \in \{i_2, i_4, \cdots, i_d\} \\ u_i & \text{inac} \end{cases}$$

Zrejme,

$$\rho(\mathbf{u}, \mathbf{w}) = d/2 = \rho(\mathbf{v}, \mathbf{w}).$$

Teraz vidime, ze ak vysleme $\mathbf{u}$ a prijmeme $\mathbf{w}$, tak neplati poziadavka

$$\rho(\mathbf{u}, \mathbf{w}) < \rho(\mathbf{x}, \mathbf{w})$$

pre $\mathbf{x} = \mathbf{v}$.

Priklad 9. *Kod dvojrozmernej kontroly parity.* Ide o prakticky dolezity binarny kod, ktory sa pouziva pri pocitacoch. Kodove slova sa zapisuju v tvare matice: kazdy riadok ma pridany symbol kontroly parity, kazdy stlpec ma tiez dole znak kontroly parity a dole v pravom rohu je este znak kontroly parity celej matice. (Jedna sa zase o ASCII kod.) Uvedieme jeden konkretny pripad. Pojde o maticu typu 8×4, pricom informacne znaky sa nachadzaju v podmatici (vlavo hore) typu 7×3.

Tento kod opravuje jednoduche chyby. Totiz, ak objavime riadkovu chybu v $i-$tom riadku a stlpcovu chybu v $j-$tom stlpci, tak chyba nastala na suradnici (i, j). Tam zmenime znak na opacny a oprava je vykonana. (Zistili sme to aj bez toho, ze by sme poznali jeho minimalnu vzdialenost.) Z toho, ze φ opravi jednoduche chyby, vidime, ze pre minimalnu vzdialenost platí $d_\varphi \geq 3$ (vid vetu 7).

$$\left(\begin{array}{ccc|c} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ \hline 1 & 1 & 0 & 0 \end{array}\right)$$

Z doterajsich prikladov sme uz zistili (opakovaci kod alebo kod celkovej kontroly parity), ze z dodanim vhodnych znakov ku existujucim slovam, mozeme zlepsit kvalitu prijatych sprav. To nas vedie k tomu, ze znaky v kodovych slovach delime na *informacne* (tie, ktore mozeme lubovolne urcit) a na *kontrolne* (tie, ktore su uplne urcene informacnymi znakmi). U blokovych kodov to mozeme este upresnit v nasledujucom zmysle

Definicia 12. Nech $\varphi(A) = K \subseteq B^n$ je blokovy kod. Ak existuje take cislo $k \leq n$ a bijektivne zobrazenie $\psi : B^k \to K$, tak hovorime, ze kod K ma k *informacnych* a $n-k$ *kontrolnych* znakov. Zobrazenie ψ volame *kodovanim informacnych znakov*.

Definicia 13. Blokovy kod $K \subseteq B^n$ volame *systematickym*, ak existuje take cislo $k < n$, ze zobrazenie $\psi : B^k \to K$ definovane vztahom

$$\psi : v_1 \cdots v_k \mapsto v_1 \cdots v_k v_{k+1} \cdots v_n$$

je kodovanim informacnych znakov. Volame ho *systematickym*.

Priklad 10. Rychle sa presvedcime, ze specialny pripad kodu celkovej kontroly parity $\varphi : B^n \to B^{n+1}$ je systematicky s $k = n$, ak pre $\mathbf{u} \in B^n$ definujeme

$$\mathbf{u} = u_1 \cdots u_n \mapsto \varphi(\mathbf{u}) = u_1 \cdots u_n u_{n+1},$$

pricom u_{n+1} je kontrolnym znakom. Podobne je to s opakovacim kodom. Tu je prvy znak informacny, t.j. $k = 1$ a zvysnych $n-1$ znakov je kontrolnych. Patricne bijektivne zobrazenie sa jednoducho najde (pozri tiez poznamku 1). Samozrejme, existuju aj nesystematicke kody. Takym je napr. "koktavy" kod. Opiseme si ho. Definujeme $\varphi : B^3 \to B^6$ nasledovne:

$$\varphi : \mathbf{u} = u_1u_2u_3 \mapsto \varphi(\mathbf{u}) = u_1u_1u_2u_2u_3u_3.$$

Vsimnime si, ze tento kod ma 3 informacne znaky a tiez 3 kontrolne znaky, ale nie je systematicky.

Dolezity je aj nasledovny pojem

Definicia 14. Nech $\varphi : A \to B^n$ je blokovy kod. Nech k je pocet informacnych znakov.

Potom cislo

$$R = k/n$$

volame *informacnym pomerom* kodu φ.

Veta 8. Nech d znamena minimalnu vzdialenost systematickeho kodu $\varphi : A \to B^n$. Potom

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

Dokaz. Podla definicie systematickeho kodu vieme, ze existuje k s vlastnostou: mame take bijektivne zobrazenie $\psi : B^k \to \varphi(A)$, ze

$$\psi : v_1 \cdots v_k \mapsto v_1 \cdots v_k v_{k+1} \cdots v_n.$$

Vyberme si slovo $\mathbf{v} = v_1 \cdots v_{k-1} \in B^{k-1}$. Nech K_0 je mnozina vsetkych slov z $\varphi(A)$, ktore maju za prefix slovo $\mathbf{v}$. Potom pre minimalnu vzdialenost d_0 slov z K_0 plati

$$d_0 \leq n - (k-1) = n - k + 1.$$

Pretoze $K_0 \subseteq \varphi(A)$, tak mame $d \leq d_0$.

Linearne kody. (8. lekcia)

Konecne polia.

Najprv musime si nieco zopakovat o konecnych poliach. Vieme, ze na mnozine $Z_n = \{0, \cdots, n-1\}$ (n je prirodzene cislo) mozeme zaviest dve binarne operacie, ktore tu budeme oznacovat (ak to bude potrebne!) ako $\oplus$ a $\star$ a budeme to volat *scitovanim* a *nasobenim* modulo n. Klasicky vysledok z algebry hovori

Veta 9. Struktura $Z_n = (Z_n; \oplus, \star)$ tvori komutativny okruh s jednotkou. (Volame ho *okruhom zvyskovych tried* modulo n.) Tento okruh je polom prave vtedy, ked n je prvocislo.

Takychto poli mame nekonecne vela: $Z_2, Z_3, \cdots$, $Z_p, \cdots$. To nie je vsetko! Plati este

Veta 10. Nech F je konecne pole charakteristiky p. Potom $\mid F \mid = p^k$. (Vieme, ze p je prvocislo.)

Veta 11. Nech p je dane prvocislo a nech n je lubovolne prirodzene cislo. Potom existuje pole F, pre ktore plati: $\mid F \mid = p^n$.

Veta 12. Nech F a E su konecne polia. Potom $F \cong E$ prave vtedy, ked $\mid F \mid = \mid E \mid$.

Veta 13. Nech F je pole. Nech $F[x]$ je okruh polynomov v neurcitej x nad polom F. Nech $p(x) \in F[x]$ je ireducibilny polynom nad F. Potom $E = F[x]/(p(x))$ je pole, ktore je izomorfne s nadpolom pola F. Specialne, nech
$q(x) \in Z_p[x]$ je ireducibilny polynom stupna n nad Z_p. Potom

$$\mid E \mid = \mid Z_p[x]/(q(x)) \mid = p^n.$$

Ak F znamena (konecne) pole, tak $\mathbf{F}^n$ bude oznacovat mnozinu vsetkych slov $\mathbf{a} = a_1 \cdots a_n$ dlzky n nad polom F. Samozrejme, na slova sa mozeme divat ako na vektory, t.j. usporiadane $n-$tice prvkov z F. Dalej vidime, ze na $\mathbf{F}^n$ mame definovanu binarnu operaciu $+$ nasledovne:

$$\mathbf{a} + \mathbf{b} = (a_1 + b_1) \cdots (a_n + b_n).$$

Rychle sa presvedcime (urobte to !), ze $(\mathbf{F}^n; +)$ je komutativna grupa. Mozeme ist dokonca dalej. Da sa jednoducho ukazat (preverte to!), ze dvojica $(F; \mathbf{F}^n)$ je konecny vektorovy priestor, ak nasobenie skalarmi definujeme nasledovne:

$$c\mathbf{a} = (ca_1) \cdots (ca_n)$$

pre kazde $c \in F$. Taktiez vidime, ze konecny priestor automaticky znamena konecno-rozmerny vektorovy priestor.

V dalsom budeme stale predpokladat, ze F je konecne pole. Budeme sa zaoberat kodami tvaru $\varphi : A \to \mathbf{F}^n$ v zmysle nasledujucej definicie.

Definicia 15. Nech F je konecne pole. Potom proste zobrazenie $\varphi : A \to \mathbf{F}^n$ volame *linearnym kodom*, ak mnozina kodovych slov

$$K = \varphi(A) \subseteq \mathbf{F}^n$$

je podpriestorom vektoroveho priestoru $(F; \mathbf{F}^n)$. Dalej, ak sme presnejsi, tak hovorime, ze φ je linearnym $(n,k)-$kodom, ked

$$\dim(\varphi(A)) = k.$$

Zvlastna situacia nastane, ked $k = 0$ alebo $k = n$. Vtedy hovorime o *trivialnom* kode. Vacsinou sa budeme zaoberat netrivialnymi kodami. Dalsia zvlastnost nastane, ked $F = Z_2$. Bude to vlastne binarny linearny kod. Pripomenieme si este jednu dolezitu vetu z linearnej algebry. K tomu potrebujeme oznacenia. Obycajne znamena $\mathtt{A}$ nejaku maticu typu $m \times n$, t.j. ktora ma

m riadkov a n stlpcov. Specialne, $\mathtt{a}$ moze znamenat riadkovu maticu, resp. slovo, dlzky n, co sme doteraz zapisovali aj ako **a**. Znakom $\mathtt{A}^T$ zapisujeme transponovanu maticu ku $\mathtt{A}$, t.j. ak v $\mathtt{A}$ navzajom vymenime riadky za odpovedajuce stlpce.

Veta 14. Nech

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = 0$$

$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = 0$$

$$\cdots$$

$$a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = 0.$$

je system linearnych homogennych rovnic nad polom F s maticou sustavy $\mathtt{A}$. Potom riesenia tohto systemu tvoria podpriestor vektoroveho priestoru $(F; \mathbf{F}^n)$. Dalej, ak $h(\mathtt{A}) = r$ (=hodnost matice sustavy), tak $n - r = k$ je dimenziou podpriestoru rieseni.

Horeuvedeny system rovnic mozeme zapisat aj pomocou matic. Vyzera to nasledovne:

$$\mathtt{A}\mathtt{x}^T = \mathtt{0}^T.$$

Priklad 11. Nech $F = Z_2$. Uvazujme o systeme linearnych homogennych rovnic

$$x_1 + \cdots + x_n = 0.$$

Jeho riesenim su vsetky slova (vektory) $\mathtt{v} = v_1 \cdots v_n \in \mathbf{Z}_2^n$ s vlastnostou $v_1 + \cdots + v_n = 0$. Teda sa jedna o kod celkovej kontroly parity, ktory je linearnym $(n, n-1)-$kodom.

Priklad 12. Nech F je konecne pole. Opakovaci kod dlzky n je linearny $(n, 1)-$kod nad F a da sa opisat nasledovnym homogennym systemom linearnych rovnic

$$x_1 + (-1)x_2 = 0$$

$$x_1 + (-1)x_3 = 0$$

$$\cdots$$

$$x_1 + (-1)x_n = 0.$$

Priklad 13. Nech F je konecne pole. "Koktavy" kod (pre $n = 6$) je (6,3)-linearnym kodom nad F a da sa opisat systemom rovnic

$$x_1 + (-1)x_2 = 0$$

$$x_3 + (-1)x_4 = 0$$

$$x_5 + (-1)x_6 = 0.$$

Priklad 14. Nie kazdy kod je linearny. Napr. kod "2 z 5" nie je linearnym kodom, pretoze tam nepatri nulove slovo. Pripominame, ze kod "2 z 5" vyzera nasledovne: $A = \{0, 1 \cdots, 9\}$ a $1 \mapsto 11000$, $2 \mapsto 10100$, $3 \mapsto 10010$, $4 \mapsto 10001$, $5 \mapsto 01001$, $6 \mapsto 00101$, $7 \mapsto 00011$, $8 \mapsto 00110$, $9 \mapsto 01100$, $0 \mapsto 01010$.

Generujuca matica. (9. lekcia)

Ukazeme, ze linearny kod mozeme charakterizovat dvoma maticami: generujucou a kontrolnou. Dalej ukazeme, ako sa z jednej matice da urcit prislusna druha matica. Vsimnime si najprv jednu dolezitu vlastnost linearnych kodov. Ak $\varphi : A \to \mathbf{F}^n$, je linearny kod, tak $\varphi(A)$ je podpriestorom vektoroveho priestoru $(F; \mathbf{F}^n)$. Pretoze φ je proste zobrazenie, tak plati:

$$| A |=| \varphi(A) | .$$

Dalej, vieme ze $\dim(\varphi(A)) = k$. Teda, $\varphi(A)$ ma aspon jednu k-prvkovu bazu $\mathrm{g}_1, \cdots, \mathrm{g}_k$. Pretoze $\mathrm{g}_i \in \mathbf{F}^n$, tak slova g_i prepiseme nasledovne:

$$\mathrm{g}_1 = g_{11}g_{12}\cdots g_{1n},$$

$$\mathrm{g}_2 = g_{21}g_{22}\cdots g_{2n},$$

$$\cdots$$

$$\mathrm{g}_k = g_{k1}g_{k2}\cdots g_{kn}.$$

Z tychto hodnot mozeme vytvorit maticu $\mathtt{G} = (g_{ij})$ typu $k \times n$.

Definicia 16. Horeuvedenu maticu $\mathtt{G} = (g_{ij})$ budeme volat *generujucou* maticou linearneho kodu φ.

Poznamka Generujuca matica nie je jednoznacne urcena. Pretoze existuje viacej baz podpriestoru $\varphi(A)$, tak preto existuje aj viacej generujucich matic. Vsetky generujuce matice kodu φ maju nasledovne vlastnosti:

(i) pocet riadkov je $k = \dim(\varphi(A))$ a n je pocet stlpcov;

(ii) riadky generujucej matice su linearne nezavisle (ako vektory);

(iii) kazdy riadok matice je kodovym slovom;

(iv) kazde kodove slovo kodu φ je linearnou kombinaciou riadkov generujucej matice.

Veta 15. Nech $\varphi : A \to \mathbf{F}^n$ je linearny (n, k)-kod a nech $\mathbf{b}_1, \cdots, \mathbf{b}_k$ je baza $\varphi(A) = K$. Potom zobrazenie $\psi : \mathbf{F}^k \to K$ definovane predpisom

$$\psi : u_1 \cdots u_k \mapsto u_1\mathbf{b}_1 + \cdots + u_k\mathbf{b}_k$$

je kodovanim informacnych znakov a sucasne je linearne.

Dokaz. Kedze φ je linearny (n, k)-kod, tak K ma horeuvedenu k-prvkovu bazu a lubovolny prvok $\mathbf{u} \in K$ sa da podla nej jednoznacne zapisat v tvare, aky je uvedeny pri tvorbe zobrazenia ψ. Teraz uz rychle preverime, ze ψ je proste linearne zobrazenie vektorovych priestorov. Kedze vsetky mnoziny su konecne, tak zobrazenie ψ je izomorfizmom. Preverte to!

Zaroven sme tiez dokazali

Dosledok.

(i) $(F, K) \cong (F, \mathbf{F}^k)$ a

(ii) φ ma k informacnych a $n - k$ kontrolnych znakov.

Poznamka 1. Poucenia z predchadzajucej vety: Pri linearnom kodovani $\varphi : A \to \mathbf{F}^n$ mozeme mnozinu A vzdy povazovat za $\mathbf{F}^k$ pre vhodne k, kde k udava pocet informacnych znakov, alebo co je to iste, $k = \dim(\varphi(A))$. Ak teda $A = \mathbf{F}^k$, tak linearny kod $\varphi : A \to \mathbf{F}^n$ mozno zamenit inym linearnym kodom $\psi_1 : \mathbf{F}^k \to \mathbf{F}^n$, co je aj linearnym zobrazenim. Pritom ψ_1 je zlozenim linearnych zobrazeni

$$\psi : \mathbf{F}^k \to K \quad \text{a} \quad \mathrm{id}(K) : K \to \mathbf{F}^n.$$

Druhe zobrazenie je vnorenim, t.j. identickym zobrazenim K do $\mathbf{F}^n$. Preto zvykneme oznacovat $\psi_1 = \psi$. Dalej, ak $A = \mathbf{F}^k$, tak jednu bazu v K ziskame jednoducho: $\psi(10\cdots0), \cdots, \psi(0\cdots01)$. (Preco ?) Specialne, ak sa jedna este o binarny kod, t.j. $F = Z_2$, tak dostaneme $\mid A \mid = 2^k$.

Poznamka 2. V dalsom vidime, ze pri vybere (konstrukcii) linearneho kodu sa snazime dosiahnut toho, aby pocet informacnych znakov k bol velky (t.j. aby redundancia bola mala) a sucasne, aby minimalna vzdialenost kodu d bola velka, t.j. aby kod mohol objavovat a opravovat vela chyb. Tieto poziadavky su rozporne a preto hladame vhodny kompromis. Samozrejme, ze su este dolezite otazky realizacie nasich poziadaviek (t.j. vhodnych algoritmov na objavovanie a opravovanie chyb). Na tazkosti poukazuje uz aj veta 8.

Je zaujimave si vsimnut, ze systematicky linearny kod ma za generujucu maticu

$$\mathtt{G} = (\mathtt{E}_k \mid \mathtt{C})$$

typu $k \times n$, pricom $\mathtt{E}_k$ je jednotkova matica stupna k a $\mathtt{C}$ je maticou typu $k \times (n-k)$. Riadky matice $\mathtt{G}$ su slova $\varphi(10\cdots 0)$, $\cdots$, $\varphi(00\cdots 1)$. (Pozri poznamku za vetou 10.)

Definicia 17. Blokove kody K a K_1 sa nazyvaju *ekvivalentne*, ak existuje taka permutacia $\pi \in S_n$, ze

$$v_1 \cdots v_n \in K \text{ prave vtedy, ked } v_{\pi(1)} \cdots v_{\pi(n)} \in K_1.$$

Veta 16. Kazdy linearny kod je ekvivalentny s nejakym systematickym linearnym kodom.

Dokaz. Pouzijeme znamu metodu z 1. rocnika: uprava matice pomocou elementarnych riadkovych operacii. Pripominame, ze su tri zakladne operacie: I. vymena riadkov, II. vynasobenie riadku nenulovym prvkom z pola F a III. vynasobeny i-riadok pripocitame ku j-temu riadku. Pomocou riadkovych operacii vieme nasu generujucu maticu $\mathtt{G}$ upravit na tzv. schodikovy tvar. Presnejsie: 1. kazdy riadok je bud nulovy, alebo nie. V druhom pripade existuje prvy $a_{ij} \neq 0$, t.j. j je minimalne. Ziadame $a_{ij} = 1$ (=pivotny prvok); 2. v tom stlpci, v ktorom existuje pivotny prvok, existuju inac len nulove prvky; 3. najprv idu nenulove riadky a po nich nulove; 4. ak s_i a s_j su stlpcove indexy pivotnych prvkov z i-teho a j-teho riadku, tak $s_i < s_j$. Pretoze $h(\mathtt{G}) = k$, tak kazdy riadok novej matice je nenulovy. Kedze mame k nenulovych riadkov, tak mame k pivotnych prvkov. V ziadnom stlpci nelezia dva pivotne prvky. Teda, pivotne prvky lezia v stlpcoch

$$1 \leq j_1 < \cdots < j_k \leq n.$$

Postarame sa o to, aby pivotne prvky (co su 1-ky) lezali v prvych k stlpcoch. Urobime to pomocou vymeny stlpcov (transpozicii). T.j. vymenime j_1-vy stlpec s 1. stlpcom atd. Zlozenim vsetkych potrebnych transpozicii ziskame permutaciu π, ktoru potrebujeme urobit na stlpcoch. Tak zaverom dostaneme novu generujucu maticu

$$\mathtt{G}_1 = (\mathtt{E}_k \mid \mathtt{C}),$$

ktora je ekvivalentna s $\mathtt{G}$. Urobte podrobnejsi dokaz!

Priklad 15. Binarny kod celkovej kontroly parity dlzky 4 ma generujucu maticu

$$\mathtt{G} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}.$$

Vseobecne, kod celkovej kontroly parity dlzky n je binarny linearny $(n, n-1)-$kod (priklad 11). Jeho generujuca matica ma tvar: $\mathtt{G} = (\mathtt{E} \mid \mathtt{I})$, kde $\mathtt{E} = \mathtt{E}_{n-1}$ a $\mathtt{I}$ je stlpcovy vektor, v ktorom kazdy prvok sa rovna 1. Zrejme sa jedna o systematicky kod.

Priklad 16. U opakovacieho kodu (priklad 12) je generujuca matica typu $1 \times n$ a tvaru

$$\mathtt{G} = (111 \cdots 1).$$

Zase sa jedna o systematicky kod.

Priklad 17. Pozrime sa este na koktavy kod dlzky 6 (priklad 13). Jeho generujuca matica vyzera nasledovne:

$$\mathtt{G} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}.$$

Nie je to systematicky kod. Ekvivalentny systematicky kod dostaneme vymenou stlpcov: $2 \leftrightarrow 3$ a $3 \leftrightarrow 5$, co dava permutaciu $\pi = (23)(35) = (253)$. Najdite $\mathtt{G}_1$!

Priklad 18. Nech K je ternarny kod s abecedou $Z_3 = \{0, 1, 2\}$ a dlzky 6, v ktorom 3-ti znak sluzi ku kontrole prvych dvoch a siesty znak zase ku kontrole 4. a 5. znaku, t.j. plati

$$a_1 + a_2 = a_3, \; a_4 + a_5 = a_6.$$

K je linearny, pretoze sa da opisat systemom linearnych rovnic

$$x_1 + x_2 + 2x_3 = 0$$

$$x_4 + x_5 + 2x_6 = 0$$

Tento kod ma 4 informacne znaky. Generujucou maticou je

$$\mathtt{G} = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}.$$

Nas kod nie je systematicky. Podla vety 16 mozeme najst k nemu ekvivalentny systematicky kod. Potrebujeme uz len urobit patricnu permutaciu stlpcov: $3 \leftrightarrow 4$ a potom $4 \leftrightarrow 5$. Odpoveda to zlozeniu dvoch transpozicii (34) a (45). Teda $\pi = (34)(45) = (354)$. Prislusna generujuca matica $\mathtt{G}_1$ ekvivalentneho systematickeho kodu bude

$$\mathtt{G}_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

Kontrolna matica. (10. lekcia)

Definicia 18. Matica $\mathtt{H}$ typu $m \times n$ nad konecnym polom F sa nazyva *kontrolnou* maticou linearneho kodu $K \subseteq \mathbf{F}^n$, ak plati

$$\mathtt{H}\mathbf{v}^T = \mathsf{0}^T \quad \text{prave vtedy, ked} \quad \mathbf{v} \in K.$$

Inac povedane, $\mathtt{H}$ je kontrolnou maticou linearneho kodu $K \subseteq \mathbf{F}^n$, ak K je podpriestor rieseni homogenneho systemu rovnic

$$\mathtt{H}\mathbf{x}^T = \mathsf{0}^T.$$

(Samozrejme $\mathtt{0}$ je nulove slovo dlzky m.) Bez ujmy na vseobecnosti mozeme predpokladat, ze

$$m = h(\mathtt{H}) \leq n.$$

Ideme sa pozriet na vztah medzi generujucou a kontrolnou maticou jedneho a toho isteho linearneho kodu.

Definicia 19. Nech $(F, \mathbf{F}^n)$ je vektorovy priestor nad konecnym polom F. Nech $\mathbf{u}, \mathbf{v} \in \mathbf{F}^n$. Potom definujeme *skalarny* sucin vektorov (slov) nasledovne:

$$\mathbf{u}_* \mathbf{v} = u_1 v_1 + \cdots + u_n v_n,$$

pricom $\mathbf{u} = u_1 \cdots u_n$, a $\mathbf{v} = v_1 \cdots v_n$.

Lema 1. Pre skalarny sucin plati:

(i) $\mathbf{u}_* \mathbf{v} = \mathbf{v}_* \mathbf{u}$;

(ii) $\mathbf{u}_*(\mathbf{v}+\mathbf{w}) = \mathbf{u}_*\mathbf{v} + \mathbf{u}_*\mathbf{w}$;

(iii) $(c\mathbf{u})_*\mathbf{v} = c(\mathbf{u}_*\mathbf{v}) = \mathbf{u}_*(c\mathbf{v})$.

Poznamka. Jedna sa o tzv. degenerovany skalarny sucin, t.j. moze platit $\mathbf{u}_*\mathbf{u} = 0$ pre nejake $\mathbf{u} \neq \mathbf{0}$. Vezmime napr. $\mathbf{u} = 0101$ nad Z_2.

Definicia 20. Nech $\emptyset \neq K \subseteq \mathbf{F}^n$. Potom definujeme

$$K^\perp = \{\mathbf{v} \in \mathbf{F}^n : \mathbf{u}_*\mathbf{v} = 0 \quad \text{pre} \quad \text{vsetky} \quad \mathbf{u} \in K\}$$

a $K^\perp$ volame *dualnym* podpriestorom ku K.

Lema 2. $K^\perp$ je podpriestorom $\mathbf{F}^n$ pre $K \neq \emptyset$.

Dokaz. Zrejme $K^\perp \neq \emptyset$, pretoze $\mathbf{0} \in K^\perp$. Teraz sa uz rychle preveri, ze $K^\perp$ je podpriestorom $\mathbf{F}^n$. Urobte to!

Lema 3. Nech K je k-rozmerny podpriestor $\mathbf{F}^n$. Potom pre $K^\perp$ platí:

$$\dim(K^\perp) = n - k = n - \dim(K).$$

Dokaz. Nech $\mathbf{g}_1, \cdots, \mathbf{g}_k$ je baza K. Predpokladajme, ze

$$\mathbf{g}_i = g_{i1} \cdots, g_{in}$$

je tvar tychto vektorov v $\mathbf{F}^n$ pre $i = 1, \cdots, k$. (Ak K je linearny kod, tak $\mathtt{G} = (g_{ij})$ je generujucou maticou pre K.) Potom plati:
$\mathbf{x} = x_1 \cdots x_n \in K^\perp$ prave vtedy, ked

$$(\mathbf{g}_1)_* \mathbf{x} = \cdots = (\mathbf{g}_k)_* \mathbf{x} = 0.$$

Ak prepiseme posledne vztahy, tak dostaneme homogenny linearny system rovnic

$$g_{11}x_1 + \cdots + g_{1n}x_n = 0$$

$$\cdots$$

$$g_{k1}x_1 + \cdots + g_{kn}x_n = 0.$$

Teda, $\mathbf{x} \in K^{\perp}$ prave vtedy, ked $\mathbf{x}$ je riesenim $\mathtt{G}\mathbf{x}^T = \mathtt{0}^T$. Pretoze $h(\mathtt{G}) = k$, tak

$$\dim(K^{\perp}) = n - k$$

podla vety 14.

Zaroven sme dokazali este

Lema 4. Ak K je linearny kod s generujucou maticou $\mathtt{G}$, tak $\mathtt{G}$ je kontrolnou maticou dualneho kodu $K^{\perp}$.

Priklad 19. Nech $F = Z_2$. Nech $K = \{00000, 11111\} \subseteq \mathbf{F}^5$ je opakovaci kod. Potom $\mathbf{u} \in K^{\perp}$ prave vtedy, ked

$$u_1 + u_2 + u_3 + u_4 + u_5 = 0.$$

Teda, $K^{\perp}$ je kodom celkovej kontroly parity.

Veta 17. Dualny kod $K^{\perp}$ linearneho (n, k)-kodu K je $(n, n-k)$-kodom, t.j. $\dim(K^{\perp}) =$

$n-k$. Dalej, generujuca matica kodu K je kontrolnou maticou kodu $K^{\perp}$ a obratene.

Dosledok. Kazdy linearny kod ma kontrolnu maticu.

Dokaz vety. Velku cast vety sme uz dokazali v lemach 1-4. Vzhladom na komutativnost skalarneho sucinu (Lema 1) vidime, ze

$$K \subseteq K^{\perp\perp}.$$

Tvrdime, ze $K = K^{\perp\perp}$. Z liem 2 a 3 vyplyva

$$\text{dim}(K^{\perp\perp}) = n - \text{dim}(K^{\perp}) =$$

$$= n - (n-k) = k = \text{dim}(K).$$

Pretoze $\text{dim}(K) = \text{dim}(K^{\perp\perp})$ a K je konecnorozmerny (!!), tak $K = K^{\perp\perp}$. (Preco?)

Poznamka. Upozornujeme, ze dokaz tvrdenia $K = K^{\perp\perp}$ plati len pre konecnorozmerne K. Vo vseobecnom pripade plati len $K \subseteq K^{\perp\perp}$. Kontrapriklady sa najdu vo funkcionalnej analyze.

Generovanie versus kontrola. (11. lekcia)

Ukazeme este, ako sa ku generujucej matici vyrobi odpovedajuca kontrolna matica a obratene. Zacneme specialnym pripadom.

Veta 18. Systematicky linearny kod K s generujucou maticou $\mathtt{G} = (\mathtt{E}_k \mid \mathtt{B})$ typu $k \times n$ ma kontrolnu maticu $\mathtt{H} = (-\mathtt{B}^T \mid \mathtt{E}_{n-k})$ typu $(n-k) \times n$, kde $\mathtt{B}^T$ je transponovana matica ku $\mathtt{B}$. Obratene, matica $\mathtt{H}_1 = (\mathtt{C} \mid \mathtt{E}_{n-k})$ typu $(n-k) \times n$ je kontrolnou maticou systematickeho kodu s generujucou maticou

$$\mathtt{G}_1 = (\mathtt{E}_k \mid -\mathtt{C}^T).$$

Dokaz. Nech $K_0 \subseteq \mathbf{F}^n$ je podpriestor rieseni systemu homogennych linearnych rovnic

$$\mathtt{H}\mathbf{x}^T = \mathsf{0}^T.$$

Najprv overime

$$\mathtt{HG}^T = (-\mathtt{B}^T \mid \mathtt{E}_{n-k}) \begin{pmatrix} \mathtt{E}_k \\ \mathtt{B}^T \end{pmatrix} =$$

$$= -\mathtt{B}^T\mathtt{E}_k + \mathtt{E}_{n-k}\mathtt{B}^T = -\mathtt{B}^T + \mathtt{B}^T = \mathtt{0}$$

pre blokove matice. Urobte si to detailne! Ukazali sme vlastne, ze riadky matice $\mathtt{G}$ su rieseniami horeuvedeneho systemu rovnic, teda riadky matice $\mathtt{G}$ patria do K_0. Pretoze K_0 je podpriestor, tak obsahuje aj vsetky linearne kombinacie riadkov matice $\mathtt{G}$. Teda $K \subseteq K_0$. Tvrdime, ze

$$\mathrm{dim}(K) = \mathrm{dim}(K_0).$$

Vieme, ze $\mathrm{dim}(K) = k$, lebo $\mathtt{G}$ je typu $k \times n$. Matica $\mathtt{H}$ je typu $(n-k) \times n$, co znamena, ze $h(\mathtt{H}) = n-k$, lebo stupen $\mathtt{E}_{n-k}$ je $n-k$. Z algebry vieme, ze

$$\mathrm{dim}(K_0) = n - h(\mathtt{H}) = n - (n-k) = k.$$

Z rovnosti dimenzii dostaneme rovnako ako v dokaze predchadzajucej vety uz $K = K_0$, co je koniec dokazu prvej casti vety.

Druha cast dokazu vyplyva z prvej, ak si uvedomime, ze $\mathtt{C} = -(-\mathtt{C}^T)^T$.

Cvicenie 1. Nech φ a φ_1 su linearne (n, k)-kody nad polom F, ktore su ekvivalentne, t.j. existuje permutacia $\pi \in S_n$ uskutocnujuca prechod od φ ku φ_1. Nech $\mathtt{H}$ je kontrolna matica kodu φ. Ako sa da najst kontrolna matica $\mathtt{H}_1$ ku kodu φ_1? (Odpoved: Pouzijeme permutaciu π na stlpce matice $\mathtt{H}$.)

Poznamka. Veta 18 a cvicenie 1 nam hovoria, ako vo vseobecnosti (nielen pre systematicke kody) pocitame kontrolnu maticu ku generujucej matici. V obratenom poradi to pracuje podobne: z kontrolnej matice vyrobime generujucu.

Da sa to urobit aj inak.

Veta 19. Nech $\mathtt{G}$ je generujuca matica linearneho $(n, k)-$kodu φ. Nech $\pi \in S_n$ je taka permutacia stlpcov matice $\mathtt{G}$, ze v novej matici $\mathtt{G}_1$ je prvych k stlpcov LN, t.j.

$$\mathtt{G}_1 = (\mathtt{A} \mid \mathtt{B}),$$

pricom $\mathtt{A}$ je regularna matica. Potom matica

$$\mathtt{G}_2 = \mathtt{A}^{-1}\mathtt{G}_1 = (\mathtt{E}_k \mid \mathtt{A}^{-1}\mathtt{B})$$

je generujucou maticou systematickeho kodu φ_2, ktory je ekvivalentny s φ.

Dokaz. Pretoze $\mathtt{G}$ je generujuca matica, tak $h(\mathtt{G}) = k$. Z vlastnosti hodnosti matice vieme, ze potom existuju stlpce $1 \leq \ell_1 < \cdots < \ell_k \leq n$ v $\mathtt{G}$, ktore su LN. Permutacia π prevedie tieto stlpce do prvych k stlpcov novej matice $\mathtt{G}_1$. Dostali sme maticu $\mathtt{G}_1 = (\mathtt{A} \mid \mathtt{B})$, kde prvych k stlpcov tvori regularnu podmaticu $\mathtt{A}$. Existuje preto $\mathtt{A}^{-1}$. Mozeme utvorit dalsiu maticu

$\mathtt{G}_2 = \mathtt{A}^{-1}\mathtt{G}_1$. Z algebry vieme, ze matica $\mathtt{G}_2$ sa da ziskat istymi elementarnymi riadkovymi upravami z matice $\mathtt{G}_1$. Vidime, ze $\mathtt{G}$ a $\mathtt{G}_2$ su generujuce matice dvoch ekvivalentnych linearnych kodov. Lenze

$$\mathtt{G}_2 = \mathtt{A}^{-1}\mathtt{G}_1 = (\mathtt{E}_k \mid \mathtt{A}^{-1}\mathtt{B}),$$

co znamena, ze $\mathtt{G}_2$ je systematicky kod.

Poznamka. Vo vete 19 sa tvrdi to iste co vo vete 16. Jedna sa vsak o iny dokaz.

Cvicenie 2. Nech $\mathtt{H}$ je kontrolna matica kodu φ. Nech $\mathtt{H}_1$ je matica, ktora vznikla z $\mathtt{H}$ elementarnou riadkovou upravou. Ukazte, ze $\mathtt{H}_1$ je znova kontrolnou maticou kodu φ. Zaver: Kontrolne matice nie su kodom jednoznacne urcene.

Veta 20. Nech $\mathtt{H}$ je matica typu $n - k \times n$ nad konecnym polom F a nech $\mathtt{H}$ je kontrolna matica kodu φ, pricom $h(\mathtt{H}) = n - k$. Utvorme maticu $\mathtt{H}_1$ z matice $\mathtt{H}$ takou permutaciou stlpcov $\tau \in S_n$, ze poslednych $n-k$ stlpcov matice $\mathtt{H}_1$ bude LN, t.j.

$$\mathtt{H}_1 = (\mathtt{A} \mid \mathtt{B}),$$

kde $\mathtt{B}$ je regularna podmatica. Potom matica

$$\mathtt{H}_2 = \mathtt{B}^{-1}\mathtt{H}_1 = (\mathtt{B}^{-1}\mathtt{A} \mid \mathtt{E}_{n-k})$$

je kontrolnou maticou systematickeho kodu φ_1, ktory je ekvivalentny s φ.

Dosledok. Nech platia predpoklady a oznacenia z vety 20. Utvorme generujucu maticu $\mathtt{G}_2$ ku matici $\mathtt{H}_2$ (vid vetu 18). Dalej, utvorme maticu $\mathtt{G}$ z matice $\mathtt{G}_2$ pomocou permutacie τ^{-1} stlpcov matice $\mathtt{G}_2$. Potom $\mathtt{G}$ je generujucou maticou kodu φ.

Dokaz vety. Vzhladom na predpoklad o hodnosti matice $\mathtt{H}$ vidime, ze existuje $n-k$ LN stlpcov matice $\mathtt{H}$. Postarame sa vymenou stlpcov (ak to treba!), aby tieto stlpce boli na poslednych $n-k$ miestach. Nech sa to uskutocni permutaciou τ. Dostaneme novu maticu $\mathtt{H}_1$, ktora je kontrolnou maticou kodu φ_1 a ten je ekvivalentny s φ. Navyse $\mathtt{H}_1 = (\mathtt{A} \mid \mathtt{B})$, kde $\mathtt{B}$ je podmatica vytvorena z poslednych $n-k$ stlpcov. Samozrejme je $\mathtt{B}$ regularna matica, t.j. existuje $\mathtt{B}^{-1}$. Mozeme utvorit $\mathtt{H}_2 = \mathtt{B}^{-1}\mathtt{H}_1$. Je zname, ze $\mathtt{H}_2$ vznikla z $\mathtt{H}_1$ riadkovymi elementarnymi operaciami. (Preco?) Podla cvicenia 2 je $\mathtt{H}_2$ kontrolnou maticou φ_1. Mame aj iny zapis

$$\mathtt{H}_2 = \mathtt{B}^{-1}\mathtt{H}_1 = (\mathtt{B}^{-1}\mathtt{A} \mid \mathtt{E}_{n-k}).$$

Podla vety 18 je $\mathtt{H}_2$ kontrolnou maticou systematickeho kodu, co sme oznacili ako φ_1.

Dokaz dosledku. Podla vety 18 vieme napisat generujucu maticu $\mathtt{G}_2$ ku kontrolnej matici $\mathtt{H}_2$ a ta prisluchala kodu φ_1. Vykonajme na stlpcoch matice $\mathtt{G}_2$ permutaciu τ^{-1}. Dostaneme maticu

$\mathtt{G}$, ktora je uz hladanou generujucou maticou kodu φ.

Vsimneme si este jednu kuriozitu. Najprv potrebujeme definiciu

Definicia 21. Nech K je linearny $(n,k)-$kod nad konecnym polom F. Potom K sa vola *samodualnym*, ak $K = K^{\perp}$.

Priklad 20. Zvolme binarny linearny $(2k,k)-$kod K s generujucou maticou $\mathtt{G} = (\mathtt{E}_k \mid \mathtt{P})$. Podla vety 18 je kontrolna matica $\mathtt{H}$ tvaru $(\mathtt{P}^T \mid \mathtt{E}_k)$. Vypocitajme

$$\mathtt{HG}^T = \mathtt{P}^T\mathtt{E}_k + \mathtt{E}_k\mathtt{P}^T = \mathtt{P}^T + \mathtt{P}^T = \mathtt{0}.$$

Dostali sme $K \subseteq K^{\perp}$. Dalej vieme, ze

$$\dim(K) = k = \dim(K^{\perp})$$

(vid vetu 17). Odtial mame (pouzijuc trik z vety 17), ze $K = K^{\perp}$.

Standardne dekodovanie (12. lekcia)

Najprv sa pozrieme na nove moznosti pri objavovani chyb prenosu sprav.

Definicia 22. Pod *Hammingovou vahou* slova $\mathbf{v} = v_1 \cdots v_n \in \mathbf{F}^n$ rozumieme pocet nenulovych znakov slova. Znacime to ako

$$\|\mathbf{v}\| = \|v_1 \cdots v_n\| = |\{i : v_i \neq 0\}| \, .$$

Lema 5. Nech $\varphi : A \to \mathbf{F}^n$ je linearny kod s minimalnou vzdialenostou d_φ. Potom

(i) $\rho(\mathbf{u}, \mathbf{v}) = \|\mathbf{u} - \mathbf{v}\|$ a

(ii) $d_\varphi = \min\{\|\mathbf{v}\| : \mathbf{v} \in \varphi(A) - \{0\}\}$.

Dokaz. Podmienka (i) je zrejma. V pripade (ii) oznacme

$$d' = \min\{\|\mathbf{v}\| : \mathbf{v} \in \varphi(A) - \{0\}\}.$$

Zrejme existuju $\mathbf{u}, \mathbf{v} \in \varphi(A)$ take, ze $d_\varphi = \rho(\mathbf{u}, \mathbf{v})$. Potom $d_\varphi = \|\mathbf{u} - \mathbf{v}\|$. Z toho vyplyva $d' \leq d_\varphi$. Obratene, nech $d' = \rho(\mathbf{w}, \mathbf{0})$ pre vhodne nenulove kodove slovo $\mathbf{w}$. Odtial, $d_\varphi \leq d'$, co dava $d_\varphi = d'$.

Veta 21. Linearny kod objavuje t-nasobne chyby prave vtedy, ked kazdych t stlpcov jeho kontrolnej matice je linearne nezavislych (=LN).

Dokaz. Predpokladajme, ze mame do cinenia s linearnym $(n, k)-$kodom $K = \varphi(A) \subseteq \mathbf{F}^n$. Dalej, nech $\mathtt{H}$ je kontrolna matica kodu K typu $m \times n$, pricom $m = h(\mathtt{H}) \leq n$ (pozri definiciu 18). Predpokladajme, ze K objavuje $t-$nasobne chyby. Chceme dokazat, ze kazdych t stlpcov matice $\mathtt{H}$ je LN. Podla vety 6 plati

$1 \leq t < d_\varphi \leq n$. Postupujme nepriamo. Teda predpokladame, ze existuje t LZ stlpcov $\mathtt{h}_{i_j}$ matice $\mathtt{H}$ s indexami

$$1 \leq i_1 < \cdots < i_t \leq n.$$

K tomu existuju skalary $c_{i_j} \in F$ s vlastnostou

$$c_{i_1}\mathtt{h}_{i_1} + \cdots + c_{i_t}\mathtt{h}_{i_t} = 0^T,$$

pricom aspon jedno $c_{i_j} \neq 0$. Teraz vidime, ze existuje slovo $\mathbf{c} = c_1 \cdots c_n \in \mathbf{F}^n$ s vlastnostou $c_i = 0$ pre $i \notin \{i_1, \cdots, i_t\}$. Rychle sa preveri, ze

$$\mathtt{H}\mathbf{c}^T = 0^T.$$

Teda $\mathbf{c} \in K$. Pretoze $\mathbf{c} \neq \mathbf{0}$ a

$$d_\varphi \leq \| \mathbf{c} \| \leq t,$$

tak sme dostali spor, lebo $t < d_\varphi$.

Obratene, nech kazdych $t \geq 1$ stlpcov matice $\mathtt{H}$ je LN. (Nech d_{LN} je najvacsie take t s touto vlastnostou.) Chceme dokazat, ze nas kod φ objavuje $t-$nasobne chyby. Najprv vsak tvrdime

$$d_\varphi = d_{LN} + 1.$$

Zrejme $d_\varphi \leq d_{LN}$ nie je mozne, pretoze existuje $\mathbf{u} \in K$ s vlastnostou $\| \mathbf{u} \| = d_\varphi$. Z toho vyplyva, ze $\mathtt{H}$ ma d_φ stlpcov, co su LZ. Nadmnozina tychto stlpcov je zase LZ, co by bolo v spore s tym, ze kazdych d_{LN} stlpcov $\mathtt{H}$ je LN. Teda $d_{LN} < d\varphi$, co je ekvivalentne s $d_{LN} + 1 \leq d\varphi$. Potrebujeme dokazat este obratenu nerovnost. Kvoli jednoduchsiemu zapisu polozme

$$\partial = d_{LN} + 1.$$

Z definicie cisla d_{LN} vidime, ze existuju LZ stlpce $\mathtt{h}_{j_1}, \cdots, \mathtt{h}_{j_\partial}$ matice $\mathtt{H}$ s indexami

$$1 \leq j_1 < \cdots < j_\partial \leq n.$$

Inac povedane, existuju prvky $c_{j_1}, \cdots, c_{j_\partial} \in F$, pre ktore plati

$$c_{j_1}\mathtt{h}_{j_1} + \cdots + c_{j_\partial}\mathtt{h}_{j_\partial} = 0^T,$$

pricom aspon jedno $c_{j_i} \neq 0$. Teda, existuje take slovo $\mathbf{c} = c_1 \cdots c_n \in \mathbf{F}^n$, ze $c_i = 0$ pre $i \notin \{j_1, \cdots, j_\partial\}$ a sucasne

$$\mathtt{H}\mathbf{c}^T = 0^T.$$

Dostali sme $\mathbf{c} \in K$. Pretoze $\mathbf{c} \neq \mathbf{0}$, tak lema 5 dava $d_\varphi \leq \|\mathbf{c}\|$. Na druhej strane priamo vidime, ze $\|\mathbf{c}\| \leq \partial = d_{LN}+1$, co znamena $d_\varphi \leq d_{LN}+1$. Teda $d_\varphi = d_{LN} + 1$.

Dokoncime dokaz. Pretoze $t \leq d_{LN} < d_\varphi$, tak φ podla vety 6 objavuje $t-$nasobne chyby, co je koniec dokazu.

Dosledok 1. Linearny kod objavuje jednoduche (dvojnasobne) chyby prave vtedy, ked kazdy stlpec kontrolnej matice je nenulovy (ked ziaden stlpec nie je nasobkom ineho stlpca kontrolnej matice).

Dosledok 2. Nech φ je linearny kod a nech $\mathtt{H}$ je jeho kontrolna matica. Nech d_{LN} je najvacsie prirodzene cislo t s vlastnostou: kazdych t stlpcov matice $\mathtt{H}$ je LN. Potom

$$d_\varphi = d_{LN} + 1.$$

V dalsom sa budeme venovat otazkam dekodovania linearneho kodu. Stale budeme predpokladat, ze $\varphi(A) = K \subseteq \mathbf{F}^n$ je linearny (n, k)-kod nad konecnym polom F. Oznacenie

$$\mathbf{e} + K = \{\mathbf{e} + \mathbf{v} : \mathbf{v} \in K\}$$

pozname z teorie grup.

Definicia 23. Ked sa vyslalo kodove slovo $\mathbf{v} = v_1 \cdots v_n$ a prijali sme $\mathbf{w} = w_1 \cdots w_n \in \mathbf{F}^n$, tak slovo

$$\mathbf{e} = e_1 \cdots e_n = \mathbf{w} - \mathbf{v}$$

volame *chybovym* slovom (vzniklo sumom). Ekvivaletne,

$$\mathbf{w} = \mathbf{v} + \mathbf{e},$$

t.j. prijate slovo je suctom vyslaneho (kodoveho) slova a chyboveho slova.

Veta 22. Nech K je linearny (n, k)-kod nad konecnym polom F. Potom mnoziny

$$\{\mathbf{x} + K : \mathbf{x} \in \mathbf{F}^n\}$$

tvoria rozklad $\mathbf{F}^n$. Dalej mame

(i) $\mathbf{e} + K = \mathbf{e}' + K \;\Leftrightarrow\; \mathbf{e} - \mathbf{e}' \in K$;

(ii) $\mathbf{e} - \mathbf{e}' \notin K \;\Leftrightarrow\; (\mathbf{e} + K) \cap (\mathbf{e}' + K) = \emptyset$;

(iii) $|\, \mathbf{e} + K \,| = |\, K \,|$;

(iv) $\mid K \mid = q^k$, ak $\mid F \mid = q = p^r$ a

$$[\mathbf{F}^n : K] = q^{n-k}.$$

Dokaz vyplyva priamo zo znamej Lagrangeovej vety pre grupy.

Definicia 24. Nech $\varphi(A) = K \subseteq \mathbf{F}^n$ je linearny $(n, k)-$kod nad konecnym polom F. Vyberme s kazdej triedy rozkladu $\mathbf{u}+K$ slovo $\mathbf{u}'$ s najmensou vahou. Budeme ho volat *reprezentantom* triedy $\mathbf{u}+K$. (Oznacenie: $\mathbf{u}' = \mathrm{repr}(\mathbf{u}+K)$.)

Veta 23. (Standardne dekodovanie). Nech $\varphi : A \to \mathbf{F}^n$ je linearny (n, k)-kod nad konecnym polom F. Potom zobrazenie

$$\delta : \mathbf{F}^n \to K$$

definovane predpisom

$$\delta(\mathbf{w}) = \mathbf{w} - \mathrm{repr}(\mathbf{w}+K)$$

je dekodovanie kodu φ (volame ho *standard-nym*).

Dokaz. Oznacme $\mathbf{w}' = \mathrm{repr}(\mathbf{w} + K)$. Potom z vety 22(i) vidime, ze $\mathbf{w} - \mathbf{w}' \in K$, t.j. $\delta(\mathbf{w}) \in K$. Dalej, $\delta(\mathbf{w}) = \mathbf{w}$ pre $\mathbf{w} \in K$, pretoze $\mathbf{0}$ je reprezentantom tejto triedy.

Priklad 21. Zoberme (4,3)-kod celkovej kontroly parity (vid priklad 15), pricom $A = (Z_2)^3$. Potom kodovymi slovami su

$$K =$$

$$\{0000, 1001, 0101, 0011, 1111, 1100, 0110, 1010\}.$$

Zvolime nekodove slovo, napr. 1000. Potom

$$1000 + K =$$

$$\{1000, 0001, 1101, 1011, 0111, 0100, 1110, 0010\}.$$

Mame len dve triedy: K a $1000 + K$. V druhej triede sme si mohli vybrat aj ineho reprezentanta: 0001. Celu situaciu si mozeme zapisat trochu inac do tzv. Slepnianovej tabulky

0000 1001 0101 0011 1111 1100 0110 1010

0001 1000 0100 0010 1110 1101 0111 1011

V prvom riadku su uvedene kodove slova a v druhom su slova tvaru 0001+**v** z triedy 0001+K, pricom 0001+**v** lezi pod **v**. Pri tomto zapise je $\delta(\mathbf{w})$ slovo z prveho riadku v tom istom stlpci, kde sa nachadza **w**. Algoritmus prehladava 2^4 slov.

Syndromy. (13. lekcia)

Zacneme s este jednym poucnym prikladom.

Priklad 22. Nech K je binarny kod dlzky 6. Potom $\mathbf{u} = u_1 \cdots u_6 \in K$ prave vtedy, ked

$u_4 = u_1 + u_2$, $u_5 = u_1 + u_3$ a $u_6 = u_2 + u_3$. Inac povedane, kodove slovo **u** ma 3 informacne znaky, t.j. u_1, u_2 a u_3. Zvysne 3 znaky su kontrolne. Rychle sa presvedcime, ze K je linearny kod nad Z_2 definovany nasledovnym homogennym systemom rovnic

$$x_1 + x_2 + x_4 = 0$$

$$x_1 + x_3 + x_5 = 0$$

$$x_2 + x_3 + x_6 = 0.$$

Kontrolna matica $\mathtt{H}$ sa teraz da jednoducho urcit

$$\mathtt{H} = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

Pretoze $h(\mathtt{H}) = 3$, tak mame $2^3 = 8$ kodovych slov: 000000, 001011, 010101, 011110, 100110, 101101, 110011, 111000. Podla vety 22 mame $2^3 = 8$ tried rozkladu aditivnej grupy $(Z_2)^3$. Ak

K znamena podgrupu kodovych slov, tak jednotlive triedy su: K, $000001+K$, $000010+K$, $000100+K$, $001000+K$, $010000+K$, $100000+K$ a $001100+K$. Mozno vytvorit Slepnianove tabulky a dostaneme jednoduchy algoritmus na dekodovanie.

Definicia 25. Nech φ je linearny $(n,k)-$kod nad konecnym polom F s kontrolnou maticou $\mathtt{H}$. Nech dalej $\mathbf{v} \in \mathbf{F}^n$. Potom slovo $\mathtt{s} \in \mathbf{F}^{n-k}$ sa nazyva *syndromom* slova $\mathbf{v}$, ak plati

$$\mathtt{s}^T = \mathtt{H}\mathbf{v}^T.$$

Veta 24. Nech φ je linearny $(n,k)-$kod s kontrolnou maticou $\mathtt{H}$ a podpriestorom kodovych slov K.

(i) Nech dalej $\mathbf{u}, \mathbf{v} \in \mathbf{F}^n$, pricom $\mathtt{s}_1, \mathtt{s}_2 \in \mathbf{F}^{n-k}$ su k nim odpovedajuce syndromy. Potom $\mathtt{s}_1 = \mathtt{s}_2$ prave vtedy, ked

$$\mathbf{u} + K = \mathbf{v} + K.$$

(ii) Nech $\mathbf{w} \in \mathbf{F}^n$ je prijate slovo vyslaneho kodoveho slova $\mathbf{u}$. Zrejme $\mathbf{w} = \mathbf{u} + \mathbf{e}$. Potom $\mathbf{w}$ a $\mathbf{e}$ maju rovnake syndromy.

Dokaz. (i) Zrejme, $\mathtt{H}\mathbf{u}^T = \mathtt{0}^T$ prave vtedy, ked $\mathbf{u} \in K$. Druha vec, ktoru potrebujeme, je

$$\mathtt{H}(\mathbf{u} + \mathbf{v})^T = \mathtt{H}\mathbf{u}^T + \mathtt{H}\mathbf{v}^T.$$

(ii) Pretoze $\mathbf{w} \in \mathbf{e} + K$, tak z (i) dostavame tvrdenie.

Dosledok. Prijate slovo ma ten isty syndrom ako chybove slovo.

Definicia 26. Hovorime, ze linearny kod φ pri dekodovani δ *opravuje chybove slovo* $\mathbf{e}$, ak plati

$$\delta(\mathbf{e} + \mathbf{v}) = \mathbf{v} \quad \text{pre kazde kodove slovo } \mathbf{v}.$$

Veta 25. Nech φ je linearny kod s minimalnou vzdialenostou $d_\varphi = d$. Potom standardne dekodovanie opravi $t-$nasobne chyby pre vsetky $t < d/2$.

Dokaz. Predpokladame, ze sme vyslali kodove slovo $\mathbf{u}$ a prijali sme slovo $\mathbf{w}$. Dalej predpokladame, ze plati $\rho(\mathbf{u}, \mathbf{w}) = t < d/2$. Potrebujeme dokazat, ze $\rho(\mathbf{u}, \mathbf{w}) < \rho(\mathbf{x}, \mathbf{w})$ pre vsetky kodove slova $\mathbf{x} \neq \mathbf{u}$. Pretoze φ je linearny kod, tak vieme, ze $\mathbf{w} = \mathbf{u} + \mathbf{e}$, kde $\mathbf{e}$ je chybove slovo. Potom

$$\rho(\mathbf{u}, \mathbf{w}) = \|\mathbf{w} - \mathbf{u}\| = \|\mathbf{e}\| = t.$$

Zoberme kodove slovo $\mathbf{x} \neq \mathbf{u}$. Teraz

$$\rho(\mathbf{x}, \mathbf{w}) = \|\mathbf{w} - \mathbf{x}\| = \|(\mathbf{u} - \mathbf{x}) + \mathbf{e}\|.$$

Lenze $\|\mathbf{u} - \mathbf{x}\| \geq d$, lebo su to rozne kodove slova. Z toho teraz vyplyva, ze

$$\rho(\mathbf{x}, \mathbf{w}) = \|(\mathbf{u} - \mathbf{x}) + \mathbf{e}\| \geq d/2$$

a dokaz je hotovy.

Veta 26. Standardne dekodovanie opravuje prave tie chybove slova, ktore sme zvolili za reprezentantov tried. Navyse, standardne dekodovanie δ je optimalne v tom zmysle, ze ziadne ine dekodovanie neopravuje vacsiu mnozinu chybovych slov ako δ.

Dokaz. Nech $\mathbf{e}$ je reprezentantom svojej triedy $\mathbf{e}+K$. Nech $\mathbf{v}$ je kodove slovo a nech δ je standardne dekodovanie. Potom

$$\delta(\mathbf{e}+\mathbf{v}) = \mathbf{e}+\mathbf{v}-\mathbf{e} = \mathbf{v}.$$

Dalej, nech $\mathbf{e}'$ nie je reprezentantom triedy $\mathbf{e}'+K$. Je nim $\mathbf{e}$. Potom vsak

$$\delta(\mathbf{e}'-\mathbf{e}) = \mathbf{e}'-\mathbf{e} \neq \mathbf{0},$$

lebo $\mathbf{e}'-\mathbf{e}$ je kodove slovo (veta 22). Predpokladajme, ze by δ opravilo aj chybove slovo $\mathbf{e}'$. Z toho dostaneme

$$\delta(\mathbf{e}') = \mathbf{e}'-\mathbf{e} \neq \mathbf{0}.$$

Pretoze $\mathbf{0}$ je kodove slovo, tak

$$\delta(\mathbf{e}' + \mathbf{0}) = \mathbf{0} = \delta(\mathbf{e}') = \mathbf{e}' - \mathbf{e} \neq \mathbf{0},$$

co je spor. Dokazali sme prvu cast vety.

Predpokladajme, ze δ^* je dalsie dekodovanie a nech δ^* opravuje vsetky tie chybove slova, co robi δ, t.j. δ^* je vsade tam definovane, kde aj δ, pricom na spolocnom definicnom obore nadobudaju rovnake hodnoty. Nech
$\mathbf{e}' \in \mathbf{e} + K$ a predpokladajme, ze $\mathbf{e}$ je reprezentantom svojej triedy, pricom $\mathbf{e} \neq \mathbf{e}'$. Zrejme $\mathbf{e}' - \mathbf{e} \in K$, lebo su z jednej triedy. Dostaneme

$$\delta^*(\mathbf{e}') = \delta(\mathbf{e}') = \mathbf{e}' - \mathbf{e} \neq \mathbf{0}.$$

Dalej postupujme nepriamo. Predpokladajme, ze by δ^* bolo lepsie ako δ, t.j. ze δ^* opravi aj chybove slovo $\mathbf{e}'$. Z toho vyplyva

$$\delta^*(\mathbf{e}') = \delta^*(\mathbf{e}' + \mathbf{0}) = \mathbf{0},$$

co je spor, lebo vyssie nam vyslo $\delta^*(\mathbf{e}') \neq \mathbf{0}$.

Poznamka. Urobime strucny zaver o standardnom dekodovani pomocou syndromov. Pred dekodovanim si pripravime zoznam reprezentatov $\mathbf{e}_0, \cdots, \mathbf{e}_{\partial-1}$, kde $\partial = q^{n-k}$ je pocet tried rozkladu $\mathbf{F}^n$ podla K (veta 22). K tymto slovam si urcime syndromy $\mathtt{s}_0, \cdots, \mathtt{s}_{\partial-1}$. Potom, ak prijmeme slovo $\mathbf{w} \in \mathbf{F}^n$, tak vypocitame jeho syndrom, povedzme $\mathtt{s}_\triangle$. Vyhladame v nasom (pripravenom) zozname syndromov syndrom $\mathtt{s}_j$, pre ktory plati

$$\mathtt{s}_j = \mathtt{s}_\triangle.$$

(Ten je jednoznacne urceny - vid vetu 24.) Vyberieme reprezentanta $\mathbf{e}_j = \text{repr}(\mathbf{w} + K)$ a mame vysledne slovo

$$\delta(\mathbf{w}) = \mathbf{w} - \mathbf{e}_j.$$

Pomocou syndromov sa standardne dekodovanie vykona rychlejsie ako pomocou Slepnianovej tabulky. V prvom pripade sme prehladavali zoznam s q^{n-k} prvkami, v druhom pripade je to uz q^n prvkov. Vratme sa kratko este k prikladu 22. Pouzijeme metodu syndromov. Tam sme uz nasli reprezentantov tried: $\mathbf{e}_0 = 000000$, $\mathbf{e}_1 = 100000$, $\mathbf{e}_2 = 010000$, $\mathbf{e}_3 = 001000$, $\mathbf{e}_4 = 000100$, $\mathbf{e}_5 = 000010$, $\mathbf{e}_6 = 000001$, $\mathbf{e}_7 = 001100$. Odpovedajuce syndromy rychle vypocitame: $\mathtt{s}_0 = 000$, $\mathtt{s}_1 = 110 = \mathtt{h}_1^T$, $\mathtt{s}_2 = 101 = \mathtt{h}_2^T$, $\mathtt{s}_3 = 011 = \mathtt{h}_3^T$, $\mathtt{s}_4 = 100 = \mathtt{h}_4^T$, $\mathtt{s}_5 = 010 = \mathtt{h}_5^T$, $\mathtt{s}_6 = 001 = \mathtt{h}_6^T$, $\mathtt{s}_7 = 111 = \mathtt{h}_3^T + \mathtt{h}_4^T$, kde $\mathtt{h}_i$ znamena $i-$ty stlpec kontrolnej matice $\mathtt{H}$. Teraz, ak prijmeme slovo $\mathbf{w} = 100100$, tak $\mathtt{H}\mathbf{w}^T = \mathtt{s}_\triangle^T$, co dava

$$\mathtt{s}_\triangle = 010 = \mathtt{h}_1^T + \mathtt{h}_4^T = \mathtt{s}_5.$$

Teda,

$$\delta(\mathbf{w}) = \mathbf{w} + \mathbf{e}_5 = 100100 + 000010 = 100110.$$