Teorie kódování - Kapitola 4
Kódy opravující chyby

Jan Paseka

Ústav matematiky a statistiky
Masarykova univerzita

12. dubna 2023

Obsah

Základní pojmy

FI Připomeňme si následující předpoklady pro kódování. Zdroj vytváří ***zprávu***, která sestává z posloupnosti zdrojových symbolů a tato zpráva je přenesena k příjemci přes kanál s možnou chybou. Přitom lze bez újmy na obecnosti předpokládat, že kanál má stejnou abecedu Σ jak na vstupu tak na výstupu. Kód $\mathcal{C}$ nad abecedou Σ je soubor posloupností symbolů ze Σ, prvky z $\mathcal{C}$ se nazývají ***kódová slova***.

Základní pojmy

FI Připomeňme si následující předpoklady pro kódování. Zdroj vytváří ***zprávu***, která sestává z posloupnosti zdrojových symbolů a tato zpráva je přenesena k příjemci přes kanál s možnou chybou. Přitom lze bez újmy na obecnosti předpokládat, že kanál má stejnou abecedu Σ jak na vstupu tak na výstupu. Kód $\mathcal{C}$ nad abecedou Σ je soubor posloupností symbolů ze Σ, prvky z $\mathcal{C}$ se nazývají ***kódová slova***.

Budeme předpokládat, že všechna kódová slova mají stejnou délku. Takovéto kódy se nazývají ***blokové kódy*** a při jejich použití je dekódování podstatně snazší. Pokud mají kódová slova z $\mathcal{C}$ délku n a $|\Sigma| = q$, pak mluvíme o ***q-árním kódu délky*** n (***binárním kódu***, pokud $q = 2$).

Základní pojmy

Zakódování zdrojové zprávy není nic jiného než přiřazení každé k-dlouhé sekvenci znaků nad zdrojovou abecedou Σ jedno kódové slovo z $\mathcal{C}$.

Základní pojmy

Zakódování zdrojové zprávy není nic jiného než přiřazení každé k-dlouhé sekvenci znaků nad zdrojovou abecedou Σ jedno kódové slovo z $\mathcal{C}$.

Během samotného dekódování se přijatá sekvence rozčlení na bloky délky n a každý se zpracovává samostatně. Jelikož přijaté n-bloky mohou mít díky chybám při přenosu obecně jinou podobu než vysílaná kódová slova, musí dekodér rozhodovat, které slovo bylo vysláno.

Základní pojmy

Zakódování zdrojové zprávy není nic jiného než přiřazení každé k-dlouhé sekvenci znaků nad zdrojovou abecedou Σ jedno kódové slovo z $\mathcal{C}$.

Během samotného dekódování se přijatá sekvence rozčlení na bloky délky n a každý se zpracovává samostatně. Jelikož přijaté n-bloky mohou mít díky chybám při přenosu obecně jinou podobu než vysílaná kódová slova, musí dekodér rozhodovat, které slovo bylo vysláno.

Pokud je kód dobře navržen, je pravěpodobnost špatného rozhodnutí mnohem menší než pravděpodobnost, že libovolný kódový znak je chybně přenesen.

Základní pojmy

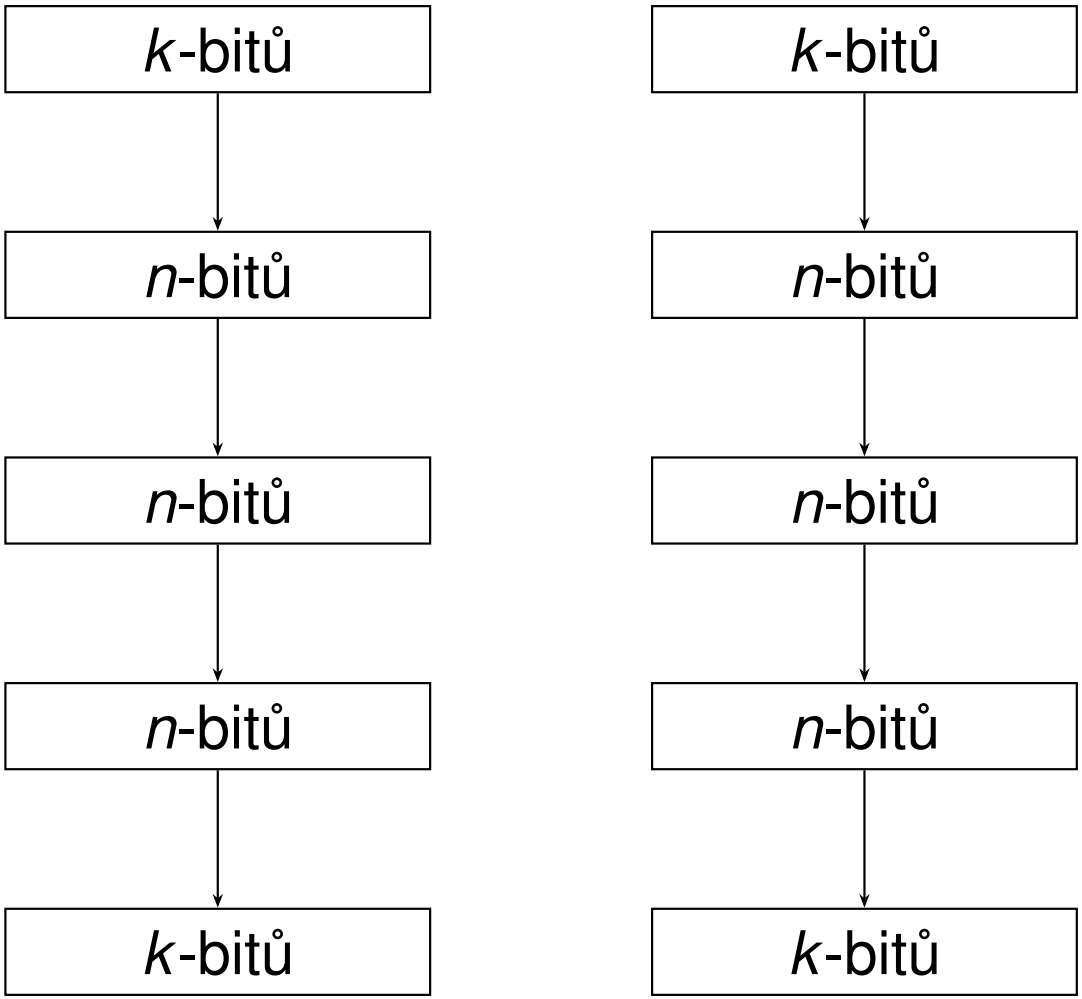
Proces rozhodování může být definován pomocí dekódovací tabulky. Kódová slova tvoří první řádek tabulky. Pokud jsme obdrželi kódové slovo, je logické předpokládat, že i stejné slovo bylo vysláno.

Základní pojmy

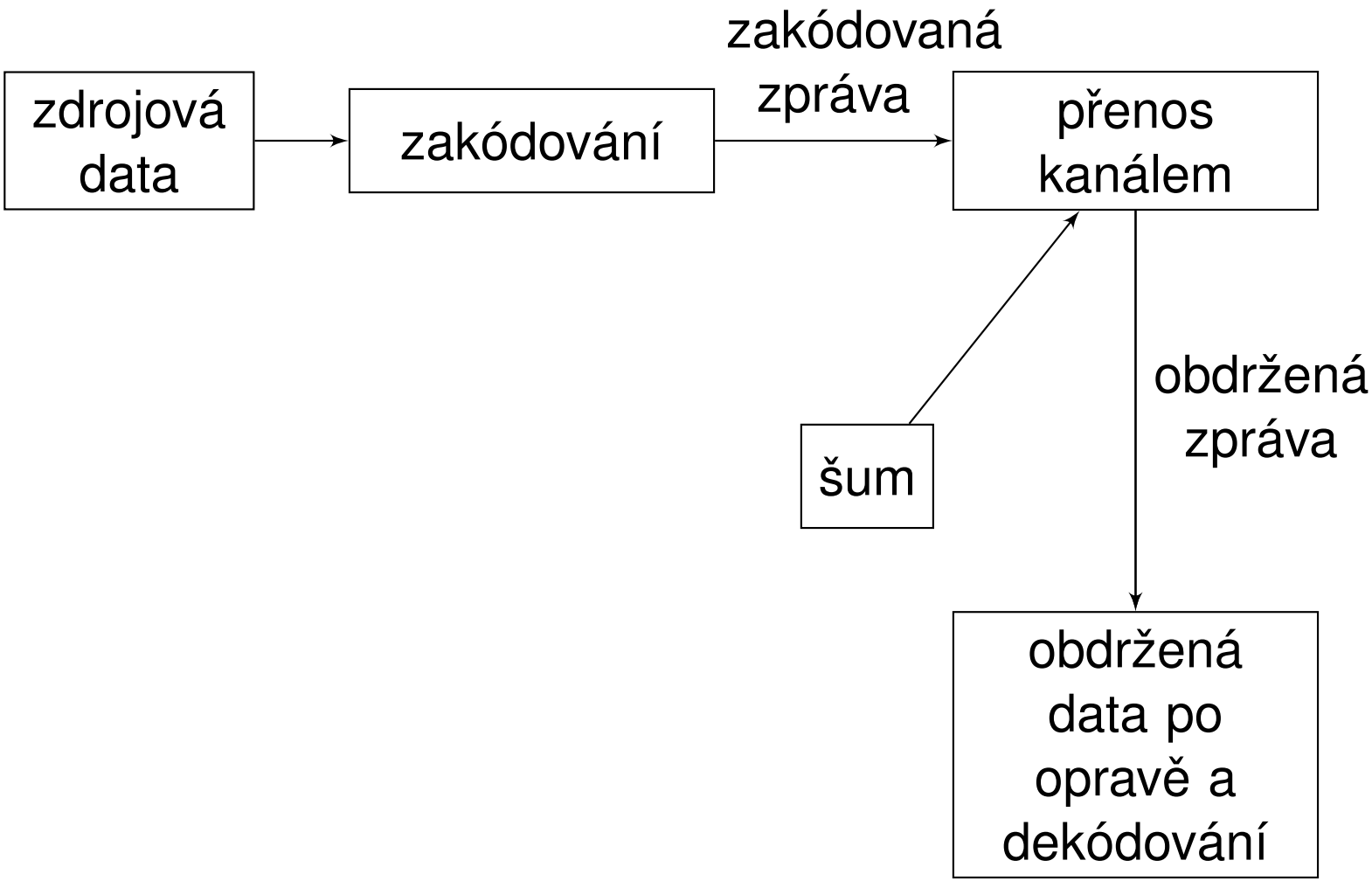
Proces rozhodování může být definován pomocí dekódovací tabulky. Kódová slova tvoří první řádek tabulky. Pokud jsme obdrželi kódové slovo, je logické předpokládat, že i stejné slovo bylo vysláno.

Rozhodovací pravidlo pro zbylá možně přijatá slova je dáno rozdělením těchto slov do seznamů pod každým kódovým slovem, podle kterého se tato přijatá slova budou dekódovat. Tedy, každé slovo délky n nad abecedou Σ se objeví v tabulce právě jednou.

Základní pojmy - Blokové kódování



Základní pojmy



Obrázek 1: Komunikační kanál

Základní pojmy

Definition 1.1

Buď u, v přirozená čísla. Řekneme, že ***kód*** $\mathcal{C}$ ***určí*** u ***chyb***, jestliže, pokud každé kódové slovo změníme alespoň na jednom a ne více než u místech, nebude výsledný řetězec kódové slovo. Řekneme, že ***kód*** $\mathcal{C}$ ***určí právě*** u ***chyb***, jestliže určí u chyb a neurčí $u + 1$ chyb.

Základní pojmy

Definition 1.1

Buď u, v přirozená čísla. Řekneme, že ***kód*** $\mathcal{C}$ ***určí*** u ***chyb***, jestliže, pokud každé kódové slovo změníme alespoň na jednom a ne více než u místech, nebude výsledný řetězec kódové slovo. Řekneme, že ***kód*** $\mathcal{C}$ ***určí právě*** u ***chyb***, jestliže určí u chyb a neurčí $u + 1$ chyb.

Řekneme, že ***kód*** $\mathcal{C}$ ***opraví*** v ***chyb***, jestliže, pokud použijeme pravidlo minimální vzdálenosti, jsme schopni opravit alespoň v chyb a v případě, kdy se nebudeme moci rozhodnout, dostaneme na výstupu chybu v dekódování. Řekneme, že ***kód*** $\mathcal{C}$ ***opraví právě*** v ***chyb***, jestliže opraví v chyb a neopraví $v + 1$ chyb.

Základní pojmy

Dále budeme předpokládat, že abychom byli schopni zjistit chyby při přenosu, bude příjemce schopen zkontrolovat přijatý řetězec proti seznamu všech kódových slov. Pokud řetězec nebude na seznamu, příjemce ví, že nastala alespoň jedna chyba, ale není schopen zjistit, kolik chyb skutečně nastalo.

Základní pojmy

Dále budeme předpokládat, že abychom byli schopni zjistit chyby při přenosu, bude příjemce schopen zkontrolovat přijatý řetězec proti seznamu všech kódových slov. Pokud řetězec nebude na seznamu, příjemce ví, že nastala alespoň jedna chyba, ale není schopen zjistit, kolik chyb skutečně nastalo.

Zároveň by mělo být jasné, že pokud obdržené slovo nebude kódové slovo, bude podle pravidla minimální vzdálenosti zpátky dekódováno, ale příjemce neví, zda se skutečně jedná o odeslané slovo. Příjemce pouze ví, že pokud nenastane, v případě kódu opravujícího v chyb, více než v chyb, pak dekódovací proces bude úspěšný tj. pomocí pravidla minimální vzdálenosti získáme odeslané kódové slovo.

Příklad

Příklad 1.2

Chceme vysílat čtyři znaky: a, b, c, d a zpráva bude přenášena pomocí binárního blokového kódu délky 5. Musíme tedy zvolit čtyři kódová slova, např. 11000 pro a, 00110 pro b, 10011 pro c a 01101 pro d. Dekódování musí zahrnout všech $2^5 = 32$ binárních slov délky 5. Jedno takové dekódovací pravidlo je na (obr.2).

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

Obrázek 2: *Příklad kódové tabulky pro binární blokový kód délky 5.*

Příklad

Konstrukce kódu a dekódovacího schématu z příkladu 1.2 opravuje ne více než jednu chybu. V tabulce je to vždy prvních 5 slov v seznamu pod kódovým slovem.

Příklad

Konstrukce kódu a dekódovacího schématu z příkladu 1.2 opravuje ne více než jednu chybu. V tabulce je to vždy prvních 5 slov v seznamu pod kódovým slovem.

U více chyb už nemáme jistotu, že dekódování proběhne správně. Například pokud by při přenosu bloku 11000 vznikly dvě chyby vedoucí k přijetí slova 11110 na výstupu kanálu, pak toto schéma chyby odstraní. Ovšem při obdržení 11011 bude toto slovo dekódováno chybně jako 10011.

Hammingova vzdálenost a váha

Označme dále $V_n(\Sigma)$ množinu všech posloupností délky n nad abecedou Σ a nazývejme prvky ze $V_n(\Sigma)$ ***slova*** nebo ***vektory***. Někdy budeme psát místo $V_n(\Sigma)$ také $V_n(q)$, pokud $q = \mathrm{card}\Sigma$.

Hammingova vzdálenost a váha

Označme dále $V_n(\Sigma)$ množinu všech posloupností délky n nad abecedou Σ a nazývejme prvky ze $V_n(\Sigma)$ ***slova*** nebo ***vektory***. Někdy budeme psát místo $V_n(\Sigma)$ také $V_n(q)$, pokud $q = \text{card}\Sigma$.

Podobně jako v binárním případě je ***Hammingova vzdálenost*** $d(\mathbf{x}, \mathbf{y})$ mezi vektory $\mathbf{x}$ a $\mathbf{y}$ počet míst, ve kterých se $\mathbf{x}$ a $\mathbf{y}$ liší.

Hammingova vzdálenost a váha

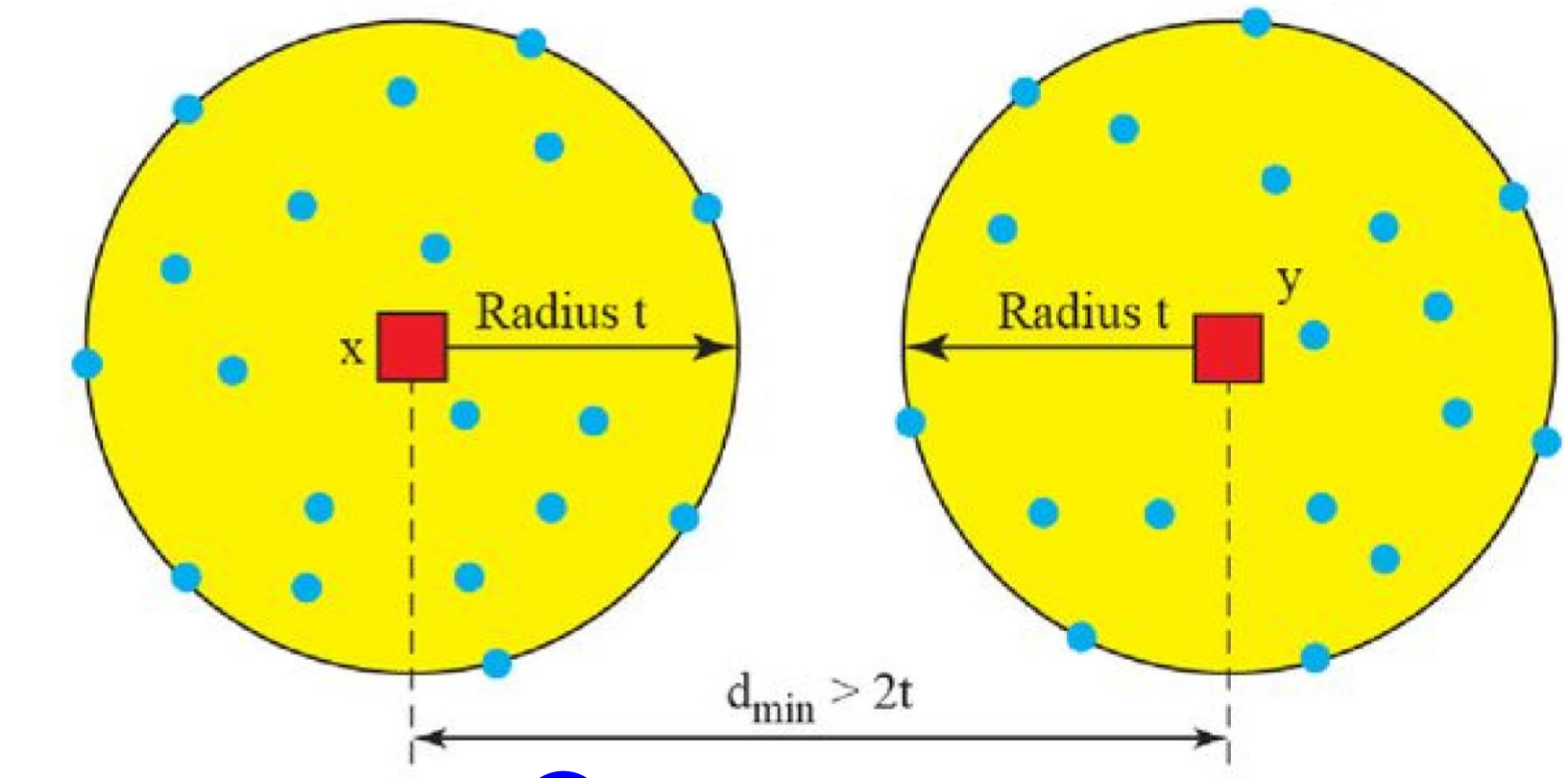
Označme dále $V_n(\Sigma)$ množinu všech posloupností délky n nad abecedou Σ a nazývejme prvky ze $V_n(\Sigma)$ ***slova*** nebo ***vektory***. Někdy budeme psát místo $V_n(\Sigma)$ také $V_n(q)$, pokud $q = \mathrm{card}\Sigma$.

Podobně jako v binárním případě je ***Hammingova vzdálenost*** $\mathrm{d}(\mathbf{x}, \mathbf{y})$ mezi vektory $\mathbf{x}$ a $\mathbf{y}$ počet míst, ve kterých se $\mathbf{x}$ a $\mathbf{y}$ liší.

V případě, že $q = \mathrm{card}\Sigma$ je mocnina prvočísla, můžeme považovat abecedu Σ za těleso $\mathbb{F}_q$, které má nulový prvek 0. Množinově můžeme ztotožnit $\mathbb{F}_q$ s $\mathbf{Z}_q$, přirozené operace na $\mathbf{Z}_q$ ale nemusí odpovídat operacím na $\mathbb{F}_q$.

Váha slova $\mathbf{x} = x_1 x_2 \cdots x_n$ je pak počet nenulových znaků slova $\mathbf{x}$, tj. $\mathrm{wt}(\mathbf{x}) = \mathrm{d}(\mathbf{x}, \mathbf{0})$, kde $\mathbf{0}$ je slovo z n nul.

Hammingova vzdálenost a váha



Obrázek 3: Použití Hammingovy vzdálenosti pro opravu až *t* chyb.

Hammingova vzdálenost a váha

Věta 1.3

Nechť $V_n(\Sigma)$ je množina všech posloupností délky n nad abecedou Σ . Potom funkce Hammingovy vzdálenosti $d(-,-)\colon V_n(\Sigma) \times V_n(\Sigma) \to \mathbb{N}$ má následující vlastnosti. Pro všechna $\mathbf{x}, \mathbf{y}, \mathbf{z} \in V_n(\Sigma)$,

(pozitivní definitivita)

$$d(\mathbf{x}, \mathbf{y}) \geq 0, \text{ a } d(\mathbf{x}, \mathbf{y}) = 0 \text{ právě tehdy, když } \mathbf{x} = \mathbf{x},$$

(symetrie)

$$d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$$

(trojúhelníková nerovnost)

$$d(\mathbf{x}, \mathbf{y}) \leq d(\mathbf{x}, \mathbf{z}) + d(\mathbf{z}, \mathbf{y}).$$

Tedy, dvojice $(V_n(\Sigma), d(-,-))$ je metrický prostor.

Hammingova vzdálenost a váha

Definice 1

Buď $\mathbf{x} \in \mathbf{Z}_r^n$, $\rho \geq 0$. **Sférou** $\mathbf{S}_\rho^{n,r}(\mathbf{x})$ **v** $\mathbf{Z}_r^n$ **se středem x a poloměrem** ρ *rozumíme množinu*

$$\mathbf{S}_\rho^{n,r}(\mathbf{x}) = \{\mathbf{y} \in \mathbf{Z}_r^n : d(\mathbf{x}, \mathbf{y}) \leq \rho\}.$$

Hammingova vzdálenost a váha

Definice 1

Buď $\mathbf{x} \in \mathbf{Z}_r^n$, $\rho \geq 0$. **Sférou** $\mathbf{S}_\rho^{n,r}(\mathbf{x})$ **v** $\mathbf{Z}_r^n$ **se středem x a poloměrem** ρ *rozumíme množinu*

$$\mathbf{S}_\rho^{n,r}(\mathbf{x}) = \{\mathbf{y} \in \mathbf{Z}_r^n : d(\mathbf{x}, \mathbf{y}) \leq \rho\}.$$

Objemem sféry $\mathbf{S}_\rho^{n,r}(\mathbf{x})$*nazveme číslo*

$$\mathbf{V}_\rho^{n,r}(\mathbf{x}) = \mathrm{card}(\{\mathbf{y} \in \mathbf{Z}_r^n : d(\mathbf{x}, \mathbf{y}) \leq \rho\}),$$

tj. počet řetězců délky n, které mají Hammingovu vzdálenost od **x** *nejvýše* ρ.

Hammingova vzdálenost a váha

Definice 1

Buď $\mathbf{x} \in \mathbf{Z}_r^n$, $\rho \geq 0$. **Sférou** $\mathbf{S}_\rho^{n,r}(\mathbf{x})$ **v** $\mathbf{Z}_r^n$ **se středem x a poloměrem** ρ *rozumíme množinu*

$$\mathbf{S}_\rho^{n,r}(\mathbf{x}) = \{\mathbf{y} \in \mathbf{Z}_r^n : d(\mathbf{x}, \mathbf{y}) \leq \rho\}.$$

Objemem sféry $\mathbf{S}_\rho^{n,r}(\mathbf{x})$*nazveme číslo*

$$\mathbf{V}_\rho^{n,r}(\mathbf{x}) = \mathrm{card}(\{\mathbf{y} \in \mathbf{Z}_r^n : d(\mathbf{x}, \mathbf{y}) \leq \rho\}),$$

tj. počet řetězců délky n, které mají Hammingovu vzdálenost od $\mathbf{x}$ *nejvýše* ρ.

Pokud budou čísla n, r jasná ze souvislosti, budeme psát jednoduše $\mathbf{S}_\rho(\mathbf{x})$ *a* $\mathbf{V}_\rho(\mathbf{x})$.

Hammingova vzdálenost a váha

Platí pak

Lemma 1.4

Objem sféry $\mathbf{S}_\rho^{n,r}(\mathbf{x})$ *je určen vztahem*

$$\mathbf{V}_\rho^{n,r}(\mathbf{x}) = \sum_{k=0}^{\rho} \binom{n}{k} (r-1)^k.$$

Hammingova vzdálenost a váha

Platí pak

Lemma 1.4

Objem sféry $\mathbf{S}_\rho^{n,r}(\mathbf{x})$ *je určen vztahem*

$$\mathbf{V}_\rho^{n,r}(\mathbf{x}) = \sum_{k=0}^{\rho} \binom{n}{k} (r-1)^k.$$

Důkaz.

Plyne z toho, že počet řetězců délky n, které mají Hammingovu vzdálenost od $\mathbf{x}$ právě k, je přesně číslo

$$\binom{n}{k} (r-1)^k.$$

Hammingova vzdálenost a váha

Příklad 1.5

Hammingova vzdálenost slov 01110010 *a* 11110101 *je* 4 *a váha slova* 01110101 *je* 5.

Hammingova vzdálenost a váha

Příklad 1.5

Hammingova vzdálenost slov 01110010 *a* 11110101 *je* 4 *a váha slova* 01110101 *je* 5.

Dekódování podle principu ***minimální vzdálenosti*** znamená, že dekódujeme obdržený vektor **y** jako to kódové slovo **c**, které má minimální vzdálenost od **y**, pokud máme možný výběr, vybereme libovolně.

Hammingova vzdálenost a váha

Příklad 1.5

Hammingova vzdálenost slov 01110010 *a* 11110101 *je* 4 *a váha slova* 01110101 *je* 5.

Dekódování podle principu ***minimální vzdálenosti*** znamená, že dekódujeme obdržený vektor **y** jako to kódové slovo **c**, které má minimální vzdálenost od **y**, pokud máme možný výběr, vybereme libovolně.

Je-li tedy $\mathcal{C}$ kód, je ***minimální vzdálenost kódu*** $\mathcal{C}$ číslo

$$\mathrm{d}(\mathcal{C}) = \min \mathrm{d}(\mathbf{c}_i, \mathbf{c}_j),$$

kde je minimum bráno přes všechny navzájem různé dvojice kódových slov z $\mathcal{C}$. Pojem minimální vzdálenosti je **klíčový pojem** pro hodnocení kódu; dobré kódy mají rozložená kódová slova tak, že jejich minimální vzdálenost je velká.

Oprava chyb podle pravidla minimální vzdálenosti

Důvod důležitosti minimální vzdálenosti je jasný z následující věty.

Věta 1.6

Má-li kód minimální vzdálenost d, lze opravit pomocí dekódování podle pravidla minimální vzdálenosti až $\frac{1}{2}(d-1)$ *chyb.*

Oprava chyb podle pravidla minimální vzdálenosti

Důvod důležitosti minimální vzdálenosti je jasný z následující věty.

Věta 1.6

Má-li kód minimální vzdálenost d, lze opravit pomocí dekódování podle pravidla minimální vzdálenosti až $\frac{1}{2}(d-1)$ *chyb.*

Důkaz.

Položme $v = \lfloor \frac{1}{2}(d-1) \rfloor$. Jsou-li $\mathbf{x}, \mathbf{z}$ různá kódová slova, platí

$$\mathbf{S}_v(\mathbf{x}) \cap \mathbf{S}_v(\mathbf{z}) = \emptyset.$$

Totiž, pokud $\mathbf{y} \in \mathbf{S}_v(\mathbf{x}) \cap \mathbf{S}_v(\mathbf{z})$, pak

$$\mathrm{d}(\mathbf{x}, \mathbf{z}) \leq \mathrm{d}(\mathbf{x}, \mathbf{y}) \leq \frac{1}{2}(d-1) + \frac{1}{2}(d-1) = d - 1 < d,$$

spor. Tedy dekódování podle pravidla minimální vzdálenosti opraví až v chyb. ∎

Detekce chyb podle pravidla minimální vzdálenosti

Věta 1.7

Buďte u, v přirozená čísla. Pak kód $\mathcal{C}$ ***určí*** *u (****opraví*** *v) **chyb** právě tehdy, když $d(\mathcal{C}) \geq u + 1$ ($d(\mathcal{C}) \geq 2v + 1$).*

Detekce chyb podle pravidla minimální vzdálenosti

Věta 1.7

Buďte u, v přirozená čísla. Pak kód $\mathcal{C}$ ***určí*** *u (****opraví*** *v) **chyb** právě tehdy, když $d(\mathcal{C}) \geq u + 1$ ($d(\mathcal{C}) \geq 2v + 1$).*

Důkaz.

První část tvrzení je jednoduché přeformulování definice kódu určujících u chyb. Pro druhou část tvrzení jsme ukázali ve větě 1.6 implikaci zprava doleva. Předpokládejme nyní, že $\mathcal{C}$ je kód opravující v chyb a že existují dvě různá kódová slova $\mathbf{c}$ a $\mathbf{d}$ tak, že $\mathrm{d}(\mathbf{c}, \mathbf{d}) = \mathrm{d}(\mathcal{C}) \leq 2v$.

Detekce chyb podle pravidla minimální vzdálenosti

Věta 1.7

Buďte u, v přirozená čísla. Pak kód $\mathcal{C}$ ***určí*** *u (****opraví*** *v) **chyb** právě tehdy, když $d(\mathcal{C}) \geq u + 1$ ($d(\mathcal{C}) \geq 2v + 1$).*

Důkaz.

První část tvrzení je jednoduché přeformulování definice kódu určujících u chyb. Pro druhou část tvrzení jsme ukázali ve větě 1.6 implikaci zprava doleva. Předpokládejme nyní, že $\mathcal{C}$ je kód opravující v chyb a že existují dvě různá kódová slova $\mathbf{c}$ a $\mathbf{d}$ tak, že $\mathrm{d}(\mathbf{c}, \mathbf{d}) = \mathrm{d}(\mathcal{C}) \leq 2v$.
Budeme chtít dokázat, že za předpokladu, že jsme odeslali kódové slovo $\mathbf{c}$ a nastalo nejvýše v chyb, je přesto možné, abychom podle pravidla minimální vzdálenosti obdrželi buď chybové hlášení nebo dekódovali obdržené slovo nesprávně jako $\mathbf{d}$. To pak bude spor s tím, že $\mathcal{C}$ opravuje v chyb. ∎

Detekce chyb podle pravidla minimální vzdálenosti

Pokračování důkazu Věty 1.7.

Nejdříve si uvědomme, že $\mathrm{d}(\mathbf{c}, \mathbf{d}) = \mathrm{d}(\mathcal{C}) \geq v + 1$. Jinak bychom totiž mohli převést **c** na **d** při nejvýše v chybách, které by zůstaly neopraveny. Můžeme pak předpokládat, že se **c** a **d** liší na prvních $k = \mathrm{d}(\mathcal{C})$ místech, přičemž $v + 1 \leq k \leq 2v$ (jinak provedeme permutaci souřadnic).

Detekce chyb podle pravidla minimální vzdálenosti

Pokračování důkazu Věty 1.7.

Nejdříve si uvědomme, že $\mathrm{d}(\mathbf{c}, \mathbf{d}) = \mathrm{d}(\mathcal{C}) \geq v + 1$. Jinak bychom totiž mohli převést **c** na **d** při nejvýše v chybách, které by zůstaly neopraveny. Můžeme pak předpokládat, že se **c** a **d** liší na prvních $k = \mathrm{d}(\mathcal{C})$ místech, přičemž $v + 1 \leq k \leq 2v$ (jinak provedeme permutaci souřadnic).

Uvažme nyní obdržené slovo **x**, které se shoduje se slovem **c** na prvních $k - v$ pozicích, dále se shoduje se slovem **d** na dalších v pozicích a shoduje se s oběma slovy **c** a **d** na posledních $n - k$ pozicích, tj.

$$\mathbf{x} = \underbrace{x_1 \ldots\ldots x_{k-v}}_{\text{shoduje se s } \mathbf{c}} \underbrace{x_{k-v+1} \ldots\ldots x_k}_{\text{shoduje se s } \mathbf{d}} \underbrace{x_{k+1} \ldots\ldots x_n}_{\text{shoduje se s oběma}}.$$

I

Detekce chyb podle pravidla minimální vzdálenosti

Pokračování důkazu Věty 1.7.

Protože nutně $\mathrm{d}(\mathbf{c}, \mathbf{x}) = v$, $\mathrm{d}(\mathbf{d}, \mathbf{x}) = k - v \leq v$, je buďto $\mathrm{d}(\mathbf{c}, \mathbf{x}) = \mathrm{d}(\mathbf{d}, \mathbf{x})$ (v tomto případě obdržíme chybové hlášení) nebo $\mathrm{d}(\mathbf{c}, \mathbf{x}) > \mathrm{d}(\mathbf{d}, \mathbf{x})$ (v tomto případě je $\mathbf{x}$ dekódováně nesprávně jako $\mathbf{d}$). ∎

Detekce chyb podle pravidla minimální vzdálenosti

Pokračování důkazu Věty 1.7.

Protože nutně $\mathrm{d}(\mathbf{c}, \mathbf{x}) = v$, $\mathrm{d}(\mathbf{d}, \mathbf{x}) = k - v \leq v$, je buďto $\mathrm{d}(\mathbf{c}, \mathbf{x}) = \mathrm{d}(\mathbf{d}, \mathbf{x})$ (v tomto případě obdržíme chybové hlášení) nebo $\mathrm{d}(\mathbf{c}, \mathbf{x}) > \mathrm{d}(\mathbf{d}, \mathbf{x})$ (v tomto případě je $\mathbf{x}$ dekódováně nesprávně jako $\mathbf{d}$). ∎

Definition 1.8

Pokud má blokový kód $\mathcal{C}$ právě M kódových slov délky n a má minimální vzdálenost d, mluvíme o (n, M, d)**-kódu.**

Detekce chyb podle pravidla minimální vzdálenosti

Pokračování důkazu Věty 1.7.

Protože nutně $\mathrm{d}(\mathbf{c}, \mathbf{x}) = v$, $\mathrm{d}(\mathbf{d}, \mathbf{x}) = k - v \leq v$, je buďto $\mathrm{d}(\mathbf{c}, \mathbf{x}) = \mathrm{d}(\mathbf{d}, \mathbf{x})$ (v tomto případě obdržíme chybové hlášení) nebo $\mathrm{d}(\mathbf{c}, \mathbf{x}) > \mathrm{d}(\mathbf{d}, \mathbf{x})$ (v tomto případě je $\mathbf{x}$ dekódováně nesprávně jako $\mathbf{d}$). ∎

Definition 1.8

Pokud má blokový kód $\mathcal{C}$ právě M kódových slov délky n a má minimální vzdálenost d, mluvíme o (n, M, d)**-kódu.**

Pro pevné n působí parametry M a d navzájem **proti sobě** tak, že zvětšení M způsobí zmenšení d a naopak.

Oprava chyb podle pravidla minimální vzdálenosti

Máme pak následující důsledek.

Důsledek 1.9

1. *(n, M, d)-kód $\mathcal{C}$ opraví právě v chyb tehdy a jen tehdy, když $d = 2v + 1$ nebo $d = 2v + 2$.*
2. *Kód $\mathcal{C}$ má minimální vzdálenost $d = \mathrm{d}(\mathcal{C})$ tehdy a jen tehdy, když opraví právě $\lfloor \frac{1}{2}(d - 1) \rfloor$ chyb.*

Oprava chyb podle pravidla minimální vzdálenosti

Máme pak následující důsledek.

Důsledek 1.9

1. *(n, M, d)-kód $\mathcal{C}$ opraví právě v chyb tehdy a jen tehdy, když $d = 2v + 1$ nebo $d = 2v + 2$.*
2. *Kód $\mathcal{C}$ má minimální vzdálenost $d = \mathrm{d}(\mathcal{C})$ tehdy a jen tehdy, když opraví právě $\lfloor \frac{1}{2}(d - 1) \rfloor$ chyb.*

Poznamenejme nyní, že určení chyby a její oprava jdou proti sobě, takže nemůžeme naráz dosáhnout jejich maximální úrovně. Uveďme si to na jednoduchém příkladě.

Oprava chyb podle pravidla minimální vzdálenosti

Příklad 1.10

Předpokládejme nyní, že kód $\mathcal{C}$ má minimální vzdálenost d. Je to tedy kód určující $d-1$ chyb a opravující $v = \lfloor \frac{1}{2}(d-1) \rfloor$ chyb. Pokud použijeme $\mathcal{C}$ pouze pro určení chyb, je schopen určit až $d-1$ chyb. Z druhé strany, pokud chceme na $\mathcal{C}$ opravit chybu, kdykoliv je to možné, pak je schopen opravit až v chyb, ale není schopen určit situaci, kdy nastalo více než v a méně než d chyb. Totiž, pokud nastalo více než v chyb, můžeme podle pravidla minimální vzdálenosti "opravit" přijatý řetězec na špatné kódové slovo a pak bude chyba nedetekovatelná.

Oprava chyb podle pravidla minimální vzdálenosti

Zkusme nyní podrobněji rozebrat případ, kdy kvalita detekování chyb kódu se sudou minimální vzdáleností závisí na tom, zda je či není kód použit pouze pro detekci chyb nebo zároveň jak pro detekci tak pro opravu chyb.

Oprava chyb podle pravidla minimální vzdálenosti

Zkusme nyní podrobněji rozebrat případ, kdy kvalita detekování chyb kódu se sudou minimální vzdáleností závisí na tom, zda je či není kód použit pouze pro detekci chyb nebo zároveň jak pro detekci tak pro opravu chyb.

Předpokládejme, že $\mathcal{C}$ je (n, M, d), kde $d = 2t + 2$. Jakožto pouze detekující kód dokáže $\mathcal{C}$ detekovat až $d - 1 = 2t + 1$ chyb.

Oprava chyb podle pravidla minimální vzdálenosti

Zkusme nyní podrobněji rozebrat případ, kdy kvalita detekování chyb kódu se sudou minimální vzdáleností závisí na tom, zda je či není kód použit pouze pro detekci chyb nebo zároveň jak pro detekci tak pro opravu chyb.

Předpokládejme, že $\mathcal{C}$ je (n, M, d), kde $d = 2t + 2$. Jakožto pouze detekující kód dokáže $\mathcal{C}$ detekovat až $d - 1 = 2t + 1$ chyb.

Na druhou stranu předpokládejme, že chceme použít $\mathcal{C}$ současně pro opravu chyb a detekci chyb.

Oprava chyb podle pravidla minimální vzdálenosti

Zkusme nyní podrobněji rozebrat případ, kdy kvalita detekování chyb kódu se sudou minimální vzdáleností závisí na tom, zda je či není kód použit pouze pro detekci chyb nebo zároveň jak pro detekci tak pro opravu chyb.

Předpokládejme, že $\mathcal{C}$ je (n, M, d), kde $d = 2t + 2$. Jakožto pouze detekující kód dokáže $\mathcal{C}$ detekovat až $d - 1 = 2t + 1$ chyb.

Na druhou stranu předpokládejme, že chceme použít $\mathcal{C}$ současně pro opravu chyb a detekci chyb.

Protože $\mathcal{C}$ opraví přesně t-chyb, pak pokud chceme maximalizovat možnosti opravy chyb, musíme „dovolit“, aby se opravilo jakýchkoliv t chyb.

Oprava chyb podle pravidla minimální vzdálenosti

To znamená, že pokud obdržíme slovo **x** a pokud existuje kódové slovo **c**, pro které $d(\mathbf{c}, \mathbf{x}) < t$, pak předpokládáme, že došlo k nejvýše t chybám, a opravíme **x** na **c**. Pokud žádné takové kódové slovo **c** neexistuje, pak můžeme prohlásit, že se vyskytly některé chyby.

Oprava chyb podle pravidla minimální vzdálenosti

To znamená, že pokud obdržíme slovo **x** a pokud existuje kódové slovo **c**, pro které $d(\mathbf{c}, \mathbf{x}) < t$, pak předpokládáme, že došlo k nejvýše t chybám, a opravíme **x** na **c**. Pokud žádné takové kódové slovo **c** neexistuje, pak můžeme prohlásit, že se vyskytly některé chyby.

Nyní, pokud při přenosu kódového slova **c** došlo k přesně $t + 1$ chybám, pak přijaté slovo **x** nemůže být ve vzdálednosti nejvýše t od žádného kódového slova **d**. Kdyby tomu tak bylo, pak by platilo

$$d(\mathbf{c}, \mathbf{d}) \leq d(\mathbf{c}, \mathbf{x}) + d(\mathbf{x}, \mathbf{d}) \leq t + 1 + t = 2t + 1 < d.$$

Oprava chyb podle pravidla minimální vzdálenosti

To znamená, že pokud obdržíme slovo **x** a pokud existuje kódové slovo **c**, pro které $d(\mathbf{c}, \mathbf{x}) < t$, pak předpokládáme, že došlo k nejvýše t chybám, a opravíme **x** na **c**. Pokud žádné takové kódové slovo **c** neexistuje, pak můžeme prohlásit, že se vyskytly některé chyby.

Nyní, pokud při přenosu kódového slova **c** došlo k přesně $t + 1$ chybám, pak přijaté slovo **x** nemůže být ve vzdálednosti nejvýše t od žádného kódového slova **d**. Kdyby tomu tak bylo, pak by platilo

$$d(\mathbf{c}, \mathbf{d}) \leq d(\mathbf{c}, \mathbf{x}) + d(\mathbf{x}, \mathbf{d}) \leq t + 1 + t = 2t + 1 < d.$$

Naše strategie vyžaduje, abychom detekovali, že došlo k

chybám, a $\mathcal{C}$ dokáže detekovat $t + 1$ chyb.

Oprava chyb podle pravidla minimální vzdálenosti

Pokud by však došlo k $t + 2$ chybám, pak protože $d = 2t + 2$, v alespoň jednom případě bude mít přijaté slovo **x** vzdálenost t od **špatného** kódového slova a přijaté slovo „opravíme“ nesprávně aniž budeme detekovat chyby. Proto $\mathcal{C}$ ne vždy detekuje $t + 2$ chyb.

Oprava chyb podle pravidla minimální vzdálenosti

Pokud by však došlo k $t + 2$ chybám, pak protože $d = 2t + 2$, v alespoň jednom případě bude mít přijaté slovo **x** vzdálenost t od **špatného** kódového slova a přijaté slovo „opravíme“ nesprávně aniž budeme detekovat chyby. Proto $\mathcal{C}$ ne vždy detekuje $t + 2$ chyb.

Předcházející úvahy lze zformulovat do následující věty.

Věta 1.11

*Pokud je (n, M, d)-kód $\mathcal{C}$, $d = 2t + 2$, použit pouze pro detekci chyb, pak **odhalí právě** $(2t + 1)$ **chyb**. Na druhou stranu, pokud je $\mathcal{C}$ použit pro opravu co největšího počtu chyb a zároveň pro odhalení chyb, pak $\mathcal{C}$ **opraví právě** t **chyb a dokáže současně detekovat** $t + 1$ **chyb**, ale nemůže vždy detekovat více než $t + 1$ chyb.*

Oprava chyb podle pravidla minimální vzdálenosti

Máme pak následující definici.

Definice 2

Uvažme následující strategii pro opravu/určení chyby. Buď u, v přirozená čísla. Obdržíme-li slovo **x** *a pokud má nejbližší kódové slovo* **c** *ke slovu* **x** *vzdálenost* ***nejvýše*** *v a existuje-li pouze jediné takové kódové slovo, budeme dekódovat* **x** *jako kódové slovo* **c**. *Pokud existuje více než jedno kódové slovo se stejnou minimální vzdáleností k* **x** *nebo má nejbližší kódové slovo vzdálenost větší než v, obdržíme na výstupu chybové hlášení.*
Řekneme, že **kód** $\mathcal{C}$ **zároveň opraví** v **chyb a určí** u **chyb**, *jestliže nastala alespoň jedna a nejvýše v chyb, výše popsaná strategie opraví tyto chyby a kdykoliv nastane alespoň $v + 1$ a nejvýše $u + v$ chyb, výše popsaná* ***strategie nahlásí chybu***.

Opravení v chyb a určení u chyb

Věta 1.12

Kód $\mathcal{C}$ ***zároveň opraví*** v ***chyb a určí*** u ***chyb*** *právě tehdy, když* $d(\mathcal{C}) \geq 2v + u + 1$.

Opravení v chyb a určení u chyb

Věta 1.12

Kód $\mathcal{C}$ ***zároveň opraví*** *v* ***chyb a určí*** *u* ***chyb*** *právě tehdy, když* $d(\mathcal{C}) \geq 2v + u + 1$.

Důkaz.

Předpokládejme nejprve, že $d(\mathcal{C}) \geq 2v + u + 1$. Obdržíme-li slovo **x** a pokud má nejbližší kódové slovo **c** ke slovu **x** vzdálenost nejvýše v a existuje-li alespoň jedno další takové kódové slovo **d**, máme

$$2v + u + 1 \leq \mathrm{d}(\mathbf{c}, \mathbf{d}) \leq \mathrm{d}(\mathbf{c}, \mathbf{x}) + \mathrm{d}(\mathbf{x}, \mathbf{d}) \leq 2v,$$

což je spor. Nutně tedy máme, že pokud obdržíme slovo **x** a nejbližší kódové slovo **c** ke slovu **x** má vzdálenost nejvýše v, je toto kódové slovo jediné s touto vlastností a podle pravidla minimální vzdálenosti budeme správně dekódovat. ∎

Opravení v chyb a určení u chyb

Pokračování důkazu Věty 1.12 .

Obdržíme-li slovo **x** a pokud má nejbližší kódové slovo **c** ke slovu **x** vzdálenost alespoň $v + 1$ a nejvýše $u + v$, při použití výše uvedené strategie dostaneme chybové hlášení.

Opravení v chyb a určení u chyb

Pokračování důkazu Věty 1.12 .

Obdržíme-li slovo **x** a pokud má nejbližší kódové slovo **c** ke slovu **x** vzdálenost alespoň $v + 1$ a nejvýše $u + v$, při použití výše uvedené strategie dostaneme chybové hlášení.

Předpokládejme nyní, že $\mathcal{C}$ je kód opravující v chyb a určující u chyb. Nechť dále $\mathrm{d}(\mathcal{C}) \leq 2v + u$. Nutně pak $2v + 1 \leq \mathrm{d}(\mathcal{C})$.

Opravení v chyb a určení u chyb

Pokračování důkazu Věty 1.12 .

Obdržíme-li slovo **x** a pokud má nejbližší kódové slovo **c** ke slovu **x** vzdálenost alespoň $v + 1$ a nejvýše $u + v$, při použití výše uvedené strategie dostaneme chybové hlášení.

Předpokládejme nyní, že $\mathcal{C}$ je kód opravující v chyb a určující u chyb. Nechť dále $\mathrm{d}(\mathcal{C}) \leq 2v + u$. Nutně pak $2v + 1 \leq \mathrm{d}(\mathcal{C})$.

Víme, že existují dvě různá kódová slova **c** a **d** tak, že $k = \mathrm{d}(\mathbf{c}, \mathbf{d}) = \mathrm{d}(\mathcal{C})$. Můžeme pak předpokládat, že se **c** a **d** liší na prvních $k = \mathrm{d}(\mathcal{C})$ místech, přičemž $2v + 1 \leq k \leq 2v + u$ (jinak provedeme permutaci souřadnic). ∎

Opravení v chyb a určení u chyb

Pokračování důkazu Věty 1.12 .

Uvažme nyní obdržené slovo **x**, které se shoduje se slovem **c** na prvních v pozicích, dále se shoduje se slovem **d** na dalších $k - v$ pozicích a shoduje se s oběma slovy **c** a **d** na posledních $n - k$ pozicích, tj.

$$\mathbf{x} = \underbrace{x_1 \dots x_v}_{\text{shoduje se s } \mathbf{c}} \underbrace{x_{v+1} \dots x_k}_{\text{shoduje se s } \mathbf{d}} \underbrace{x_{k+1} \dots x_n}_{\text{shoduje se s oběma}} .$$

Opravení v chyb a určení u chyb

Pokračování důkazu Věty 1.12 .

Uvažme nyní obdržené slovo **x**, které se shoduje se slovem **c** na prvních v pozicích, dále se shoduje se slovem **d** na dalších $k - v$ pozicích a shoduje se s oběma slovy **c** a **d** na posledních $n - k$ pozicích, tj.

$$\mathbf{x} = \underbrace{x_1 \dots x_v}_{\text{shoduje se s } \mathbf{c}} \underbrace{x_{v+1} \dots x_k}_{\text{shoduje se s } \mathbf{d}} \underbrace{x_{k+1} \dots x_n}_{\text{shoduje se s oběma}} .$$

Nutně pak $\mathrm{d}(\mathbf{c}, \mathbf{x}) = k - v$, $\mathrm{d}(\mathbf{d}, \mathbf{x}) = v$, $v + 1 \leq k - v \leq v + u$. Je-li tedy **c** odesláno a **x** je obdrženo, nutně je pak počet chyb při přenosu slova **c** (tj. číslo $k - v$) mezi $v + 1$ a $v + u$, uvažovaná strategie by nám měla dát na výstupu chybové hlášení, ale místo toho nám dekóduje **x** nesprávně na **d**. ∎

Maximální kódy

Definition 1.13

(n, M, d)-kód $\mathcal{C}$ se nazývá **maximální**, pokud není obsažen v žádném větším kódu se stejnou minimální vzdáleností tj. není obsažen v žádném $(n, M + 1, d)$-kódu.

Je zřejmé, že pro každý kód $\mathcal{C}$ můžeme vždy najít maximální kód $\mathcal{C}'$, který jej obsahuje. Přitom platí

Věta 1.14

*(n, M, d)-kód $\mathcal{C}$ je **maximální** právě tehdy, když pro všechna slova $\mathbf{x}$ existuje kódové slovo $\mathbf{c}$ s vlastností $d(\mathbf{x}, \mathbf{c}) < d$.*

Maximální kódy

Důkaz Věty 1.14.

(n, M, d)-kód $\mathcal{C}$ je maximální právě tehdy, když není obsažen v žádném $(n, M + 1, d)$-kódu. Předpokládejme, že existuje slovo $\mathbf{x}$ tak, že pro všechna kódová slova $\mathbf{c}$ platí $\mathrm{d}(\mathbf{x}, \mathbf{c}) \geq d$. Položíme-li $\mathcal{C}' = \mathcal{C} \cup \{x\}$, je pak evidentně $\mathcal{C}'$ $(n, M + 1, d)$-kód obsahující kód $\mathcal{C}$.

Maximální kódy

Důkaz Věty 1.14.

(n, M, d)-kód $\mathcal{C}$ je maximální právě tehdy, když není obsažen v žádném $(n, M+1, d)$-kódu. Předpokládejme, že existuje slovo $\mathbf{x}$ tak, že pro všechna kódová slova $\mathbf{c}$ platí $\mathrm{d}(\mathbf{x}, \mathbf{c}) \geq d$. Položíme-li $\mathcal{C}' = \mathcal{C} \cup \{x\}$, je pak evidentně $\mathcal{C}'$ $(n, M+1, d)$-kód obsahující kód $\mathcal{C}$.

Obráceně, nechť pro všechna slova $\mathbf{x}$ existuje kódové slovo $\mathbf{c}$ s vlastností $\mathrm{d}(\mathbf{x}, \mathbf{c}) < d$. Předpokládejme, že kód $\mathcal{C}$ není maximální tj. existuje $(n, M+1, d)$-kód obsahující kód $\mathcal{C}$. Vyberme slovo $\mathbf{x} \in \mathcal{C}' - \mathcal{C}$. Pak ale existuje kódové slovo $\mathbf{c} \in \mathcal{C} \subseteq \mathcal{C}'$ tak, že $\mathrm{d}(\mathcal{C}') \leq \mathrm{d}(\mathbf{x}, \mathbf{c}) < d$, spor. ∎

Maximální kódy

Poznamenejme, že pokud $(n, M, d)-$kód $\mathcal{C}$ není maximální, mohou nastat jak výhodné tak nevýhodné situace při jeho rozšíření na maximální kód $\mathcal{C}'$.

Maximální kódy

Poznamenejme, že pokud $(n, M, d)-$kód $\mathcal{C}$ není maximální, mohou nastat jak výhodné tak nevýhodné situace při jeho rozšíření na maximální kód $\mathcal{C}'$.

Víme, že kód $\mathcal{C}'$ rovněž opraví $\lfloor \frac{1}{2}(d-1) \rfloor$ chyb, což je dobré a přitom $\mathcal{C}'$ může zakódovat více zdrojových symbolů, což je rovněž dobré.

Maximální kódy

Poznamenejme, že pokud $(n, M, d)-$kód $\mathcal{C}$ není maximální, mohou nastat jak výhodné tak nevýhodné situace při jeho rozšíření na maximální kód $\mathcal{C}'$.

Víme, že kód $\mathcal{C}'$ rovněž opraví $\lfloor \frac{1}{2}(d-1) \rfloor$ chyb, což je dobré a přitom $\mathcal{C}'$ může zakódovat více zdrojových symbolů, což je rovněž dobré.

Ale zatímco $\mathcal{C}$ může být případně schopen opravit více než $\lfloor \frac{1}{2}(d-1) \rfloor$ chyb, kód $\mathcal{C}'$ nebude nikdy schopen opravit více než $\lfloor \frac{1}{2}(d-1) \rfloor$ chyb.

Maximální kódy

Příklad 1.15

Uvažme kód $\mathcal{C} = \{00000, 11000\}$*, který má minimální vzdálenost* 2*. Tento kód opravuje jednu chybu, ale je přitom schopen opravit další jiné chyby. Například, pokud bylo odesláno slovo* 00000 *a přijato slovo* 00111*, bude toto slovo správně dekódováno (totiž* $d(00000, 00111) = 3$*,* $d(11000, 00111) = 5$*), ačkoliv při přenosu nastaly tři chyby. Pokud ale doplníme* $\mathcal{C}$ *do maximálního kódu, bude dekódování chybné.*

Maximální kódy

Příklad 1.15

Uvažme kód $\mathcal{C} = \{00000, 11000\}$*, který má minimální vzdálenost* 2*. Tento kód opravuje jednu chybu, ale je přitom schopen opravit další jiné chyby. Například, pokud bylo odesláno slovo* 00000 *a přijato slovo* 00111*, bude toto slovo správně dekódováno (totiž* $d(00000, 00111) = 3$*,* $d(11000, 00111) = 5$*), ačkoliv při přenosu nastaly tři chyby. Pokud ale doplníme* $\mathcal{C}$ *do maximálního kódu, bude dekódování chybné.*

Z výše uvedeného příkladu vyplývá, že maximální kódy jsou nejlepší, pokud nás u kódu pouze zajímá předem určená schopnost opravení chyby. Je tedy daleko obtížnější zkoumat pravděpodobnost chyby při dekódování u kódů, které nejsou maximální. Pro maximální kódy je to jednodušší.

Maximální kódy

Věta 1.16

Buď $\mathcal{C}$ (n, M, d)-kód. Pak pro binární symetrický kanál s pravděpodobností chyby p je při použití dekódovacího pravidla minimální vzdálenosti

$$P(\text{nastala chyba při dekódování}) \leq 1 - \sum_{k=0}^{\lfloor \frac{1}{2}(d-1) \rfloor} \binom{n}{k} p^k (1-p)^{n-k}.$$

Je-li navíc kód $\mathcal{C}$ maximální, je

$$\sum_{k=d}^{n} \binom{n}{k} p^k (1-p)^{n-k} \leq P(\text{nastala chyba při dekódování}).$$

Maximální kódy

Důkaz Věty 1.16.

Každý (n, M, d)-kód $\mathcal{C}$ opravuje evidentně $\lfloor \frac{1}{2}(d-1) \rfloor$ chyb. Je tedy pravděpodobnost správného dekódování alespoň tak velká jako je pravděpodobnost, že nastane nejvýše $\lfloor \frac{1}{2}(d-1) \rfloor$ chyb tj.

$$P(\text{správné dekódování}) \geq \sum_{k=0}^{\lfloor \frac{1}{2}(d-1) \rfloor} \binom{n}{k} p^k (1-p)^{n-k}.$$

Maximální kódy

Důkaz Věty 1.16.

Každý (n, M, d)-kód $\mathcal{C}$ opravuje evidentně $\lfloor \frac{1}{2}(d-1) \rfloor$ chyb. Je tedy pravděpodobnost správného dekódování alespoň tak velká jako je pravděpodobnost, že nastane nejvýše $\lfloor \frac{1}{2}(d-1) \rfloor$ chyb tj.

$$P(\text{správné dekódování}) \geq \sum_{k=0}^{\lfloor \frac{1}{2}(d-1) \rfloor} \binom{n}{k} p^k (1-p)^{n-k}.$$

Máme pak

$$\begin{aligned} P(\text{nastala chyba při dekódování}) &= 1 - P(\text{správné dekódování}) \\ &\leq 1 - \sum_{k=0}^{\lfloor \frac{1}{2}(d-1) \rfloor} \binom{n}{k} p^k (1-p)^{n-k}. \end{aligned}$$

Maximální kódy

Pokračování Důkazu Věty 1.16.

Buď dále (n, M, d)-kód $\mathcal{C}$ maximální. Pak, je-li přeneseno slovo $\mathbf{c}$ a nastane-li d nebo více chyb, tj. $\mathrm{d}(\mathbf{c}, \mathbf{x}) \geq d$, bude nutně $\mathbf{x}$ bližší k jinému kódovému slovu $\mathbf{d} \neq \mathbf{c}$ a tedy při použití pravidla minimální vzdálenosti nastane dekódovací chyba.

Maximální kódy

Pokračování Důkazu Věty 1.16.

Buď dále (n, M, d)-kód $\mathcal{C}$ maximální. Pak, je-li přeneseno slovo **c** a nastane-li d nebo více chyb, tj. $\mathrm{d}(\mathbf{c}, \mathbf{x}) \geq d$, bude nutně **x** bližší k jinému kódovému slovu $\mathbf{d} \neq \mathbf{c}$ a tedy při použití pravidla minimální vzdálenosti nastane dekódovací chyba. Protože pravděpodobnost, že nastane právě k chyb při průchodem binárním symetrickým kanálem, je

$$\binom{n}{k} p^k (1-p)^{n-k},$$

obdržíme následující dolní hranici pro pravěpodobnost dekódovací chyby

$$\sum_{k=d}^{n} \binom{n}{k} p^k (1-p)^{n-k} \leq P(\text{nastala chyba při dekódování}).$$

Obsah

Základní pojmy

FI Užitečným prostředkem pro redukci množství práce při nalezení dobrých kódů je pojem ekvivalence kódů.
Předpokládejme, že máme (n, M, d)-kód $\mathcal{C}$.
Přirozeným způsobem jeho prezentace je pomocí matice o rozměrech $M \times n$, přičemž řádky jsou různá kódová slova.

Základní pojmy

FI Užitečným prostředkem pro redukci množství práce při nalezení dobrých kódů je pojem ekvivalence kódů.
Předpokládejme, že máme (n, M, d)-kód $\mathcal{C}$.
Přirozeným způsobem jeho prezentace je pomocí matice o rozměrech $M \times n$, přičemž řádky jsou různá kódová slova.

Předpokládejme nyní, že π je permutace množiny $\{1, 2, \ldots, n\}$ a pro každé kódové slovo $\mathbf{c} \in \mathcal{C}$ budeme aplikovat transformaci $\overline{\pi} : \mathcal{C} \to \mathcal{C}'$ definovanou předpisem $\overline{\pi}(\mathbf{c}) = (c_{\pi(1)}, \ldots, c_{\pi(n)})$.
Takovou transformaci nazýváme ***poziční permutací***.

Základní pojmy

Podobně, je-li π permutace množiny Σ, pak pro každý index i, $1 \leq i \leq n$ můžeme aplikovat transformaci $\hat{\pi}_i : \mathcal{C} \to \mathcal{C}'$ definovanou předpisem

$$\hat{\pi}_i(\mathbf{c})_j = \begin{cases} c_j & \text{pokud } i \neq j \\ \pi(c_i) & \text{pokud } i = j. \end{cases}$$

Základní pojmy

Podobně, je-li π permutace množiny Σ, pak pro každý index i, $1 \leq i \leq n$ můžeme aplikovat transformaci $\hat{\pi}_i : \mathcal{C} \to \mathcal{C}'$ definovanou předpisem

$$\hat{\pi}_i(\mathbf{c})_j = \begin{cases} c_j & \text{pokud } i \neq j \\ \pi(c_i) & \text{pokud } i = j. \end{cases}$$

Mluvíme pak o ***symbolové permutací***.

Základní pojmy

Podobně, je-li π permutace množiny Σ, pak pro každý index i, $1 \leq i \leq n$ můžeme aplikovat transformaci $\hat{\pi}_i : \mathcal{C} \to \mathcal{C}'$ definovanou předpisem

$$\hat{\pi}_i(\mathbf{c})_j = \begin{cases} c_j & \text{pokud } i \neq j \\ \pi(c_i) & \text{pokud } i = j. \end{cases}$$

Mluvíme pak o ***symbolové permutací***.

Lze-li kód $\mathcal{C}'$ získat z kódu $\mathcal{C}$ pomocí konečné posloupnosti pozičních nebo symbolových permutací, říkáme že ***kód*** $\mathcal{C}'$ ***je ekvivalentní kódu*** $\mathcal{C}$.

Příklady

Příklad 2.1

Předpokládejme, že máme kód $\mathcal{C}$ *délky* 5 *nad abecedou* $\Sigma = \{a, b, c\}$ *s kódovými slovy* $\mathbf{c}_1$, $\mathbf{c}_2$, $\mathbf{c}_3$ *a* $\mathbf{c}_4$ *tak, že*

$$\mathcal{C} = \begin{matrix} \mathbf{c}_1 \\ \mathbf{c}_2 \\ \mathbf{c}_3 \\ \mathbf{c}_4 \end{matrix} \begin{bmatrix} a & b & c & a & c \\ b & a & b & a & b \\ b & c & c & b & a \\ c & b & a & c & a \end{bmatrix}.$$

Aplikujeme-li permutaci $(1 \mapsto 2, 2 \mapsto 3, \ldots, 5 \mapsto 1)$, *obdržíme poziční permutaci a k ní odpovídající ekvivalentní kód je*

$$\mathcal{C}' = \begin{matrix} \mathbf{c}'_1 \\ \mathbf{c}'_2 \\ \mathbf{c}'_3 \\ \mathbf{c}'_4 \end{matrix} \begin{bmatrix} c & a & b & c & a \\ b & b & a & b & a \\ a & b & c & c & b \\ a & c & b & a & c \end{bmatrix}.$$

Příklady

Příklad 2.1 (Pokračování)

Podobně, aplikujeme-li permutaci $(a \mapsto b, b \mapsto c, c \mapsto a)$ *na první sloupec kódu* $\mathcal{C}'$*, obdržíme symbolovou permutaci a k ní odpovídající ekvivalentní kód je*

$$\mathcal{C}'' = \begin{matrix} \mathbf{c}_1'' \\ \mathbf{c}_2'' \\ \mathbf{c}_3'' \\ \mathbf{c}_4'' \end{matrix} \begin{bmatrix} a & a & b & c & a \\ c & b & a & b & a \\ b & b & c & c & b \\ b & c & b & a & c \end{bmatrix}.$$

Vlastnosti ekvivalentních kódů

Lemma 2.2

Jsou-li $\mathcal{C}$ a $\mathcal{C}'$ ekvivalentní kódy, jsou množiny vzdáleností kódových slov z $\mathcal{C}$ a $\mathcal{C}'$ stejné.

Vlastnosti ekvivalentních kódů

Lemma 2.2

Jsou-li $\mathcal{C}$ a $\mathcal{C}'$ ekvivalentní kódy, jsou množiny vzdáleností kódových slov z $\mathcal{C}$ a $\mathcal{C}'$ stejné.

Důkaz.

Protože jak poziční tak symbolová permutace zachovávají vzdálenost permutovaných slov, platí totéž i pro takovouto posloupnost permutací. ∎

Vlastnosti ekvivalentních kódů

Lemma 2.3

Je-li $\mathcal{C}$ kód délky n a $\mathbf{u}$ vektor délky n nad stejnou abecedou, pak existuje kód $\mathcal{C}'$, který je ekvivalentní s $\mathcal{C}$ a obsahuje vektor $\mathbf{u}$.

Důkaz.

První kódové slovo $\mathbf{c}_1$ lze převést na $\mathbf{u}$ pomocí nejvýše n symbolových permutací. ∎

Konstrukce kódů

Definition 2.4

Buď $\mathbf{x}, \mathbf{y}$ binární slova délky n. *Průnik* $\mathbf{x} \cap \mathbf{y}$ *binárních slov* $\mathbf{x}$ *a* $\mathbf{y}$ je binární řetězec délky n, který má jedničku přesně na těch místech, na kterých ji mají obě slova $\mathbf{x}$ a $\mathbf{y}$. Všude jinde má pak nuly. Jinak řečeno, $\mathbf{x} \cap \mathbf{y} = x_1 \cdot y_1 x_2 \cdot y_2 \ldots x_n \cdot y_n$.

Platí pak následující jednoduché lemma.

Lemma 2.5

Jsou-li $\mathbf{x}$ *a* $\mathbf{y}$ *binární řetězce délky n, pak*

$$d(\mathbf{x}, \mathbf{y}) = \mathrm{wt}(\mathbf{x}) + \mathrm{wt}(\mathbf{y}) - 2\mathrm{wt}(\mathbf{x} \cap \mathbf{y}).$$

Konstrukce kódů - rozšíření kódu

Důkaz Věty 2.5.

Položme $A_{11} = \{i : 1 \leq i \leq n, x_i = y_i = 1\}$, $a_{11} = \text{card}(A_{11})$,
$A_{10} = \{i : 1 \leq i \leq n, x_i = 1, y_i = 0\}$, $a_{10} = \text{card}(A_{10})$,
$A_{01} = \{i : 1 \leq i \leq n, x_i = 0, y_i = 1\}$, $a_{01} = \text{card}(A_{01})$,
$A_{00} = \{i : 1 \leq i \leq n, x_i = 0, y_i = 0\}$, $a_{00} = \text{card}(A_{00})$. Pak platí

$$\begin{aligned} \text{d}(\mathbf{x}, \mathbf{y}) = a_{10} + a_{01} &= (a_{11} + a_{10}) + (a_{11} + a_{01}) - 2a_{11} \\ &= \text{wt}(\mathbf{x}) + \text{wt}(\mathbf{y}) - 2\text{wt}(\mathbf{x} \cap \mathbf{y}), \end{aligned}$$

čímž je lemma dokázáno. ∎

Konstrukce kódů - rozšíření kódu

Důkaz Věty 2.5.

Položme $A_{11} = \{i : 1 \leq i \leq n, x_i = y_i = 1\}$, $a_{11} = \text{card}(A_{11})$,
$A_{10} = \{i : 1 \leq i \leq n, x_i = 1, y_i = 0\}$, $a_{10} = \text{card}(A_{10})$,
$A_{01} = \{i : 1 \leq i \leq n, x_i = 0, y_i = 1\}$, $a_{01} = \text{card}(A_{01})$,
$A_{00} = \{i : 1 \leq i \leq n, x_i = 0, y_i = 0\}$, $a_{00} = \text{card}(A_{00})$. Pak platí

$$\begin{aligned} \text{d}(\mathbf{x}, \mathbf{y}) = a_{10} + a_{01} &= (a_{11} + a_{10}) + (a_{11} + a_{01}) - 2a_{11} \\ &= \text{wt}(\mathbf{x}) + \text{wt}(\mathbf{y}) - 2\text{wt}(\mathbf{x} \cap \mathbf{y}), \end{aligned}$$

čímž je lemma dokázáno. ∎

Definice 3

Postup, při kterém přidáme ke všem kódovým slovům z daného kódu jednu nebo více dodatečných pozic, a tedy zvýšíme délku kódu, se nazývá **rozšíření kódu**.

Konstrukce kódů - kontrola parity

Nejznámější metoda rozšíření kódu se nazývá ***kontrola parity***.
Pro jednoduchost uvažme binární případ.

Konstrukce kódů - kontrola parity

Nejznámější metoda rozšíření kódu se nazývá ***kontrola parity***. Pro jednoduchost uvažme binární případ.

Je-li $\mathcal{C}$ binární (n, M, d)-kód, budeme konstruovat nový kód následovně. Ke každému kódovému slovu $\mathbf{c} = c_1 c_2 \ldots c_n \in \mathcal{C}$ přidáme dodatečný bit tak, že nové výsledné kódové slovo bude mít sudou váhu. Tedy, mělo-li $\mathbf{c}$ lichou váhu, přidáme jedničku, mělo-li $\mathbf{c}$ sudou váhu, přidáme nulu.

Konstrukce kódů - kontrola parity

Nejznámější metoda rozšíření kódu se nazývá ***kontrola parity***. Pro jednoduchost uvažme binární případ.

Je-li $\mathcal{C}$ binární (n, M, d)-kód, budeme konstruovat nový kód následovně. Ke každému kódovému slovu $\mathbf{c} = c_1 c_2 \ldots c_n \in \mathcal{C}$ přidáme dodatečný bit tak, že nové výsledné kódové slovo bude mít sudou váhu. Tedy, mělo-li $\mathbf{c}$ lichou váhu, přidáme jedničku, mělo-li $\mathbf{c}$ sudou váhu, přidáme nulu.

Označíme-li tedy výsledné slovo jako $\overline{\mathbf{c}}$, máme

$$\overline{\mathbf{c}} = \begin{cases} c_1 c_2 \ldots c_n 0 & \text{pokud } \mathrm{wt}(\mathbf{c}) \text{ je sudá,} \\ c_1 c_2 \ldots c_n 1 & \text{pokud } \mathrm{wt}(\mathbf{c}) \text{ je lichá.} \end{cases}$$

Konstrukce kódů - kontrola parity

Nejznámější metoda rozšíření kódu se nazývá ***kontrola parity***. Pro jednoduchost uvažme binární případ.

Je-li $\mathcal{C}$ binární (n, M, d)-kód, budeme konstruovat nový kód následovně. Ke každému kódovému slovu $\mathbf{c} = c_1 c_2 \ldots c_n \in \mathcal{C}$ přidáme dodatečný bit tak, že nové výsledné kódové slovo bude mít sudou váhu. Tedy, mělo-li $\mathbf{c}$ lichou váhu, přidáme jedničku, mělo-li $\mathbf{c}$ sudou váhu, přidáme nulu.

Označíme-li tedy výsledné slovo jako $\overline{\mathbf{c}}$, máme

$$\overline{\mathbf{c}} = \begin{cases} c_1 c_2 \ldots c_n 0 & \text{pokud } \mathrm{wt}(\mathbf{c}) \text{ je sudá,} \\ c_1 c_2 \ldots c_n 1 & \text{pokud } \mathrm{wt}(\mathbf{c}) \text{ je lichá.} \end{cases}$$

Nový kód $\overline{\mathcal{C}}$ má pak délku $n + 1$ a velikost M. Minimální vzdálenost kódu $\overline{\mathcal{C}}$ bude buď d nebo $d + 1$ a toto číslo bude záviset na tom, zda bude d sudé nebo liché číslo.

Konstrukce kódů - kontrola parity

Totiž, protože všechna kódová slova v $\overline{\mathcal{C}}$ mají sudou váhu, bude vzdálenost mezi každými dvěma slovy sudé číslo (to plyne z lemmatu 2.5).

Konstrukce kódů - kontrola parity

Totiž, protože všechna kódová slova v $\overline{\mathcal{C}}$ mají sudou váhu, bude vzdálenost mezi každými dvěma slovy sudé číslo (to plyne z lemmatu 2.5).

Je tedy i minimální vzdálenost kódu $\overline{\mathcal{C}}$ sudé číslo. Nutně pak dostáváme, že, je-li $\mathrm{d}(\mathcal{C}) = d$ sudé číslo a nastává pro pro slova $\mathbf{c}$, $\mathbf{d}$, pak nutně mají obě slova stejnou paritu a tedy nutně platí $\mathrm{d}(\mathbf{c}, \mathbf{d}) = \mathrm{d}(\overline{\mathbf{c}}, \overline{\mathbf{d}})$. Je tedy $\mathrm{d}(\mathcal{C}) = \mathrm{d}(\overline{\mathcal{C}})$.

Konstrukce kódů - kontrola parity

Totiž, protože všechna kódová slova v $\overline{\mathcal{C}}$ mají sudou váhu, bude vzdálenost mezi každými dvěma slovy sudé číslo (to plyne z lemmatu 2.5).

Je tedy i minimální vzdálenost kódu $\overline{\mathcal{C}}$ sudé číslo. Nutně pak dostáváme, že, je-li $\mathrm{d}(\mathcal{C}) = d$ sudé číslo a nastává pro pro slova **c**, **d**, pak nutně mají obě slova stejnou paritu a tedy nutně platí $\mathrm{d}(\mathbf{c}, \mathbf{d}) = \mathrm{d}(\overline{\mathbf{c}}, \overline{\mathbf{d}})$. Je tedy $\mathrm{d}(\mathcal{C}) = \mathrm{d}(\overline{\mathcal{C}})$.

Nechť je minimální vzdálenost kódu $\mathcal{C}$ liché číslo a nastává pro pro slova **c**, **d**, pak nutně má jedno ze slov sudou paritu (například **c**) a druhé lichou paritu (**d**).

Konstrukce kódů - kontrola parity

Pak $\mathrm{wt}(\mathbf{c}) = \mathrm{wt}(\overline{\mathbf{c}})$, $\mathrm{w}(\mathbf{d}) + 1 = \mathrm{w}(\overline{\mathbf{d}})$, $\mathrm{w}(\mathbf{c} \cap \mathbf{d}) = \mathrm{w}(\overline{\mathbf{c}} \cap \overline{\mathbf{d}})$. Tedy $\mathrm{d}(\mathcal{C}) + 1 = \mathrm{d}(\overline{\mathcal{C}})$.

Konstrukce kódů - kontrola parity

Pak $\mathrm{wt}(\mathbf{c}) = \mathrm{wt}(\overline{\mathbf{c}})$, $\mathrm{w}(\mathbf{d}) + 1 = \mathrm{w}(\overline{\mathbf{d}})$, $\mathrm{w}(\mathbf{c} \cap \mathbf{d}) = \mathrm{w}(\overline{\mathbf{c}} \cap \overline{\mathbf{d}})$. Tedy $\mathrm{d}(\mathcal{C}) + 1 = \mathrm{d}(\overline{\mathcal{C}})$.

V obou případech pak máme

$$\lfloor \frac{1}{2}(\mathrm{d}(\mathcal{C}) - 1) \rfloor = \lfloor \frac{1}{2}(\mathrm{d}(\overline{\mathcal{C}}) - 1) \rfloor.$$

Konstrukce kódů - kontrola parity

Pak $\mathrm{wt}(\mathbf{c}) = \mathrm{wt}(\overline{\mathbf{c}})$, $\mathrm{w}(\mathbf{d}) + 1 = \mathrm{w}(\overline{\mathbf{d}})$, $\mathrm{w}(\mathbf{c} \cap \mathbf{d}) = \mathrm{w}(\overline{\mathbf{c}} \cap \overline{\mathbf{d}})$. Tedy $\mathrm{d}(\mathcal{C}) + 1 = \mathrm{d}(\overline{\mathcal{C}})$.

V obou případech pak máme

$$\lfloor \frac{1}{2}(\mathrm{d}(\mathcal{C}) - 1) \rfloor = \lfloor \frac{1}{2}(\mathrm{d}(\overline{\mathcal{C}}) - 1) \rfloor.$$

Z toho pak plyne, že se nám při použití kontroly parity nezvýší schopnost opravit chybu.

Konstrukce kódů - kontrola parity

Pak $\mathrm{wt}(\mathbf{c}) = \mathrm{wt}(\overline{\mathbf{c}})$, $\mathrm{w}(\mathbf{d}) + 1 = \mathrm{w}(\overline{\mathbf{d}})$, $\mathrm{w}(\mathbf{c} \cap \mathbf{d}) = \mathrm{w}(\overline{\mathbf{c}} \cap \overline{\mathbf{d}})$. Tedy $\mathrm{d}(\mathcal{C}) + 1 = \mathrm{d}(\overline{\mathcal{C}})$.

V obou případech pak máme

$$\lfloor \frac{1}{2}(\mathrm{d}(\mathcal{C}) - 1) \rfloor = \lfloor \frac{1}{2}(\mathrm{d}(\overline{\mathcal{C}}) - 1) \rfloor.$$

Z toho pak plyne, že se nám při použití kontroly parity nezvýší schopnost opravit chybu.

Obecně pak, máme-li kódovou abecedu vybránu tak, že nám tvoří konečné pole, řekněme $\mathbf{Z}_p$, kde p je prvočíslo, nebo obecně $\mathbb{F}_q$, můžeme definovat ***kontrolu parity*** jako

$$\overline{\mathbf{c}} = c_1 c_2 \ldots c_n c_{n+1}, \text{kde } c_{n+1} = -\sum_{i=1}^{n} c_i.$$

Konstrukce kódů - zkrácení kódu

Definice 4

Postup, při kterém odebereme ta kódová slova z daného kódu, která se liší na určené pozici i od určeného symbolu s, a ze zbývajících slov tuto pozici odstraníme, a tedy zkrátíme délku kódu, se nazývá **zkrácení kódu typu** $x_i = s$.

Je-li pak $\mathcal{C}$ (n, M, d)-kód, má zkrácený kód $\mathcal{C}^{\odot}$ délku $n - 1$ a minimální vzdálenost alespoň d.

Konstrukce kódů - zkrácení kódu

Definice 4

Postup, při kterém odebereme ta kódová slova z daného kódu, která se liší na určené pozici i od určeného symbolu s, a ze zbývajících slov tuto pozici odstraníme, a tedy zkrátíme délku kódu, se nazývá **zkrácení kódu typu** $x_i = s$.

Je-li pak $\mathcal{C}$ (n, M, d)-kód, má zkrácený kód $\mathcal{C}^{\odot}$ délku $n - 1$ a minimální vzdálenost alespoň d.

Opravdu, zkrácení kódu může mít za důsledek podstatné zvětšení minimální vzdálenosti tedy i schopnosti opravit nového kódu, protože můžeme odstranit ta kódová slova, která se "**špatně chovají vzhledem ke vzdálenosti**".

Konstrukce kódů - zkrácení kódu

Definice 4

Postup, při kterém odebereme ta kódová slova z daného kódu, která se liší na určené pozici i od určeného symbolu s, a ze zbývajících slov tuto pozici odstraníme, a tedy zkrátíme délku kódu, se nazývá **zkrácení kódu typu** $x_i = s$.

Je-li pak $\mathcal{C}$ (n, M, d)-kód, má zkrácený kód $\mathcal{C}^{\odot}$ délku $n - 1$ a minimální vzdálenost alespoň d.

Opravdu, zkrácení kódu může mít za důsledek podstatné zvětšení minimální vzdálenosti tedy i schopnosti opravit nového kódu, protože můžeme odstranit ta kódová slova, která se "**špatně chovají vzhledem ke vzdálenosti**".

Samozřejmě ale zkrácením kódu se nám **zmenší** i počet kódových slov, což není zrovna lákavé.

Konstrukce kódů - zkrácení kódu

Věta 2.6

Buď d liché přirozené číslo. Pak existuje binární (n, M, d)-kód právě tehdy, když existuje binární $(n+1, M, d+1)$-kód.

Konstrukce kódů - zkrácení kódu

Věta 2.6

Buď d liché přirozené číslo. Pak existuje binární (n, M, d)-kód právě tehdy, když existuje binární $(n+1, M, d+1)$-kód.

Důkaz.

Pokud existuje binární $(n+1, M, d+1)$-kód $\mathcal{C}$, můžeme snadno zkonstruovat binární (n, M, d)-kód. Jednoduše vybereme dvě kódová slova **c** a **d** tak, že $\mathrm{d}(\mathbf{c}, \mathbf{d}) = d + 1$, najděme pozici, na které se liší a odebereme tuto pozici z každého kódového slova. Nový kód označíme $\mathcal{C}'$.

Konstrukce kódů - zkrácení kódu

Věta 2.6

Buď d liché přirozené číslo. Pak existuje binární (n, M, d)-kód právě tehdy, když existuje binární $(n+1, M, d+1)$-kód.

Důkaz.

Pokud existuje binární $(n+1, M, d+1)$-kód $\mathcal{C}$, můžeme snadno zkonstruovat binární (n, M, d)-kód. Jednoduše vybereme dvě kódová slova $\mathbf{c}$ a $\mathbf{d}$ tak, že $\mathrm{d}(\mathbf{c}, \mathbf{d}) = d + 1$, najděme pozici, na které se liší a odebereme tuto pozici z každého kódového slova. Nový kód označíme $\mathcal{C}'$.

Nutně pak mají nová zkrácená slova $\mathbf{c}'$ a $\mathbf{d}'$ vzdálenost $\mathrm{d}(\mathbf{c}', \mathbf{d}') = d$ a žádná jiná dvě zkrácená slova nemají od sebe menší vzdálenost než d. Celkem je tedy $\mathcal{C}'$ binární (n, M, d)-kód. ∎

Konstrukce kódů - zkrácení kódu

Pokračování důkazu Věty 2.6.

Obráceně, předpokládejme, že máme binární (n, M, d)-kód $\mathcal{D}$ (d liché). Kód $\overline{\mathcal{D}}$, který vznikl jako kód kontroly parity z $\mathcal{D}$, má délku $n + 1$, velikost M a minimální vzdálenost $d + 1$.

∎

Obsah

Hlavní problém teorie kódování

Definice 5

FI *Buď dána přirozená čísla d, n, q. Položme $A_q(n, d)$ jakožto maximální M takové, že existuje q-ární (n, M, d)-kód. Každý takový q-ární (n, M, d)-kód nazýváme* **optimální**.

Hlavní problém teorie kódování

Definice 5

FI *Buď dána přirozená čísla d, n, q. Položme $A_q(n, d)$ jakožto maximální M takové, že existuje q-ární (n, M, d)-kód. Každý takový q-ární (n, M, d)-kód nazýváme* **optimální**.

Čísla $A_q(n, d)$ hrají ústřední roli v teorii kódování a na jejich nalezení bylo vynaloženo velké úsilí. Často se mluví o ***hlavním problému teorie kódování***. V dalším pro ilustraci určíme jisté hodnoty $A_q(n, d)$ pro malé hodnoty n a d a dokážeme obecná tvrzení o těchto číslech.

Hlavní problém teorie kódování

Definice 5

FI *Buď dána přirozená čísla d, n, q. Položme $A_q(n, d)$ jakožto maximální M takové, že existuje q-ární (n, M, d)-kód. Každý takový q-ární (n, M, d)-kód nazýváme* **optimální**.

Čísla $A_q(n, d)$ hrají ústřední roli v teorii kódování a na jejich nalezení bylo vynaloženo velké úsilí. Často se mluví o ***hlavním problému teorie kódování***. V dalším pro ilustraci určíme jisté hodnoty $A_q(n, d)$ pro malé hodnoty n a d a dokážeme obecná tvrzení o těchto číslech.

Poznamenejme, že abychom dokázali, že $A_q(n, d) = K$ pro jisté přirozené číslo K, stačí ověřit, že $A_q(n, d) \leq K$ a následně najít vhodný q-ární (n, K, d')-kód $\mathcal{C}$, kde $d \leq d'$. Pak totiž $K \leq A_q(n, d') \leq A_q(n, d)$.

Hlavní problém teorie kódování

Věta 3.1

Je-li d sudé číslo, je $A_2(n, d) = A_2(n-1, d-1)$.

Důkaz.

Plyne okamžitě z věty 2.6. Totiž pak nutně platí $A_2(n, d) \leq A_2(n-1, d-1)$ a $A_2(n, d) \geq A_2(n-1, d-1)$. ∎

Hlavní problém teorie kódování

Věta 3.1

Je-li d sudé číslo, je $A_2(n, d) = A_2(n-1, d-1)$.

Důkaz.

Plyne okamžitě z věty 2.6. Totiž pak nutně platí $A_2(n, d) \leq A_2(n-1, d-1)$ a $A_2(n, d) \geq A_2(n-1, d-1)$. ∎

Důsledek 3.2

Je-li d sudé číslo a existuje binární (n, M, d)-kód, pak existuje binární (n, M, d)-kód, ve kterém mají všechna kódová slova sudou váhu.

Důkaz.

Dle Věty 3.1 nutně existuje binární kód $(n-1, M, d-1)$-kód a pomocí operace kontroly parity existuje binární (n, M, d)-kód, ve kterém mají všechna kódová slova sudou váhu. ■

Hlavní problém teorie kódování

Následující dva snadné výsledky nám budou ilustrovat, jakým způsobem můžeme určit hodnoty $A_2(n, d)$ pro malé hodnoty n a d.

Použijeme přitom lemma 2.3, ze kterého plyne, že pro daný (n, M, d)-kód $\mathcal{C}$ existuje ekvivalentní (n, M, d)-kód $\mathcal{C}'$, který obsahuje nulové slovo (samozřejmě za předpokladu, že kódová abeceda obsahuje 0 – jinak ji lze dodat záměnou za jiný symbol).

Můžeme tedy v dalším předpokládat, že naše kódy obsahují nulové slovo.

Věta 3.3

Platí $A_2(4, 3) = 2$.

$A_2(4, 3) = 2$

Důkaz.

Buď $\mathcal{C}$ nějaký $(4, M, 3)$-kód. Můžeme bez újmy na obecnosti předpokládat, že $\mathbf{0} = 0000 \in \mathcal{C}$. Protože $\mathrm{d}(\mathcal{C}) = 3$, libovolné další kódové slovo $\mathbf{c}$ z $\mathcal{C}$ musí splňovat $\mathrm{d}(\mathbf{c}, \mathbf{0}) \geq 3$ a tedy musí obsahovat alespoň tři jedničky. Máme tedy celkem pět možností pro nenulová slova ležící v $\mathcal{C}$, a to

$$1110, 1101, 1011, 0111, 1111.$$

Ale každá takováto dvě slova mají vzdálenost nejvýše 2. Proto pouze jedno z nich může být obsaženo v $\mathcal{C}$. Platí tedy $A_2(4, 3) \leq 2$. Dále platí, protože $\mathcal{C} = \{0000, 1110\}$ je $(4, 2, 3)$-kód, máme $A_2(4, 3) \geq 2$ a tedy celkem $A_2(4, 3) = 2$.
∎

$A_2(5, 3) = 4$

Věta 3.4

Platí $A_2(5, 3) = 4$.

Důkaz.

Buď $\mathcal{C}$ nějaký $(5, M, 3)$-kód. Můžeme bez újmy na obecnosti předpokládat, že $\mathbf{0} = 00000 \in \mathcal{C}$ a přitom pro vhodné $\mathbf{c}$ z $\mathcal{C}$ platí $\mathrm{d}(\mathbf{0}, \mathbf{c}) = 3$, $c_1 = 0$. Uvažme nyní zkrácení $\mathcal{C}^{\odot}$ typu $x_1 = 0$. Víme pak, že $\mathbf{0}^{\odot}, \mathbf{c}^{\odot} \in \mathcal{C}^{\odot}$ a $\mathrm{d}(\mathbf{0}^{\odot}, \mathbf{c}^{\odot}) = 3$. Dále víme, že $A_2(4, 3) = 2$. Tedy i $\mathrm{card}(\mathcal{C}^{\odot}) = 2$.
Definujme nyní zkrácený kód $\mathcal{C}^{\ominus}$ jakožto zkrácení typu typu $x_1 = 1$. Pak buďto $\mathrm{card}(\mathcal{C}^{\ominus}) = 1$ nebo $\mathrm{card}(\mathcal{C}^{\ominus}) > 1$ a $\mathrm{d}(\mathcal{C}^{\ominus}) = 3$. V posledním případě pak $\mathrm{card}(\mathcal{C}^{\ominus}) \leq A_2(4, 3) = 2$. Celkem tedy $\mathrm{card}(\mathcal{C}^{\ominus}) + \mathrm{card}(\mathcal{C}^{\odot}) = \mathrm{card}(\mathcal{C}) \leq 4$. Platí tedy $A_2(5, 3) \leq 4$. Dále platí, protože $\mathcal{C} = \{00000, 11100, 00111, 11011\}$ je $(5, 4, 3)$-kód, máme $A_2(5, 3) \geq 4$ a tedy celkem $A_2(5, 3) = 4$. ∎

Vlastnosti maximálních kódů

Věta 3.5

Platí následující:

1. $A_q(n, d) \leq q^n$ *pro všechna* $1 \leq d \leq n$;
2. $A_q(n, 1) = q^n$;
3. $A_q(n, n) = q$.

Vlastnosti maximálních kódů

Věta 3.5

Platí následující:

1. $A_q(n, d) \leq q^n$ *pro všechna* $1 \leq d \leq n$;
2. $A_q(n, 1) = q^n$;
3. $A_q(n, n) = q$.

Důkaz.

První tvrzení platí, protože pro každý kód $\mathcal{C}$ je $\mathcal{C} \subseteq V_n(q)$ tj. $\mathrm{card}(\mathcal{C}) \leq q^n$.

Vlastnosti maximálních kódů

Věta 3.5

Platí následující:

1. $A_q(n, d) \leq q^n$ *pro všechna* $1 \leq d \leq n$;
2. $A_q(n, 1) = q^n$;
3. $A_q(n, n) = q$.

Důkaz.

První tvrzení platí, protože pro každý kód $\mathcal{C}$ je $\mathcal{C} \subseteq V_n(q)$ tj. $\mathrm{card}(\mathcal{C}) \leq q^n$.
Druhé tvrzení plyne z toho, že uvážíme-li $\mathcal{C} = V_n(q)$, máme $\mathrm{d}(V_n(q)) = 1$.

Vlastnosti maximálních kódů

Věta 3.5

Platí následující:

1. $A_q(n, d) \leq q^n$ *pro všechna* $1 \leq d \leq n$;
2. $A_q(n, 1) = q^n$;
3. $A_q(n, n) = q$.

Důkaz.

První tvrzení platí, protože pro každý kód $\mathcal{C}$ je $\mathcal{C} \subseteq V_n(q)$ tj. $\mathrm{card}(\mathcal{C}) \leq q^n$.
Druhé tvrzení plyne z toho, že uvážíme-li $\mathcal{C} = V_n(q)$, máme $\mathrm{d}(V_n(q)) = 1$.
Třetí část plyne z toho, že se kódová slova musí lišit na všech pozicích a takových kódových slov můžeme vybrat nejvýše q.
Ale máme, pro kód $\mathcal{D} = \{\mathbf{0}, \ldots, \mathbf{q\text{-}1}\}$, že $\mathcal{D}$ je (n, q, n)-kód. ∎

Vlastnosti maximálních kódů

Už pro malé hodnoty q, n a d není velikost $A_q(n, d)$ známa.
Následující tabulka shrnuje většinu našich znalostí o $A_2(n, d)$.

n	$d = 3$	$d = 5$	d=7
5	4	2	-
6	8	2	-
7	16	2	2
8	20	4	2
9	40	6	2
10	72-79	12	2
11	144-158	24	4
12	256	32	4
13	512	64	8
14	1024	128	16
15	2048	256	32
16	2560-3276	256-340	36-37

Vlastnosti maximálních kódů

Poznamenejme, že problém určení $A_2(n, d)$ je problémem konečných geometrií.

Vlastnosti maximálních kódů

Poznamenejme, že problém určení $A_2(n, d)$ je problémem konečných geometrií. Pro odhad $A_q(n, d)$ platí následující jednoduché tvrzení.

Věta 3.6

Pro všechna $n \geq 2$,

$$A_q(n, d) \leq qA_q(n-1, d). \tag{3.1}$$

Vlastnosti maximálních kódů

Poznamenejme, že problém určení $A_2(n,d)$ je problémem konečných geometrií. Pro odhad $A_q(n,d)$ platí následující jednoduché tvrzení.

Věta 3.6

Pro všechna $n \geq 2$,

$$A_q(n,d) \leq qA_q(n-1,d). \tag{3.1}$$

Důkaz.

Buď $\mathcal{C}$ kód realizující hodnotu $A_q(n,d)$. Uvažme nyní zkrácení $\mathcal{C}_j$ typu $x_n = j$. Pak nutně $\mathrm{card}(\mathcal{C}_j) \leq A_q(n-1,d)$ (mohou totiž nastat pouze dva případy: $\mathrm{card}(\mathcal{C}_j) = 1$, což evidentně platí, a $K = \mathrm{card}(\mathcal{C}_j) > 1$, kde pak $\mathcal{C}_j$ je $(n-1,K,d')$-kód, $d' \geq d$ a tedy tvrzení rovněž platí). Celkem pak $\mathcal{C} = \bigcup_{j=0}^{q-1} \mathcal{C}_j$ tj.

$A_q(n,d) = \sum_{j=0}^{q-1} \mathrm{card}(\mathcal{C}_j) \leq qA_q(n-1,d)$. ∎

Vlastnosti maximálních kódů

Cvičení 3.7

1. *Ukažte, že* $A_2(3, 2) = 4$.

Obsah

Dolní a horní hranice $A_q(n, d)$

FI Určeme nejprve horní hranici (sphere-packing upper bound) čísla $A_q(n, d)$.

Dolní a horní hranice $A_q(n, d)$

FI Určeme nejprve horní hranici (sphere-packing upper bound) čísla $A_q(n, d)$.

Lemma 4.1

Nechť $e = \lfloor \frac{1}{2}(d-1) \rfloor$. *Pak platí*

$$A_q(n, d) \sum_{k=0}^{e} \binom{n}{k} (q-1)^k \leq q^n. \tag{4.1}$$

Dolní a horní hranice $A_q(n, d)$

Důkaz Lemmatu 4.1.

Buď $\mathcal{C}$ blokový kód s minimální vzdáleností d a počtem kódových slov $A_q(n, d)$. Je-li pak $\mathbf{S}_e(\mathbf{x})$ koule o poloměru e se středem $\mathbf{x}$, máme pro každou dvojici kódových slov $\mathbf{x}$ a $\mathbf{y}$, že

$$\mathbf{S}_e(\mathbf{x}) \cap \mathbf{S}_e(\mathbf{y}) = \emptyset.$$

Dolní a horní hranice $A_q(n, d)$

Důkaz Lemmatu 4.1.

Buď $\mathcal{C}$ blokový kód s minimální vzdáleností d a počtem kódových slov $A_q(n, d)$. Je-li pak $\mathbf{S}_e(\mathbf{x})$ koule o poloměru e se středem $\mathbf{x}$, máme pro každou dvojici kódových slov $\mathbf{x}$ a $\mathbf{y}$, že

$$\mathbf{S}_e(\mathbf{x}) \cap \mathbf{S}_e(\mathbf{y}) = \emptyset.$$

Ale je evidentní, že

$$|\mathbf{S}_e(\mathbf{x})| = \sum_{k=0}^{e} \binom{n}{k} (q-1)^k. \tag{4.2}$$

Dolní a horní hranice $A_q(n, d)$

Důkaz Lemmatu 4.1.

Buď $\mathcal{C}$ blokový kód s minimální vzdáleností d a počtem kódových slov $A_q(n, d)$. Je-li pak $\mathbf{S}_e(\mathbf{x})$ koule o poloměru e se středem $\mathbf{x}$, máme pro každou dvojici kódových slov $\mathbf{x}$ a $\mathbf{y}$, že

$$\mathbf{S}_e(\mathbf{x}) \cap \mathbf{S}_e(\mathbf{y}) = \emptyset.$$

Ale je evidentní, že

$$|\mathbf{S}_e(\mathbf{x})| = \sum_{k=0}^{e} \binom{n}{k} (q-1)^k. \tag{4.2}$$

Pravá strana nerovnosti (4.1) je celkový počet slov délky n nad abecedou o q symbolech.

Dolní a horní hranice $A_q(n, d)$

Důkaz Lemmatu 4.1.

Buď $\mathcal{C}$ blokový kód s minimální vzdáleností d a počtem kódových slov $A_q(n, d)$. Je-li pak $\mathbf{S}_e(\mathbf{x})$ koule o poloměru e se středem $\mathbf{x}$, máme pro každou dvojici kódových slov $\mathbf{x}$ a $\mathbf{y}$, že

$$\mathbf{S}_e(\mathbf{x}) \cap \mathbf{S}_e(\mathbf{y}) = \emptyset.$$

Ale je evidentní, že

$$|\mathbf{S}_e(\mathbf{x})| = \sum_{k=0}^{e} \binom{n}{k} (q-1)^k. \tag{4.2}$$

Pravá strana nerovnosti (4.1) je celkový počet slov délky n nad abecedou o q symbolech.
Levá strana je počet prvků obsažených v disjunktním sjednocení koulí, jejichž středy jsou navzájem různá kódová slova.
Takovýchto různých kódových slov je $A_q(n, d)$. Tedy dostáváme nerovnost (4.1).

Dolní a horní hranice $A_q(n, d)$

Podobně platí

Lemma 4.2

(**Gilbert-Varshamovova hranice**)

$$A_q(n, d) \sum_{k=0}^{d-1} \binom{n}{k} (q-1)^k \geq q^n. \tag{4.3}$$

Dolní a horní hranice $A_q(n, d)$

Podobně platí

Lemma 4.2

(**Gilbert-Varshamovova hranice**)

$$A_q(n, d) \sum_{k=0}^{d-1} \binom{n}{k} (q-1)^k \geq q^n. \tag{4.3}$$

Důkaz.

Buď $\mathcal{C}$ (n, M, d)-kód s maximálním počtem kódových slov. Pak zcela jistě neexistuje vektor z $V_n(q) - \mathcal{C}$, jehož vzdálenost od všech kódových slov je alespoň d, jinak by totiž M nebyl maximální počet kódových slov.

Dolní a horní hranice $A_q(n, d)$

Podobně platí

Lemma 4.2

(**Gilbert-Varshamovova hranice**)

$$A_q(n, d) \sum_{k=0}^{d-1} \binom{n}{k} (q-1)^k \geq q^n. \tag{4.3}$$

Důkaz.

Buď $\mathcal{C}$ (n, M, d)-kód s maximálním počtem kódových slov. Pak zcela jistě neexistuje vektor z $V_n(q) - \mathcal{C}$, jehož vzdálenost od všech kódových slov je alespoň d, jinak by totiž M nebyl maximální počet kódových slov.
Jinak řečeno, koule o poloměru d musí pokrývat celý prostor $V_n(q)$. Ale to je přesně podmínka (4.3). ∎

Poloměr pokrytí blokového kódu

Definice 6

Poloměr pokrytí *blokového kódu $\mathcal{C}$ je nejmenší poloměr ρ takový, že*

$$\mathbb{F}_q^n \subseteq \bigcup_{\mathbf{c}\in\mathbf{C}} S_\rho(\mathbf{c}).$$

Dále pro každé $\mathbf{f} \in \mathbb{F}_q^n$ klademe $\mathrm{d}(\mathbf{f}, \mathcal{C}) = \min_{\mathbf{c}\in\mathcal{C}} \mathrm{d}(\mathbf{c}, \mathbf{f})$.

Poloměr pokrytí blokového kódu

Definice 6

Poloměr pokrytí *blokového kódu $\mathcal{C}$ je nejmenší poloměr ρ takový, že*

$$\mathbb{F}_q^n \subseteq \bigcup_{\mathbf{c}\in\mathbf{C}} S_\rho(\mathbf{c}).$$

Dále pro každé $\mathbf{f} \in \mathbb{F}_q^n$ klademe $\mathrm{d}(\mathbf{f}, \mathcal{C}) = \min_{\mathbf{c}\in\mathcal{C}} \mathrm{d}(\mathbf{c}, \mathbf{f})$.

Poloměr pokrytí je další charakterizací kódů, nemá však tak hojné uplatnění jako minimální vzdálenost.

Poloměr pokrytí blokového kódu

Věta 4.3

Nechť $\mathcal{C}$ je blokový kód délky n. Pak ρ je poloměr pokrytí kódu $\mathcal{C}$ právě tehdy, když platí

$$\rho = \max_{\mathbf{f} \in \mathbb{F}_q^n} \min_{\mathbf{c} \in \mathcal{C}} \mathrm{d}(\mathbf{c}, \mathbf{f}).$$

Poloměr pokrytí blokového kódu

Věta 4.3

Nechť $\mathcal{C}$ je blokový kód délky n. Pak ρ je poloměr pokrytí kódu $\mathcal{C}$ právě tehdy, když platí

$$\rho = \max_{\mathbf{f} \in \mathbb{F}_q^n} \min_{\mathbf{c} \in \mathcal{C}} \mathrm{d}(\mathbf{c}, \mathbf{f}).$$

Důkaz Věty 4.3.

Nechť $\mathbb{F}_q^n \subseteq \bigcup_{\mathbf{c} \in \mathcal{C}} S_\rho(\mathbf{c})$, kde ρ je minimální. Pak pro každé $\mathbf{f} \in \mathbb{F}_q^n$ existuje $\mathbf{c} \in \mathcal{C}$ takové, že $\mathrm{d}(\mathbf{c}, \mathbf{f}) \leq \rho$, a současně existují $\mathbf{f}' \in \mathbb{F}_q^n$ a $\mathbf{c}' \in \mathbf{C}$ splňující $d(\mathbf{c}', \mathbf{f}') = \rho$.

Poloměr pokrytí blokového kódu

Věta 4.3

Nechť $\mathcal{C}$ je blokový kód délky n. Pak ρ je poloměr pokrytí kódu $\mathcal{C}$ právě tehdy, když platí

$$\rho = \max_{\mathbf{f}\in\mathbb{F}_q^n} \min_{\mathbf{c}\in\mathcal{C}} \mathrm{d}(\mathbf{c}, \mathbf{f}).$$

Důkaz Věty 4.3.

Nechť $\mathbb{F}_q^n \subseteq \bigcup_{\mathbf{c}\in\mathcal{C}} S_\rho(\mathbf{c})$, kde ρ je minimální. Pak pro každé $\mathbf{f} \in \mathbb{F}_q^n$ existuje $\mathbf{c} \in \mathcal{C}$ takové, že $\mathrm{d}(\mathbf{c}, \mathbf{f}) \leq \rho$, a současně existují $\mathbf{f}' \in \mathbb{F}_q^n$ a $\mathbf{c}' \in \mathbf{C}$ splňující $d(\mathbf{c}', \mathbf{f}') = \rho$.

Z minimality ρ plyne, že
$\rho = \max_{\mathbf{f}\in\mathbb{F}_q^n} \mathrm{d}(\mathbf{f}, \mathcal{C}) = \max_{\mathbf{f}\in\mathbb{F}_q^n} \min_{\mathbf{c}\in\mathcal{C}} \mathrm{d}(\mathbf{c}, \mathbf{f})$.

∎

Poloměr pokrytí blokového kódu

Pokračování důkazu Věty 4.3.

Naopak, nechť $\rho = \max_{\mathbf{f}\in\mathbb{F}_q^n} \min_{\mathbf{c}\in\mathcal{C}} \mathrm{d}(\mathbf{c},\mathbf{f}) = \max_{\mathbf{f}\in\mathbb{F}_q^n} \mathrm{d}(\mathbf{f},\mathcal{C})$.

Pak pro všechna $\mathbf{f} \in \mathbb{F}_q^n$ platí $\rho \geq \mathrm{d}(\mathbf{f},\mathcal{C})$ a existují $\mathbf{f}' \in \mathbb{F}_q^n$ a $\mathbf{c}' \in \mathcal{C}$ splňující rovnost $\rho = \mathrm{d}(\mathbf{c}',\mathbf{f}')$ a pro všechna $\mathbf{c} \in \mathcal{C}$ je $\rho \leq \mathrm{d}(\mathbf{c},\mathbf{f}')$.

Poloměr pokrytí blokového kódu

Pokračování důkazu Věty 4.3.

Naopak, nechť $\rho = \max_{\mathbf{f}\in\mathbb{F}_q^n} \min_{\mathbf{c}\in\mathcal{C}} \mathrm{d}(\mathbf{c},\mathbf{f}) = \max_{\mathbf{f}\in\mathbb{F}_q^n} \mathrm{d}(\mathbf{f},\mathcal{C})$.

Pak pro všechna $\mathbf{f} \in \mathbb{F}_q^n$ platí $\rho \geq \mathrm{d}(\mathbf{f},\mathcal{C})$ a existují $\mathbf{f}' \in \mathbb{F}_q^n$ a $\mathbf{c}' \in \mathcal{C}$ splňující rovnost $\rho = \mathrm{d}(\mathbf{c}',\mathbf{f}')$ a pro všechna $\mathbf{c} \in \mathcal{C}$ je $\rho \leq \mathrm{d}(\mathbf{c},\mathbf{f}')$.

Předpokládejme, že existuje s, $\rho > s$, takové, že každé $\mathbf{f} \in \mathbb{F}_q^n$ je prvkem množiny $\{x \in \mathbb{F}_q^n \mid \mathrm{d}(\mathbf{c},x) \leq s\}$ pro nějaké $\mathbf{c} \in \mathcal{C}$.

To je ale spor s existencí slov $\mathbf{f}'$ a $\mathbf{c}'$, a tedy ρ je poloměr pokrytí kódu $\mathcal{C}$.

∎

Perfektní kódy

Ideální situace z ekonomického pohledu je najít kód $\mathcal{C}$ nad $V_n(q)$ tak, že pro jisté kladné $t > 0$ jsou všechny prvky z $V_n(q)$ obsaženy v disjunktním sjednocení koulí, jejichž středy jsou navzájem různá kódová slova. Takový kód se pak nazývá ***perfektní***. Z jeho definice je zřejmé, že perfektní kód dokáže pomocí pravidla minimální vzdálenosti opravit až t chyb, a nedokáže opravit $t + 1$ chyb.

Perfektní kódy

Ideální situace z ekonomického pohledu je najít kód $\mathcal{C}$ nad $V_n(q)$ tak, že pro jisté kladné $t > 0$ jsou všechny prvky z $V_n(q)$ obsaženy v disjunktním sjednocení koulí, jejichž středy jsou navzájem různá kódová slova. Takový kód se pak nazývá ***perfektní***. Z jeho definice je zřejmé, že perfektní kód dokáže pomocí pravidla minimální vzdálenosti opravit až t chyb, a nedokáže opravit $t + 1$ chyb.

Je tedy nutná podmínka pro to, aby (n, M, d)-kód byl perfektní, že d je liché číslo. Celkem tedy je (n, M, d)-kód perfektní právě tehdy, když $M = A_q(n, d)$ a

$$A_q(n, d) \sum_{k=0}^{\frac{d-1}{2}} \binom{n}{k} (q-1)^k = q^n. \tag{4.4}$$

Perfektní kódy

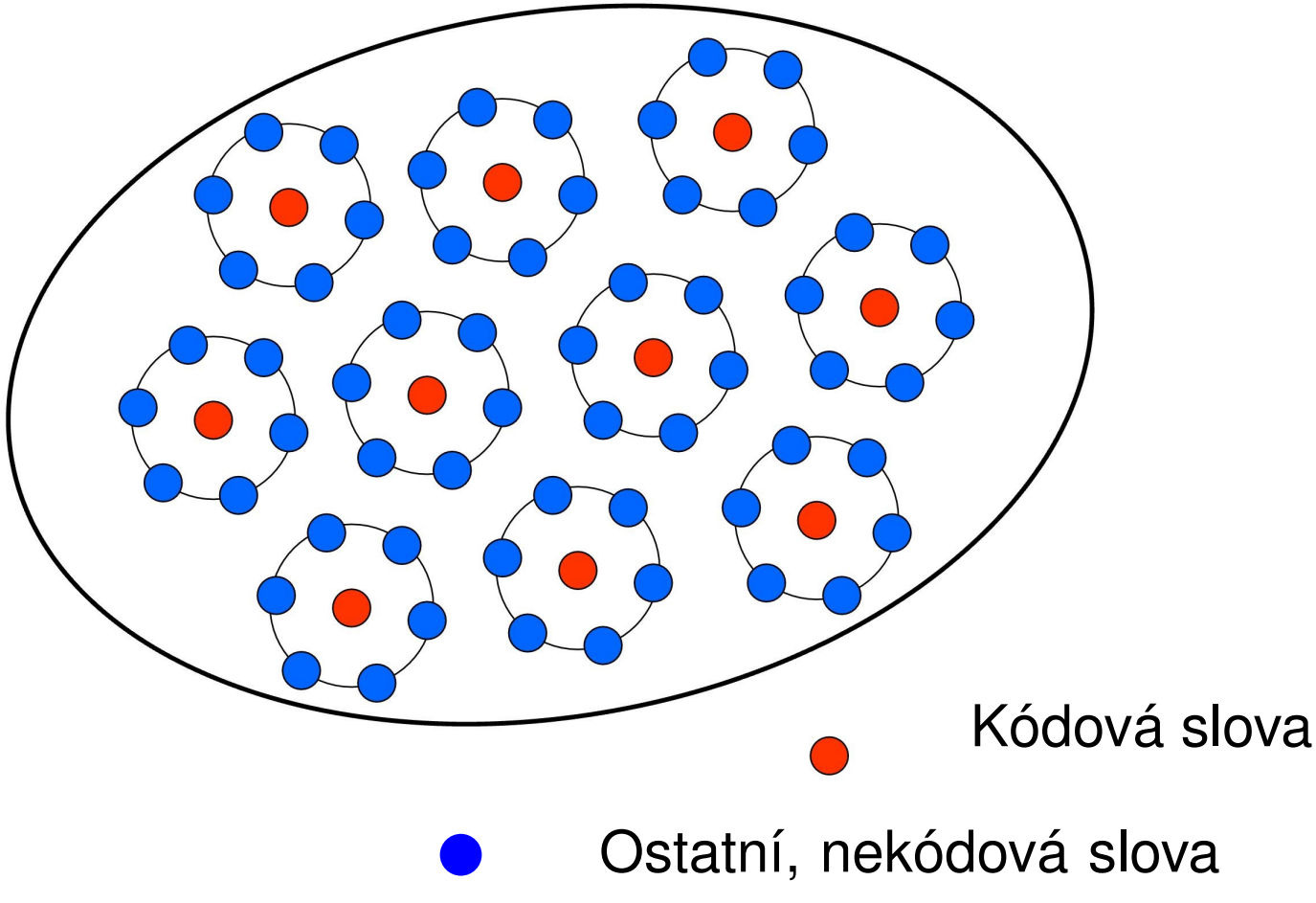
Příklad 4.4

Zřejmým příkladem perfektního kódu je

1. každý kód s právě jedním kódovým slovem,
2. každý binární kód s právě dvěma slovy lichých délek, např. $00\ldots0$ a $11\ldots1$.

Tyto kódy se nazývají ***triviální perfektní kódy***.

Perfektní kódy



Obrázek 4: Perfektní kód s minimální vzdáleností 1.

Další hranice - Singletonova hranice

Věta 4.5

(**Singletonova hranice**) *Platí*

$$A_q(n, d) \leq q^{n-d+1}. \tag{4.5}$$

Další hranice - Singletonova hranice

Věta 4.5

(**Singletonova hranice**) *Platí*

$$A_q(n, d) \leq q^{n-d+1}. \tag{4.5}$$

Důkaz.

Buď $\mathcal{C}$ nějaký (n, M, d)-kód.
Pokud odstraníme posledních $d - 1$ pozic z každého kódového slova z $\mathcal{C}$, musí být nutně výsledná zkrácená slova navzájem různá (jinak by původní slova musela mít vzdálenost $\leq d - 1$).

Další hranice - Singletonova hranice

Věta 4.5

(**Singletonova hranice**) *Platí*

$$A_q(n, d) \leq q^{n-d+1}. \tag{4.5}$$

Důkaz.

Buď $\mathcal{C}$ nějaký (n, M, d)-kód.
Pokud odstraníme posledních $d-1$ pozic z každého kódového slova z $\mathcal{C}$, musí být nutně výsledná zkrácená slova navzájem různá (jinak by původní slova musela mít vzdálenost $\leq d-1$).
Ale počet všech slov délky $n-(d-1)$ je právě q^{n-d+1} tj. $A_q(n, d) \leq q^{n-d+1}$. ∎

Další hranice - Plotkinova hranice

Lemma 4.6

Buď M přirozené číslo. Pak funkce $f : \{0, \ldots, M\} \to \mathbf{N}$ definovaná jako $f(k) = k(M - k)$ nabývá svého maxima pro

$$\overline{k} = \begin{cases} \frac{M}{2} & \text{pokud } M \text{ je sudé,} \\ \frac{M \pm 1}{2} & \text{pokud } M \text{ je liché,} \end{cases} \quad \text{a } f(\overline{k}) = \begin{cases} \frac{M^2}{4} & \text{pokud } M \text{ je sudé,} \\ \frac{M^2 - 1}{4} & \text{pokud } M \text{ je liché.} \end{cases}$$

Důkaz.

Důkaz okamžitě plyne ze vztahu $\sqrt{a \cdot b} \leq \frac{1}{2}(a + b)$ a z průběhu funkce f. ∎

Další hranice - Plotkinova hranice

Lemma 4.6

Buď M přirozené číslo. Pak funkce $f : \{0, \ldots, M\} \to \mathbf{N}$ definovaná jako $f(k) = k(M - k)$ nabývá svého maxima pro

$$\overline{k} = \begin{cases} \frac{M}{2} & \text{pokud } M \text{ je sudé,} \\ \frac{M \pm 1}{2} & \text{pokud } M \text{ je liché,} \end{cases} \quad \text{a } f(\overline{k}) = \begin{cases} \frac{M^2}{4} & \text{pokud } M \text{ je sudé,} \\ \frac{M^2 - 1}{4} & \text{pokud } M \text{ je liché.} \end{cases}$$

Důkaz.

Důkaz okamžitě plyne ze vztahu $\sqrt{a \cdot b} \leq \frac{1}{2}(a + b)$ a z průběhu funkce f. ∎

Věta 4.7

(**Plotkinova hranice**) *Je-li $n < 2d$, máme*

$$A_2(n, d) \leq 2\lfloor \frac{d}{2d - n} \rfloor. \tag{4.6}$$

Další hranice - Plotkinova hranice

Důkaz.

MA Buď $\mathcal{C} = \{\mathbf{c}_1, \ldots, \mathbf{c}_M\}$ nějaký (n, M, d)-kód. Uvažme součet $S = \sum_{i<j} \mathrm{d}(\mathbf{c}_i, \mathbf{c}_j)$. To není nic jiného, než součet všech vzdáleností kódových slov z $\mathcal{C}$. Protože ale $d \leq \mathrm{d}(\mathbf{c}_i, \mathbf{c}_j)$ pro všechna i, j a máme právě $\binom{M}{2}$ dvojic kódových slov z $\mathcal{C}$, platí

$$S = \sum_{i<j} \mathrm{d}(\mathbf{c}_i, \mathbf{c}_j) \geq d\binom{M}{2}. \tag{4.7}$$

Další hranice - Plotkinova hranice

Důkaz.

MA Buď $\mathcal{C} = \{\mathbf{c}_1, \ldots, \mathbf{c}_M\}$ nějaký (n, M, d)-kód. Uvažme součet $S = \sum_{i<j} \mathrm{d}(\mathbf{c}_i, \mathbf{c}_j)$. To není nic jiného, než součet všech vzdáleností kódových slov z $\mathcal{C}$. Protože ale $d \leq \mathrm{d}(\mathbf{c}_i, \mathbf{c}_j)$ pro všechna i, j a máme právě $\binom{M}{2}$ dvojic kódových slov z $\mathcal{C}$, platí

$$S = \sum_{i<j} \mathrm{d}(\mathbf{c}_i, \mathbf{c}_j) \geq d\binom{M}{2}. \tag{4.7}$$

Pokusme se nyní spočítat S tím, že se podíváme na každou pozici zvlášť. Uvažme tedy kódová slova ve tvaru

$$\begin{aligned} \mathbf{c}_1 &= c_{11}\, c_{12} \ldots c_{1n} \\ \mathbf{c}_2 &= c_{21}\, c_{22} \ldots c_{2n} \\ &\vdots \\ \mathbf{c}_M &= c_{M1}\, c_{M2} \ldots c_{Mn}. \end{aligned}$$

Další hranice - Plotkinova hranice

Důkaz.

Máme pak, pro všechna $j, 1 \leq j \leq n$, že pokud k_j bitů slova $c_{1j} \ldots c_{Mj}$ je rovno 1 a zbývajících $M - k_j$ bitů je nulových, pak tyto bity přispějí k součtu všech vzdáleností číslem $f(k_j) = k_j(M - k_j)$. Máme tedy celkem (protože všech sloupců je n) $S \leq nf(\overline{k})$. Zejména tedy platí

$$S \leq nf(\overline{k}) = \begin{cases} n\frac{M^2}{4} & \text{pokud } M \text{ je sudé} \\ n\frac{M^2-1}{4} & \text{pokud } M \text{ je liché.} \end{cases}$$

Dáme-li obě nerovnosti dohromady, máme

$$\binom{M}{2} \cdot d \leq \begin{cases} n\frac{M^2}{4} & \text{pokud } M \text{ je sudé} \\ n\frac{M^2-1}{4} & \text{pokud } M \text{ je liché.} \end{cases}$$

Další hranice - Plotkinova hranice

Důkaz.

Po jednoduché úpravě pak obdržíme

$$M \leq \begin{cases} \frac{2d}{2d-n} & \text{pokud } M \text{ je sudé} \\ \frac{n}{2d-n} < \frac{2d}{2d-n} & \text{pokud } M \text{ je liché.} \end{cases}$$

položme $a = \frac{d}{2d-n}$. Pak máme

$$M \leq \begin{cases} \lfloor 2a \rfloor & \text{pokud } M \text{ je sudé} \\ \lfloor 2a \rfloor - 1 & \text{pokud } M \text{ je liché.} \end{cases}$$

Předpokládejme nejprve, že $k \leq a < k + \frac{1}{2}$ pro nějaké přirozené číslo k. Pak $\lfloor 2a \rfloor = 2k$ a $2\lfloor a \rfloor = \lfloor 2a \rfloor$. Máme tedy, nezávisle na paritě M, že $M \leq 2\lfloor a \rfloor$.

∎

Další hranice - Plotkinova hranice

Důkaz.

Předpokládejme nyní, že $k + \frac{1}{2} \leq a < k + 1$ pro nějaké přirozené číslo k. Pak $\lfloor 2a \rfloor = 2k + 1$ a $2\lfloor a \rfloor = 2k$.

Je-li M liché, máme $M \leq \lfloor 2a \rfloor - 1 = 2k = 2\lfloor a \rfloor$ a, je-li M sudé, máme $M \leq \lfloor 2a \rfloor = 2k + 1$ tj. $M \leq 2k = 2\lfloor a \rfloor$.

Nutně tedy celkem $M \leq 2\lfloor \frac{d}{2d-n} \rfloor$. ∎

Další hranice - Plotkinova hranice

FI

Lemma 4.8

Buď k přirozené číslo. Pak $A_2(4k-1, 2k-1) \leq 8k$ a $A_2(4k, 2k) \leq 8k$.

Další hranice - Plotkinova hranice

FI

Lemma 4.8

Buď k přirozené číslo. Pak $A_2(4k-1, 2k-1) \leq 8k$ a $A_2(4k, 2k) \leq 8k$.

Důkaz.

Ověřme nejprve, že $A_2(4k, 2k) \leq 8k$. Víme ale, že $A_2(4k, 2k) \leq 2A_2(4k-1, 2k) \leq 4\lfloor \frac{2k}{1} \rfloor = 8k$ dle 4.7. Protože dále platí $A_2(4k-1, 2k-1) = A_2(4k, 2k) \leq 8k$, tvrzení je dokázáno. ∎

Hranice $A_q(n, d)$

Cvičení 4.9

1. *Ukažte, že* $19 \leq A_2(10, 3) \leq 93$.
2. *Ukažte, že pro všechna přirozená čísla* $q > 1$, *parametry* $n = (q^r - 1)/(q - 1)$, $M = q^{n-r}$, $d = 3$, *kde* r *je nějaké přirozené číslo* ≥ 2, *splňují podmínku (4.4) proto, aby se jednalo o perfektní kód.*
   Poznamenejme, že ačkoliv tyto parametry splňují (4.4) pro každé přirozené číslo q, byla vyslovena hypotéza, že příslušné perfektní kódy existují právě tehdy, když je q mocnina prvočísla.

Obsah

Lineární kódy

FI Předpokládejme, že $\mathcal{C}$ je kód s minimální vzdáleností $d = 2e + 1$ a lze tedy pomocí metody nejbližšího kódového slova opravit až e chyb.
Má-li kód $\mathcal{C}$ málo prvků, jedná se o velmi praktickou metodu. V případě, že číslo $|\mathcal{C}|$ bude velké, bude tato metoda opravdu *velmi časově náročná*, protože musíme srovnávat přijatý vektor $\mathbf{y}$ s velkým množstvím kódových slov.

Lineární kódy

FI Předpokládejme, že $\mathcal{C}$ je kód s minimální vzdáleností $d = 2e + 1$ a lze tedy pomocí metody nejbližšího kódového slova opravit až e chyb.
Má-li kód $\mathcal{C}$ málo prvků, jedná se o velmi praktickou metodu. V případě, že číslo $|\mathcal{C}|$ bude velké, bude tato metoda opravdu *velmi časově náročná*, protože musíme srovnávat přijatý vektor $\mathbf{y}$ s velkým množstvím kódových slov.

To je důvod pro studium více strukturovaných kódů, jako jsou např. lineární kódy.
Předpokládejme tedy, že počet prvků q naší abecedy je prvočíselná mocnina p^m.
Můžeme tedy považovat Σ za těleso $\mathbb{F}_q$ o q-prvcích.

Lineární kódy

Buď dále $V_n(q)$ vektorový prostor dimenze n nad tělesem F_q. Typický prvek tohoto vektorového prostoru budeme psát jakožto $\mathbf{x} = (x_1, \ldots, x_n)$, občas pak zkráceně jakožto $\mathbf{x} = x_1 \ldots x_n$, kde $x_i \in \mathbb{F}_q$.

Lineární kódy

Buď dále $V_n(q)$ vektorový prostor dimenze n nad tělesem F_q. Typický prvek tohoto vektorového prostoru budeme psát jakožto $\mathbf{x} = (x_1, \ldots, x_n)$, občas pak zkráceně jakožto $\mathbf{x} = x_1 \ldots x_n$, kde $x_i \in \mathbb{F}_q$.

Lineární kód $\mathcal{C}$ nad Σ je definován jakožto podprostor prostoru $V_n(q)$. Má-li tento podprostor dimenzi k, mluvíme o $[n, k]$-kódu nebo, chceme-li specifikovat minimální vzdálenost, mluvíme o $[n, k, d]$-kódu.

Lineární kódy

Buď dále $V_n(q)$ vektorový prostor dimenze n nad tělesem F_q. Typický prvek tohoto vektorového prostoru budeme psát jakožto $\mathbf{x} = (x_1, \ldots, x_n)$, občas pak zkráceně jakožto $\mathbf{x} = x_1 \ldots x_n$, kde $x_i \in \mathbb{F}_q$.

Lineární kód $\mathcal{C}$ nad Σ je definován jakožto podprostor prostoru $V_n(q)$. Má-li tento podprostor dimenzi k, mluvíme o $[n, k]$-kódu nebo, chceme-li specifikovat minimální vzdálenost, mluvíme o $[n, k, d]$-kódu.
Protože každý k-dimenzionální podprostor nad $\mathbb{F}_q$ má q^k prvků, máme:

Každý $[n, k, d]$-kód nad $\mathbb{F}_q$ je (n, q^k, d)-kód.

Lineární kódy

Výhoda lineárních kódů je to, že pomocí k kódových slov délky n můžeme zcela popsat kód s právě q^k kódovými slovy délky n. Tím dosáhneme obrovské úspory paměti. Totiž každý podprostor dimenze k je úplně popsán k lineárně nezávislými vektory.

Lineární kódy

Výhoda lineárních kódů je to, že pomocí k kódových slov délky n můžeme zcela popsat kód s právě q^k kódovými slovy délky n. Tím dosáhneme obrovské úspory paměti. Totiž každý podprostor dimenze k je úplně popsán k lineárně nezávislými vektory.

Definujeme pak ***generující matici*** pro lineární $[n, k]$-kód $\mathcal{C}$ jakožto libovolnou matici rozměru $k \times n$, jejíž řádky tvoří k lineárně nezávislých kódových slov z $\mathcal{C}$.

Lineární kódy

Předpokládejme nyní, že G je generující matice kódu $\mathcal{C}$ a G' je matice, kterou můžeme obdržet z G pomocí konečné posloupnosti permutací následujícího typu:

1. záměna řádků,
2. násobení řádku nenulovým skalárem,
3. přičtení k řádku skalární násobek jiného řádku,
4. záměna sloupců,
5. násobení sloupce nenulovým skalárem.

Lineární kódy

Pak lze snadno ukázat následující tvrzení.

Lemma 5.1

G' je generující matice kódu $\mathcal{C}'$, který je ekvivalentní s kódem $\mathcal{C}$.

Důkaz.

Stačí ukázat, že každá z operací (1)-(5) odpovídá vytvoření generující matice G' kódu $\mathcal{C}'$, který je ekvivalentní s kódem $\mathcal{C}$.

Lineární kódy

Pak lze snadno ukázat následující tvrzení.

Lemma 5.1

G' je generující matice kódu $\mathcal{C}'$, který je ekvivalentní s kódem $\mathcal{C}$.

Důkaz.

Stačí ukázat, že každá z operací (1)-(5) odpovídá vytvoření generující matice G' kódu $\mathcal{C}'$, který je ekvivalentní s kódem $\mathcal{C}$. Evidentně, záměna řádků, vynásobení řádku nenulovým skalárem a přičtení řádků jsou operace takové, že dokonce kód $\mathcal{C}'$ je totožný s kódem $\mathcal{C}$.

Lineární kódy

Pak lze snadno ukázat následující tvrzení.

Lemma 5.1

G' je generující matice kódu $\mathcal{C}'$, který je ekvivalentní s kódem $\mathcal{C}$.

Důkaz.

Stačí ukázat, že každá z operací (1)-(5) odpovídá vytvoření generující matice G' kódu $\mathcal{C}'$, který je ekvivalentní s kódem $\mathcal{C}$. Evidentně, záměna řádků, vynásobení řádku nenulovým skalárem a přičtení řádků jsou operace takové, že dokonce kód $\mathcal{C}'$ je totožný s kódem $\mathcal{C}$. Záměna sloupců v matici G znamená, že provedeme poziční permutaci určenou transpozicí sloupců. Vynásobení sloupce nenulovým skalárem je symbolová permutace tohoto sloupce (pracujeme nad tělesem a pak je množina nenulových skalárů grupou). Jsou tedy odpovídající kódy ekvivalentní s kódem $\mathcal{C}$. ∎

Lineární kódy

Lemma 5.2

Buď G matice typu $k \times n$ jejíž řádky jsou lineárně nezávislé; pak, aplikujeme-li posloupnost operací typu (1)-(5) na G, je možné G převést na matici $G' = [E_k, A]$, kde E_k je jednotková matice typu $k \times k$.

Důkaz.

Důkaz je veden indukcí vzhledem ke k. Pokud $k = 1$, je tvrzení evidentní. Lze bez újmy na obecnosti předpokládat, že prvek g_{11} je nenulový (to lze vždy zajistit záměnou sloupců). Stačí pak vynásobit řádek matice G prvkem inverzním k prvku g_{11}. Předpokládejme, že tvrzení platí pro k a chceme jej dokázat pro $k + 1$. Protože hodnost matice G je $k + 1$ existuje v matici G $k + 1$ nezávislých sloupců. ∎

Pokračování důkazu Lemmatu 5.2

Pomocí operace typu (4) tyto sloupce dostaneme na prvních $k + 1$ sloupců nové matice G'.

Pokračování důkazu Lemmatu 5.2

Pomocí operace typu (4) tyto sloupce dostaneme na prvních $k + 1$ sloupců nové matice G'.

Na prvních k nezávislých řádků matice G' lze pak aplikovat indukční předpoklad a obdržíme matici, která má na prvních k řádcích submatici typu $[E_k, A]$.

Pokračování důkazu Lemmatu 5.2

Pomocí operace typu (4) tyto sloupce dostaneme na prvních $k + 1$ sloupců nové matice G'.

Na prvních k nezávislých řádků matice G' lze pak aplikovat indukční předpoklad a obdržíme matici, která má na prvních k řádcích submatici typu $[E_k, A]$.

Pomocí řádkových úprav pak zajistíme, že $(k + 1)$-ní řádek bude mít na prvních k pozicích nulové prvky.

Pokračování důkazu Lemmatu 5.2

Pomocí operace typu (4) tyto sloupce dostaneme na prvních $k + 1$ sloupců nové matice G'.

Na prvních k nezávislých řádků matice G' lze pak aplikovat indukční předpoklad a obdržíme matici, která má na prvních k řádcích submatici typu $[E_k, A]$.

Pomocí řádkových úprav pak zajistíme, že $(k + 1)$-ní řádek bude mít na prvních k pozicích nulové prvky.

Protože všechny naše úpravy zachovávají hodnost, bude v $(k + 1)$-ním sloupci v $(k + 1)$-ním řádku nenulový prvek. Stačí pak vynásobit $(k + 1)$-ní řádek inverzním prvkem k tomuto prvku. ■

Pokračování důkazu Lemmatu 5.2

Pomocí operace typu (4) tyto sloupce dostaneme na prvních $k + 1$ sloupců nové matice G'.

Na prvních k nezávislých řádků matice G' lze pak aplikovat indukční předpoklad a obdržíme matici, která má na prvních k řádcích submatici typu $[E_k, A]$.

Pomocí řádkových úprav pak zajistíme, že $(k + 1)$-ní řádek bude mít na prvních k pozicích nulové prvky.

Protože všechny naše úpravy zachovávají hodnost, bude v $(k + 1)$-ním sloupci v $(k + 1)$-ním řádku nenulový prvek. Stačí pak vynásobit $(k + 1)$-ní řádek inverzním prvkem k tomuto prvku. ■

V důsledku 5.2 můžeme bez újmy na obecnosti pracovat s generujícími maticemi ve výše uvedené standardní formě.

Lineární kódy - minimální vzdálenost

Jinou užitečnou vlastností lineárních kódů je, že jejich minimální vzdálenost lze najít mnohem snáze, než v případu obecných kódů.

Lineární kódy - minimální vzdálenost

Jinou užitečnou vlastností lineárních kódů je, že jejich minimální vzdálenost lze najít mnohem snáze, než v případu obecných kódů. Máme pak následující výsledek

Věta 5.3

Minimální vzdálenost d lineárního kódu $\mathcal{C}$ je minimální váha w všech nenulových vektorů z $\mathcal{C}$.

Lineární kódy - minimální vzdálenost

Jinou užitečnou vlastností lineárních kódů je, že jejich minimální vzdálenost lze najít mnohem snáze, než v případu obecných kódů. Máme pak následující výsledek

Věta 5.3

Minimální vzdálenost d lineárního kódu $\mathcal{C}$ je minimální váha w všech nenulových vektorů z $\mathcal{C}$.

Důkaz.

Buď d minimální vzdálenost lineárního kódu $\mathcal{C}$ a předpokládejme, že $\mathbf{x}, \mathbf{y} \in \mathcal{C}$ tak, že $\mathrm{d}(\mathbf{x}, \mathbf{y}) = d$. Pak $\mathbf{x} - \mathbf{y} \in \mathcal{C}$, $\mathrm{w}(\mathbf{x} - \mathbf{y}) = d \geq w = \mathrm{w}(\mathbf{z}) = \mathrm{d}(\mathbf{z}, \mathbf{0}) \geq d$ pro vhodný vektor $\mathbf{z} \in \mathcal{C}$. ∎

Lineární kódy - použití

FI Předpokládejme, že $\mathcal{C}$ je lineární $[n, k]$-kód nad $\mathbb{F}_q = \Sigma$ a že má generující matice G tvaru

$$G = \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \vdots \\ \mathbf{r}_k \end{bmatrix} = [E_k, A],$$

kde $\mathbf{r}_i$ jsou vektory délky n nad $\mathbb{F}_q$ a A je matice typu $k \times (n - k)$. Kódová slova kódu $\mathcal{C}$ jsou vektory délky n tvaru

$$\sum_{i=1}^{k} a_i \mathbf{r}_i, \quad a_i \in \mathbb{F}_q.$$

Lineární kódy - použití

Základní myšlenka zakódování je následující. Pokud je zpráva braná jako posloupnost $\mathbf{s} = (s_1, \ldots, s_k)$, zakódujeme $\mathbf{s}$ pomocí kódového slova $\mathbf{c} = (c_1, \ldots, c_n)$, kde c_i jsou určena předpisem

$$[c_1, \ldots, c_n] = [s_1, \ldots, s_k]\,[E_k, A]\,, \tag{5.1}$$

tj. $c_i = s_i$ pro $1 \leq i \leq k$.

Příklad 5.4

Předpokládejme, že kód $\mathcal{C}$ nad tělesem F_3 (což je těleso zbytkových tříd po dělení 3) má generující matici G tvaru

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 2 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 2 & 0 & 1 \end{bmatrix}.$$

Lineární kódy - použití

Příklad 5.4 (Pokračování)

Je-li vstupní zpráva ze zdroje tvaru

$$102101210122\ldots$$

rozdělíme ji nejprve do bloků délky tři a obdržíme

$$102 \mid 101 \mid 210 \mid 122 \mid \ldots$$

a pak zakódujeme zdrojová slova jakožto

$$102 \mapsto \mathbf{r}_1 + 2\mathbf{r}_3 = 102222, \qquad 101 \mapsto \mathbf{r}_1 + \mathbf{r}_3 = 101021$$
$$201 \mapsto 2\mathbf{r}_1 + \mathbf{r}_3 = 210221, \qquad 122 \mapsto \mathbf{r}_1 + 2\mathbf{r}_2 + 2\mathbf{r}_3 = 122211.$$

Dostaneme tedy posloupnost

$$102 \mid 222 \mid 101 \mid 021 \mid 210 \mid 221 \mid 122 \mid 211 \mid \ldots$$

Tedy jsme, při zdvojení délky zprávy, zpomalili rychlost přenosu na polovic. Zvýšili jsme ale spolehlivost.

Lineární kódy - použití

Poznamenejme, že vztah (5.1) je ekvivalentní s rovnicí

$$\left[-A^{\top}, E_{n-k}\right] [c_1, \ldots, c_n]^{\top} = \mathbf{0}. \tag{5.2}$$

Lineární kódy - použití

Poznamenejme, že vztah (5.1) je ekvivalentní s rovnicí

$$\left[-A^\top, E_{n-k}\right] [c_1, \ldots, c_n]^\top = \mathbf{0}. \tag{5.2}$$

Matice $H = \left[-A^\top, E_{n-k}\right]$ se nazývá matice ***kontroly parity*** kódu $\mathcal{C}$. Zejména tedy platí, že $\mathbf{z} \in \mathcal{C}$ právě tehdy, když $H\,[z_1, \ldots, z_n]^\top = \mathbf{0}$.

Lineární kódy - použití

Poznamenejme, že vztah (5.1) je ekvivalentní s rovnicí

$$\left[-A^\top, E_{n-k}\right] [c_1, \ldots, c_n]^\top = \mathbf{0}. \tag{5.2}$$

Matice $H = \left[-A^\top, E_{n-k}\right]$ se nazývá matice ***kontroly parity*** kódu $\mathcal{C}$. Zejména tedy platí, že $\mathbf{z} \in \mathcal{C}$ právě tehdy, když $H\,[z_1, \ldots, z_n]^\top = \mathbf{0}$.

Přitom matice H kontroly parity definuje jak vlastní kód $\mathcal{C}$ tak i příslušnou generující matici G. Název matice kontroly parity znamená, že na jistých kontrolních místech jsou přidány jisté kontrolní součty, které zkontrolují naše kódová slova.

Lineární kódy - použití

Občas budeme pro $[n, k]$-kód říkat, že prvních k složek kódového slova je nazýváno ***informačními znaky*** a zbývajících $n - k$ složek jsou ***symboly kontroly parity*** (***kontrolní znaky***).

Příklad 5.5

Určeme nyní matici H kontroly parity z příkladu 5.4. Ta má tvar

$$H = \begin{bmatrix} -1 & 0 & -2 & 1 & 0 & 0 \\ -2 & -1 & 0 & 0 & 1 & 0 \\ 0 & -1 & -1 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 1 & 1 & 0 & 0 \\ 1 & 2 & 0 & 0 & 1 & 0 \\ 0 & 2 & 2 & 0 & 0 & 1 \end{bmatrix}.$$

Lineární kódy - použití

Příklad 5.5 (Pokračování)

Kódové slovo 102222 *sestává ze zprávových čísel* 102 *a symbolů kontroly parity* 222. *Evidentně, rovnost (5.2) je pro toto kódové slovo (a všechna zbývající) splněna.*
Obecně musí tedy kódová slova splnit systém rovnic

$$2c_1 + c_3 + c_4 = 0 \quad c_1 + 2c_2 + c_5 = 0 \quad 2c_2 + 2c_3 + c_6 = 0. \quad (5.3)$$

Základní idea pro metodu opravování chyb je vidět na tomto příkladě. Předpokládejme, že naše obdržené slovo nesplňuje první a třetí rovnici (5.3) v rovnicích kontroly parity. Pak můžeme dedukovat, že chybná číslice zprávy je číslice c_3, *protože to je jediná číslice, která se vyskytuje v obou rovnicích.*

Lineární kódy - použití

Věta 5.6

Je-li H matice kontroly parity kódu $\mathcal{C}$ délky n, pak kód $\mathcal{C}$ má minimální vzdálenost d tehdy a jen tehdy, když každých $d-1$ sloupců matice H je nezávislých, ale některých d sloupců je lineárně závislých.

Důkaz.

Označme po řadě sloupce matice H jako $\mathbf{h}_1, \ldots, \mathbf{h}_n$.
Připomeňme, že pro každý řádkový vektor $[c_1, \ldots, c_n]$ máme

$$[c_1, \ldots, c_n]H^T = \sum_{i=1}^{n} c_i \mathbf{h}_i^T.$$

∎

Pokračování důkazu Věty 5.6

Předpokládejme, že d je minimální počet lineárně závislých sloupců matice H. Pak existují skaláry $c_1, \ldots, c_n$, z nichž je právě d nenulových tak, že

$$[c_1, \ldots, c_n]H^T = \mathbf{0}^T.$$

Pokračování důkazu Věty 5.6

Předpokládejme, že d je minimální počet lineárně závislých sloupců matice H. Pak existují skaláry $c_1, \ldots, c_n$, z nichž je právě d nenulových tak, že

$$[c_1, \ldots, c_n] H^T = \mathbf{0}^T.$$

Ale to neříká nic jiného, že $\mathbf{c} = c_1 \ldots c_n \in \mathcal{C}$.
Protože $\mathrm{wt}(\mathbf{c}) = d$, máme $\mathrm{d}(\mathcal{C}) \leq \mathrm{wt}(\mathbf{c}) = d$.

Pokračování důkazu Věty 5.6

Předpokládejme, že d je minimální počet lineárně závislých sloupců matice H. Pak existují skaláry $c_1, \ldots, c_n$, z nichž je právě d nenulových tak, že

$$[c_1, \ldots, c_n]H^T = \mathbf{0}^T.$$

Ale to neříká nic jiného, že $\mathbf{c} = c_1 \ldots c_n \in \mathcal{C}$.
Protože $\mathrm{wt}(\mathbf{c}) = d$, máme $\mathrm{d}(\mathcal{C}) \leq \mathrm{wt}(\mathbf{c}) = d$.

Obráceně, je-li $\mathbf{c} = c_1 \ldots c_n \in \mathcal{C}$ kódové slovo minimální délky, pak nutně $[c_1, \ldots, c_n]H^T = \mathbf{0}^T$ a tedy je $\mathrm{d}(\mathcal{C})$ sloupců z H odpovídajícím d nenulovým prvkům lineárně závislých. Je tedy $d \leq \mathrm{d}(\mathcal{C})$ tj. $d = \mathrm{d}(\mathcal{C})$. ■

Pravidlo minimální vzdálenosti

FI Uvažme problém dekódování pro lineární kódy. Je-li $\mathcal{C}$ $[n, k]$-kód nad abecedou $\Sigma = \mathbb{F}_q$, pak $\mathcal{C}$ obsahuje q^k kódových slov délky n a počet možných obdržených vektorů je q^n.

Pravidlo minimální vzdálenosti

FI Uvažme problém dekódování pro lineární kódy. Je-li $\mathcal{C}$ $[n, k]$-kód nad abecedou $\Sigma = \mathbb{F}_q$, pak $\mathcal{C}$ obsahuje q^k kódových slov délky n a počet možných obdržených vektorů je q^n.

Prohlížecí tabulka, která by pro každý možný obdržený vektor obsahovala "nejbližší" kódové slovo by zabírala příliš velké množství paměti, dokonce pro malá n a k. Jednou z hlavních výhod používání lineárních kódů je, že existuje elegantní způsob vyhnutí se výše uvedenému problému.

Pravidlo minimální vzdálenosti

FI Uvažme problém dekódování pro lineární kódy. Je-li $\mathcal{C}$ $[n, k]$-kód nad abecedou $\Sigma = \mathbb{F}_q$, pak $\mathcal{C}$ obsahuje q^k kódových slov délky n a počet možných obdržených vektorů je q^n.

Prohlížecí tabulka, která by pro každý možný obdržený vektor obsahovala "nejbližší" kódové slovo by zabírala příliš velké množství paměti, dokonce pro malá n a k. Jednou z hlavních výhod používání lineárních kódů je, že existuje elegantní způsob vyhnutí se výše uvedenému problému.

Tento postup popíšeme pouze v binárním případě. Rozšíření na jiné abecedy mohutnosti $q = p^m$, kde p je prvočíslo je bezprostřední i když technicky náročnější.

Pravidlo minimální vzdálenosti

Předpokládejme, že $\mathcal{C}$ je $[n, k]$-binární kód. Protože je $\mathcal{C}$ podprostor vektorového prostoru V_n binárních vektorů délky n, musí být $\mathcal{C}$ podgrupa aditivní grupy V_n.

Pravidlo minimální vzdálenosti

Předpokládejme, že $\mathcal{C}$ je $[n, k]$-binární kód. Protože je $\mathcal{C}$ podprostor vektorového prostoru V_n binárních vektorů délky n, musí být $\mathcal{C}$ podgrupa aditivní grupy V_n.

Připomeňme, že ***řád*** konečné grupy G je definovaný jako mohutnost $|G|$ nosné množiny G, tj. počet prvků G a ***index*** $[G : S]$ podgrupy $S \subseteq G$ je počet $|G/S|$ různých (levých) tříd rozkladu G podle S.

Pravidlo minimální vzdálenosti

Předpokládejme, že $\mathcal{C}$ je $[n, k]$-binární kód. Protože je $\mathcal{C}$ podprostor vektorového prostoru V_n binárních vektorů délky n, musí být $\mathcal{C}$ podgrupa aditivní grupy V_n.

Připomeňme, že ***řád*** konečné grupy G je definovaný jako mohutnost $|G|$ nosné množiny G, tj. počet prvků G a ***index*** $[G : S]$ podgrupy $S \subseteq G$ je počet $|G/S|$ různých (levých) tříd rozkladu G podle S.

Přitom (levá) třída určená prvkem $a \in G$ má tvar $Sa = \{sa : s \in S\}$. Speciálně je přiřazení $a \mapsto Sa$ surjektivní homomorfismus $G \to G/S$. Přitom dvě třídy Sa, Sb jsou totožné právě tehdy, když $a = s_0 b$ pro některé $s_0 \in S$.

Pravidlo minimální vzdálenosti

Předpokládejme, že $\mathcal{C}$ je $[n, k]$-binární kód. Protože je $\mathcal{C}$ podprostor vektorového prostoru V_n binárních vektorů délky n, musí být $\mathcal{C}$ podgrupa aditivní grupy V_n.

Připomeňme, že ***řád*** konečné grupy G je definovaný jako mohutnost $|G|$ nosné množiny G, tj. počet prvků G a ***index*** $[G : S]$ podgrupy $S \subseteq G$ je počet $|G/S|$ různých (levých) tříd rozkladu G podle S.

Přitom (levá) třída určená prvkem $a \in G$ má tvar $Sa = \{sa : s \in S\}$. Speciálně je přiřazení $a \mapsto Sa$ surjektivní homomorfismus $G \to G/S$. Přitom dvě třídy Sa, Sb jsou totožné právě tehdy, když $a = s_0 b$ pro některé $s_0 \in S$.

Pravé násobení prvkem a je bijekce $S \to Sa$ a proto má každá (levá) třída rozkladu stejný počet prvků jako podgrupa S.

Pravidlo minimální vzdálenosti

Protože G je sjednocení příslušných levých tříd rozkladu, je počet prvků celé grupy G stejný jako počet tříd rozkladu krát počet prvků v S:

$$|G| = |G/S| \cdot |S|.$$

Pravidlo minimální vzdálenosti

Protože G je sjednocení příslušných levých tříd rozkladu, je počet prvků celé grupy G stejný jako počet tříd rozkladu krát počet prvků v S:

$$|G| = |G/S| \cdot |S|.$$

Zejména tedy kód $\mathcal{C}$ určuje soubor levých tříd (***kosetů***) podprostoru V_n a libovolný vektor $\mathbf{a} \in V_n$ určuje jediný koset $\mathbf{a} + \mathcal{C} = \{\mathbf{b} : \mathbf{b} = \mathbf{a} + \mathbf{c}$ pro vhodný vektor $\mathbf{c} \in \mathcal{C}\}$.

Pravidlo minimální vzdálenosti

Protože G je sjednocení příslušných levých tříd rozkladu, je počet prvků celé grupy G stejný jako počet tříd rozkladu krát počet prvků v S:

$$|G| = |G/S| \cdot |S|.$$

Zejména tedy kód $\mathcal{C}$ určuje soubor levých tříd (***kosetů***) podprostoru V_n a libovolný vektor $\mathbf{a} \in V_n$ určuje jediný koset $\mathbf{a} + \mathcal{C} = \{\mathbf{b} : \mathbf{b} = \mathbf{a} + \mathbf{c}$ pro vhodný vektor $\mathbf{c} \in \mathcal{C}\}$.

Předpokládejme tedy, že **y** je obdržený vektor v tom případě, že jisté kódové slovo bylo přeneseno kanálem. Řekneme, že vektor **e** je ***možný chybový vektor*** vektoru **y**, pokud existuje kódové slovo $\mathbf{c} \in \mathcal{C}$ tak, že

$$\mathbf{y} - \mathbf{c} = \mathbf{e}.$$

Pravidlo minimální vzdálenosti

Speciálně je tedy interpretace následující: vektor **e** je chybový vektor přidružený k obdrženému vektoru, jestliže je schopen reprezentovat jistou možnou posloupnost chyb při přenosu.

Pravidlo minimální vzdálenosti

Speciálně je tedy interpretace následující: vektor **e** je chybový vektor přidružený k obdrženému vektoru, jestliže je schopen reprezentovat jistou možnou posloupnost chyb při přenosu.

Následující triviální pozorování je klíčové.

Lemma 5.7

Je-li **y** *obdržený vektor, je množina možných chybových vektorů ten koset množiny* $\mathcal{C}$*, který obsahuje vektor* **y**.

Pravidlo minimální vzdálenosti

Speciálně je tedy interpretace následující: vektor **e** je chybový vektor přidružený k obdrženému vektoru, jestliže je schopen reprezentovat jistou možnou posloupnost chyb při přenosu.

Následující triviální pozorování je klíčové.

Lemma 5.7

Je-li **y** *obdržený vektor, je množina možných chybových vektorů ten koset množiny* $\mathcal{C}$*, který obsahuje vektor* **y**.

Důkaz.

Bylo-li obdrženo slovo **y**, je vektor **e** chybový vektor právě tehdy, když existuje kódové slovo $\mathbf{c} \in \mathcal{C}$ tak, že $\mathbf{y} - \mathbf{c} = \mathbf{e}$. Ale $\mathcal{C}$ je podprostor; tedy i $-\mathbf{c} \in \mathcal{C}$, tj. $\mathbf{e} = \mathbf{y} + \mathbf{c}'$ a $\mathbf{e} \in \mathbf{y} + \mathcal{C}$. ∎

Pravidlo minimální vzdálenosti - dekódování

Co to je dekódování podle pravidla minimální vzdálenosti? To není nic jiného, než nalezení chybového vektoru s minimální vahou.

Pravidlo minimální vzdálenosti - dekódování

Co to je dekódování podle pravidla minimální vzdálenosti? To není nic jiného, než nalezení chybového vektoru s minimální vahou.

Známe-li tedy pro každý koset jeho prvek minimální váhy, pak máme základ pro dekódování podle pravidla minimální vzdálenosti. Řekneme pak, že vektor **e** je ***reprezentant*** kosetu $\mathcal{H}$, jestliže má nejmenší váhu ze všech vektorů obsažených v $\mathcal{H} = \mathbf{e} + \mathcal{C}$.

Pravidlo minimální vzdálenosti - dekódování

Co to je dekódování podle pravidla minimální vzdálenosti? To není nic jiného, než nalezení chybového vektoru s minimální vahou.

Známe-li tedy pro každý koset jeho prvek minimální váhy, pak máme základ pro dekódování podle pravidla minimální vzdálenosti. Řekneme pak, že vektor **e** je ***reprezentant*** kosetu $\mathcal{H}$, jestliže má nejmenší váhu ze všech vektorů obsažených v $\mathcal{H} = \mathbf{e} + \mathcal{C}$.

Zdůrazněme, že takovýto reprezentant nemusí být vybrán jednoznačně.

Algoritmus I - dekódování

Krok 1: Po přijetí vektoru $\mathbf{y}$ najdeme reprezentanta $\mathbf{z}_0$ kosetu $\mathbf{y} - \mathcal{C}$.

Krok 2: Vektor $\mathbf{y}$ pak dekódujeme jakožto kódové slovo $\mathbf{y} - \mathbf{z}_0$.

Evidentně, Krok 1 může být velmi časově náročný. Urychlíme jej použitím následující vlastností lineárních kódů.

Lemma 5.8

Dva vektory $\mathbf{y}_1$ a $\mathbf{y}_2$ leží v témže kosetu právě tehdy, když

$$H\mathbf{y}_1^\top = H\mathbf{y}_2^\top.$$

Algoritmus I - dekódování

Důkaz.

Dva vektory $\mathbf{y}_1$ a $\mathbf{y}_2$ leží v témže kosetu právě tehdy, když existuje kódové slovo $\mathbf{c} \in \mathcal{C}$ tak, že

$$\mathbf{y}_1 = \mathbf{y}_2 + \mathbf{c}$$

Algoritmus I - dekódování

Důkaz.

Dva vektory $\mathbf{y}_1$ a $\mathbf{y}_2$ leží v témže kosetu právě tehdy, když existuje kódové slovo $\mathbf{c} \in \mathcal{C}$ tak, že

$$\mathbf{y}_1 = \mathbf{y}_2 + \mathbf{c}$$

tj.

$$H\mathbf{c}^\top = H(\mathbf{y}_1 - \mathbf{y}_2)^\top = \mathbf{0}$$

Algoritmus I - dekódování

Důkaz.

Dva vektory $\mathbf{y}_1$ a $\mathbf{y}_2$ leží v témže kosetu právě tehdy, když existuje kódové slovo $\mathbf{c} \in \mathcal{C}$ tak, že

$$\mathbf{y}_1 = \mathbf{y}_2 + \mathbf{c}$$

tj.

$$H\mathbf{c}^\top = H(\mathbf{y}_1 - \mathbf{y}_2)^\top = \mathbf{0}$$

tj.

$$H\mathbf{y}_1^\top = H\mathbf{y}_2^\top.$$

∎

Algoritmus I - dekódování

Důkaz.

Dva vektory $\mathbf{y}_1$ a $\mathbf{y}_2$ leží v témže kosetu právě tehdy, když existuje kódové slovo $\mathbf{c} \in \mathcal{C}$ tak, že

$$\mathbf{y}_1 = \mathbf{y}_2 + \mathbf{c}$$

tj.

$$H\mathbf{c}^\top = H(\mathbf{y}_1 - \mathbf{y}_2)^\top = \mathbf{0}$$

tj.

$$H\mathbf{y}_1^\top = H\mathbf{y}_2^\top.$$

∎

Definujme ***syndrom*** kosetu $\mathcal{H} = \mathbf{a} + \mathcal{C}$ jakožto vektor $H\mathbf{a}^\top$ délky k. Evidentně, je tato definice korektní. Máme tedy pro každý koset $\mathcal{H}$ jeho syndrom a jeho reprezentanta kosetu.

Algoritmus II – poloeficientní - dekódování

Máme-li tedy předem vypočtenou ***tabulku***, ve které je pro každý syndrom určen příslušný reprezentant, můžeme urychlit výše uvedný algoritmus následovně:

Algoritmus II – poloeficientní - dekódování

Máme-li tedy předem vypočtenou ***tabulku***, ve které je pro každý syndrom určen příslušný reprezentant, můžeme urychlit výše uvedný algoritmus následovně:

Krok 1a: Po přijetí vektoru $\mathbf{y}$ najdeme syndrom $H\mathbf{y}^\top$.

Krok 1b: Z výše uvedené tabulky najdeme odpovídajícího reprezentanta $\mathbf{z}_0$ kosetu $\mathbf{y} - \mathcal{C}$.

Krok 2: Vektor $\mathbf{y}$ pak dekódujeme jakožto kódové slovo $\mathbf{y} - \mathbf{z}_0$.

Algoritmus II – poloeficientní - dekódování

Věta 5.9

Algoritmus II pracuje jakožto dekódovací pravidlo podle minimální vzdálenosti pro lineární kód $\mathcal{C}$.

Důkaz.

Poznamenejme nejprve, že každé obdržené slovo $\mathbf{y}$ můžeme dekódovat jakožto kódové slovo. To je z toho důvodu, že $\mathbf{y}$ a $\mathbf{z}_0$ jsou ve stejném kosetu a tedy $\mathbf{y} - \mathbf{z}_0 \in \mathcal{C}$.

Algoritmus II – poloeficientní - dekódování

Věta 5.9

Algoritmus II pracuje jakožto dekódovací pravidlo podle minimální vzdálenosti pro lineární kód $\mathcal{C}$.

Důkaz.

Poznamenejme nejprve, že každé obdržené slovo $\mathbf{y}$ můžeme dekódovat jakožto kódové slovo. To je z toho důvodu, že $\mathbf{y}$ a $\mathbf{z}_0$ jsou ve stejném kosetu a tedy $\mathbf{y} - \mathbf{z}_0 \in \mathcal{C}$. Předpokládejme, že existuje kódové slovo $\mathbf{c}$ tak, že

$$\mathrm{d}(\mathbf{y}, \mathbf{y} - \mathbf{z}_0) > \mathrm{d}(\mathbf{y}, \mathbf{c}).$$

Algoritmus II – poloeficientní - dekódování

Věta 5.9

Algoritmus II pracuje jakožto dekódovací pravidlo podle minimální vzdálenosti pro lineární kód $\mathcal{C}$.

Důkaz.

Poznamenejme nejprve, že každé obdržené slovo $\mathbf{y}$ můžeme dekódovat jakožto kódové slovo. To je z toho důvodu, že $\mathbf{y}$ a $\mathbf{z}_0$ jsou ve stejném kosetu a tedy $\mathbf{y} - \mathbf{z}_0 \in \mathcal{C}$. Předpokládejme, že existuje kódové slovo $\mathbf{c}$ tak, že

$$\mathrm{d}(\mathbf{y}, \mathbf{y} - \mathbf{z}_0) > \mathrm{d}(\mathbf{y}, \mathbf{c}).$$

To je ekvivalentní s tím, že

$$\mathrm{d}(\mathbf{0}, \mathbf{z}_0) > \mathrm{d}(\mathbf{y} - \mathbf{c}, \mathbf{0}).$$

Algoritmus II – poloeficientní - dekódování

Pokračování důkazu Věty 5.9 .

$$\mathrm{d}(\mathbf{0}, \mathbf{z}_0) > \mathrm{d}(\mathbf{y} - \mathbf{c}, \mathbf{0})$$

právě tehdy, když $\mathrm{w}(\mathbf{z}_0) > \mathrm{w}(\mathbf{y} - \mathbf{c})$.

Algoritmus II – poloeficientní - dekódování

Pokračování důkazu Věty 5.9 .

$$\mathrm{d}(\mathbf{0}, \mathbf{z}_0) > \mathrm{d}(\mathbf{y} - \mathbf{c}, \mathbf{0})$$

právě tehdy, když $\mathrm{w}(\mathbf{z}_0) > \mathrm{w}(\mathbf{y} - \mathbf{c})$.
Ale pak

$$H(\mathbf{y} - \mathbf{c})^\top = H\mathbf{y}^\top - H\mathbf{c}^\top = H\mathbf{y}^\top,$$

protože $\mathbf{c}$ je kódové slovo. Zejména tedy má $\mathbf{y} - \mathbf{c}$ stejný syndrom jako $\mathbf{y}$, leží ve stejném kosetu a a má váhu ostře menší než je váha reprezentanta kosetu $\mathbf{z}_0$, což je spor. ∎

Algoritmus II – poloeficientní - dekódování

V dalším budeme předpokládat, že kódování a dekódování pro lineární $[n, k]$-kód $\mathcal{C} = \{\mathbf{c}_1, \ldots, \mathbf{c}_M\}$, $\mathbf{c}_1 = \mathbf{0}$ při přenosu binárním symetrickým kanálem s pravděpodobností chyby p bude probíhat pomocí následující tabulky

$\mathbf{0}$	$\mathbf{c}_2$	$\mathbf{c}_3$	$\cdots$	$\mathbf{c}_M$
$\mathbf{f}_2$	$\mathbf{f}_2 + \mathbf{c}_2$	$\mathbf{f}_2 + \mathbf{c}_3$	$\cdots$	$\mathbf{f}_2 + \mathbf{c}_M$
$\mathbf{f}_3$	$\mathbf{f}_3 + \mathbf{c}_2$	$\mathbf{f}_3 + \mathbf{c}_3$	$\cdots$	$\mathbf{f}_3 + \mathbf{c}_M$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\mathbf{f}_s$	$\mathbf{f}_s + \mathbf{c}_2$	$\mathbf{f}_s + \mathbf{c}_3$	$\cdots$	$\mathbf{f}_s + \mathbf{c}_M$

Dále předpokládejme, že každý reprezentant $\mathbf{f}_i$ příslušného kosetu má váhu $wt(\mathbf{f}_i)$. Platí pak následující tvrzení. Označme, pro $1 \leq j \leq n$, w_j počet reprezentantů váhy j.

Algoritmus II – poloeficientní - dekódování

Věta 5.10

Buď $\mathcal{C}$ binární lineární $[n, k]$-kód. Pak pravděpodobnost správného dekódování při přenosu binárním symetrickým kanálem je

$$P(\text{správné dekódování}) = \sum_{i=1}^{s} p^{wt(\mathbf{f}_i)}(1-p)^{n-wt(\mathbf{f}_i)}$$

tj.

$$P(\text{správné dekódování}) = \sum_{j=1}^{n} w_j p^j (1-p)^{n-j}.$$

Algoritmus II – poloeficientní - dekódování

Důkaz.

Protože reprezentant $\mathbf{f}_i$ příslušného kosetu má váhu $wt(\mathbf{f}_i)$, pravděpodobnost, že nám vznikne z **c** slovo **d** je stejná, že nám vznikne z **0** příslušný reprezentant $\mathbf{f}_i$ tj.

$$P(\text{reprezentant je } \mathbf{f}_i) = p^{wt(\mathbf{f}_i)}(1-p)^{n-wt(\mathbf{f}_i)}.$$

Posčítáme-li přes všechny reprezentanty, obdržíme požadované tvrzení. ∎

Algoritmus II – poloeficientní - dekódování

Důkaz.

Protože reprezentant $\mathbf{f}_i$ příslušného kosetu má váhu $wt(\mathbf{f}_i)$, pravděpodobnost, že nám vznikne z **c** slovo **d** je stejná, že nám vznikne z **0** příslušný reprezentant $\mathbf{f}_i$ tj.

$$P(\text{reprezentant je } \mathbf{f}_i) = p^{wt(\mathbf{f}_i)}(1-p)^{n-wt(\mathbf{f}_i)}.$$

Posčítáme-li přes všechny reprezentanty, obdržíme požadované tvrzení. ∎

Nevýhody této kódovací metody lze nejlépe vidět na následujícím příkladu.

Algoritmus II – poloeficientní - dekódování

Příklad 5.11

Předpokládejme, že C je binární kód, jehož generující matice G je určena následovně

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

Pak kontrolní matice H má tvar

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

Kódová slova kódu $\mathcal{C}$ jsou

$$0000, 1010, 0111, 1101.$$

Algoritmus II – poloeficientní - dekódování

Příklad 5.11

Příslušná prohlížecí tabulka má tvar

Syndrom $\mathbf{s}$	*Reprezentant* $\mathbf{z}_0$ *kosetu* $\mathcal{H}$, $s = H\mathbf{z}_0^\top$
00	*0000*
10	*0010*
01	*0001*
11	*0100*

.

Předpokládejme tedy, že jsme obdrželi vektor $\mathbf{y} = 1111$. *Je ho syndrom je vektor* $H\mathbf{y}^\top = 10$. *Odpovídající reprezentant je* 0010, *je tedy slovo* 1111 *dekódováno jakožto* 1101.

Poznamenejme, že tato prohlížecí tabulka není určena jednoznačně, například za reprezentanta syndromu 10 *lze vzít vektor* 1000.

Algoritmus II – poloeficientní - dekódování

Příklad 5.11

V případě binárního $[n, k]$-kódu máme právě $|V_n|/|\mathcal{C}| = 2^{n-k}$ (obecně pak q^{n-k}) různých kosetů; zejména tedy bude mít prohlížecí tabulka v Kroku 1(b) právě 2^{n-k} různých položek. Prohledávání takovéto tabulky je pro velká k, n velmi náročné. Avšak ostatní výhody této metody opravňují její široké používání.

Binární Hammingovy kódy

FI Abychom ilustrovali dříve uvedené techniky, uvažme následující příklad. Omezme naši pozornost na binární příklad; buď r nějaké kladné celé číslo a položme $n = 2^r - 1$. Dále definujme kontrolní matici H jakožto matici typu $r \times (2^r - 1)$, jejíž sloupce tvoří všechny navzájem různé nenulové vektory z V_r.

Binární Hammingovy kódy

FI Abychom ilustrovali dříve uvedené techniky, uvažme následující příklad. Omezme naši pozornost na binární příklad; buď r nějaké kladné celé číslo a položme $n = 2^r - 1$. Dále definujme kontrolní matici H jakožto matici typu $r \times (2^r - 1)$, jejíž sloupce tvoří všechny navzájem různé nenulové vektory z V_r. Pak H je kontrolní matice binárního $[n, k]$-kódu, kde

$$n = 2^r - 1, \quad k = n - r.$$

Mluvíme pak o ***Hammingově*** $[n, k]$***-kódu*** .

Binární Hammingovy kódy

FI Abychom ilustrovali dříve uvedené techniky, uvažme následující příklad. Omezme naši pozornost na binární příklad; buď r nějaké kladné celé číslo a položme $n = 2^r - 1$. Dále definujme kontrolní matici H jakožto matici typu $r \times (2^r - 1)$, jejíž sloupce tvoří všechny navzájem různé nenulové vektory z V_r. Pak H je kontrolní matice binárního $[n, k]$-kódu, kde

$$n = 2^r - 1, \quad k = n - r.$$

Mluvíme pak o ***Hammingově*** $[n, k]$***-kódu*** .
Klíčovou vlastnost Hammingových kódů lze zformulovat v následující větě.

Věta 5.12

Každý Hammingův kód je ***perfektní kód*** *opravující jednu chybu.*

Důkaz Věty 5.12

Nejprve ukažmě, že minimální vzdálenost každého Hammingova kódu je alespoň 3.

Důkaz Věty 5.12

Nejprve ukažmě, že minimální vzdálenost každého Hammingova kódu je alespoň 3.
Protože $\mathcal{C}$ je lineární kód, je minimální vzdálenost d($\mathcal{C}$) rovna minimální váze vektorů z $\mathcal{C}$.

Důkaz Věty 5.12

Nejprve ukažmě, že minimální vzdálenost každého Hammingova kódu je alespoň 3.
Protože $\mathcal{C}$ je lineární kód, je minimální vzdálenost d($\mathcal{C}$) rovna minimální váze vektorů z $\mathcal{C}$.
Předpokládejme nejprve, že $\mathcal{C}$ má kódové slovo $\mathbf{u}$ váhy 1 s nenulovým vstupem v i-té souřadnici. Pak platí

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

tj. i-tý sloupec $\mathbf{h}_i$ matice H je nulový, což není z definice matice H možné.

Důkaz Věty 5.12

Nejprve ukažmě, že minimální vzdálenost každého Hammingova kódu je alespoň 3.
Protože $\mathcal{C}$ je lineární kód, je minimální vzdálenost d($\mathcal{C}$) rovna minimální váze vektorů z $\mathcal{C}$.
Předpokládejme nejprve, že $\mathcal{C}$ má kódové slovo **u** váhy 1 s nenulovým vstupem v i-té souřadnici. Pak platí

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

tj. i-tý sloupec $\mathbf{h}_i$ matice H je nulový, což není z definice matice H možné. Předpokládejme dále, že $\mathcal{C}$ má kódové slovo **v** váhy 2 s nenulovými vstupy v i-té a j-té souřadnicích. Pak platí

$$H\mathbf{v}^\top = \mathbf{0},$$

tj.

$$\mathbf{h}_i + \mathbf{h}_j = \mathbf{0}.$$

Důkaz Věty 5.12 - pokračování

Protože pracujeme s binárními kódy, je nutně

$$\mathbf{h}_i = \mathbf{h}_j,$$

což není možné. Je tedy $\mathrm{d}(\mathcal{C}) \geq 3$. Že je to právě 3, plyne okamžitě z Věty 5.6, protože najdeme 3 sloupce matice H, které jsou lineárně závislé (binární reprezentanty čísel 1, 2 a 3).

Důkaz Věty 5.12 - pokračování

Protože pracujeme s binárními kódy, je nutně

$$\mathbf{h}_i = \mathbf{h}_j,$$

což není možné. Je tedy $\mathrm{d}(\mathcal{C}) \geq 3$. Že je to právě 3, plyne okamžitě z Věty 5.6, protože najdeme 3 sloupce matice H, které jsou lineárně závislé (binární reprezentanty čísel 1, 2 a 3). Ukažme, že $\mathcal{C}$ je perfektní. Poznamenejme, že každá 1-koule kolem kódového slova bude obsahovat právě $1 + n = 2^r$ vektorů délky $n = 2^r - 1$.

Důkaz Věty 5.12 - pokračování

Protože pracujeme s binárními kódy, je nutně

$$\mathbf{h}_i = \mathbf{h}_j,$$

což není možné. Je tedy $\mathrm{d}(\mathcal{C}) \geq 3$. Že je to právě 3, plyne okamžitě z Věty 5.6, protože najdeme 3 sloupce matice H, které jsou lineárně závislé (binární reprezentanty čísel 1, 2 a 3). Ukažme, že $\mathcal{C}$ je perfektní. Poznamenejme, že každá 1-koule kolem kódového slova bude obsahovat právě $1 + n = 2^r$ vektorů délky $n = 2^r - 1$.
Protože $\mathcal{C}$ obsahuje právě $2^k = 2^{n-r}$ kódových slov, disjunktní sjednocení těchto 1-koulí je právě celá množina V_n vektorů délky n, jichž je právě $2^n = 2^{n-r} \cdot 2^r$. ■

Binární Hammingovy kódy

Důležitým důsledkem perfektnosti Hammingových kódů je, že

1. Pro Hammingův $[n, k]$-kód jsou reprezentanti kosetů vektory z V_n váhy ≤ 1.

Binární Hammingovy kódy

Důležitým důsledkem perfektnosti Hammingových kódů je, že

1. Pro Hammingův $[n, k]$-kód jsou reprezentanti kosetů vektory z V_n váhy ≤ 1. To vede k následujícímu elegantnímu dekódovacímu algoritmu pro Hammingovy kódu.

Binární Hammingovy kódy

Důležitým důsledkem perfektnosti Hammingových kódů je, že

1. Pro Hammingův $[n, k]$-kód jsou reprezentanti kosetů vektory z V_n váhy ≤ 1. To vede k následujícímu elegantnímu dekódovacímu algoritmu pro Hammingovy kódu. Nejprve poznamenejme:
2. Sloupce matice H lze přemístit tak, že j-tý sloupec matice H je právě binární reprezentace čísla j.
   Je-li obdržen vektor $\mathbf{y}$, spočtěme jeho syndrom $H\mathbf{y}^\perp$ a předpokládejme, že reprezentuje číslo j. Předpokládáme-li pouze jednu chybu, pravidlo minimální vzdálenosti (=pravidlo maximální pravděpodobnosti) nám dává:
   1. Pokud $j = H\mathbf{y}^\perp = \mathbf{0}$, pak nepředpokládáme žádnou chybu a $\mathbf{y}$ je kódové slovo.
   2. Pokud $j = H\mathbf{y}^\perp \neq \mathbf{0}$, pak předpokládáme chybu v j-té pozici a dekódujeme $\mathbf{y}$ jeho změnou v j-té pozici.

Binární Hammingovy kódy

Příklad 5.13

Hammingův $[7, 4]$-kód má matici kontroly parity

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

Předpokládejme, že jsme obdrželi vektor $\mathbf{y} = (1, 0, 1, 0, 1, 1, 0)$. Pak $H\mathbf{y}^\top = (001)$. Tedy za předpokladu, že nenastala více než jedna chyba, předpokládáme, že se chyba vyskytla na prvním místě a dekódujeme pak $\mathbf{y}$ jakožto $\mathbf{y}^*$,

$$\mathbf{y}^* = (0, 0, 1, 0, 1, 1, 0).$$

Binární Hammingovy kódy

Cvičení 5.14

1. *Napište matici kontroly parity binárního* $[15, 11, 3]$*-kódu. Jak bychom dekódovali obdržené vektory:*
   1. (100000000000000),
   2. (111111111111111)*?*

Obsah

Cyklické kódy

FI Diskutujme nyní důležitou skupinu lineárních kódů. Kód $\mathcal{C}$ se nazývá *cyklický*, jestliže platí následující podmínky:

1. $\mathcal{C}$ je lineární,
2. pokud vektor $\mathbf{w} = (w_1, \ldots, w_n) \in \mathcal{C}$, pak i vektor $\mathbf{w}' = (w_n, w_1, \ldots, w_{n-1}) \in \mathcal{C}$.

Cyklické kódy

FI Diskutujme nyní důležitou skupinu lineárních kódů. Kód $\mathcal{C}$ se nazývá *cyklický*, jestliže platí následující podmínky:

1. $\mathcal{C}$ je lineární,
2. pokud vektor $\mathbf{w} = (w_1, \ldots, w_n) \in \mathcal{C}$, pak i vektor $\mathbf{w}' = (w_n, w_1, \ldots, w_{n-1}) \in \mathcal{C}$.

Tyto kódy mají atraktivní algebraické vlastnosti a můžeme je snadno sestrojit pomocí lineárních posouvacích registrů.

Cyklické kódy

FI Diskutujme nyní důležitou skupinu lineárních kódů. Kód $\mathcal{C}$ se nazývá *cyklický*, jestliže platí následující podmínky:

1. $\mathcal{C}$ je lineární,
2. pokud vektor $\mathbf{w} = (w_1, \ldots, w_n) \in \mathcal{C}$, pak i vektor $\mathbf{w}' = (w_n, w_1, \ldots, w_{n-1}) \in \mathcal{C}$.

Tyto kódy mají atraktivní algebraické vlastnosti a můžeme je snadno sestrojit pomocí lineárních posouvacích registrů. Budeme dále pracovat pouze v binárním případě a během tohoto paragrafu budeme identifikovat vektor

$$\mathbf{w} = (w_1, \ldots, w_n)$$

s polynomem

$$w(x) = w_1 + w_2 x + w_3 x^2 + \cdots + w_n x^{n-1}.$$

Cyklické kódy

Dále budeme počítat pouze v okruhu R_n binárních polynomů stupně nejvýše $n-1$ modulo polynom $x^n - 1$. Tedy R_n se skládá z polynomů stupně $\leq n-1$ s koeficienty 0 a 1 tak, že platí následující pravidla pro sčítání a násobení polynomů:

$$\begin{aligned} a(x) + b(x) &= \sum_{i=0}^{n-1} (a_i + b_i) x^i \\ a(x) \cdot b(x) &= a(x)b(x) \bmod (x^n - 1). \end{aligned}$$

Cyklické kódy

Dále budeme počítat pouze v okruhu R_n binárních polynomů stupně nejvýše $n-1$ modulo polynom $x^n - 1$. Tedy R_n se skládá z polynomů stupně $\leq n-1$ s koeficienty 0 a 1 tak, že platí následující pravidla pro sčítání a násobení polynomů:

$$\begin{aligned} a(x) + b(x) &= \sum_{i=0}^{n-1} (a_i + b_i) x^i \\ a(x) \cdot b(x) &= a(x)b(x) \bmod (x^n - 1). \end{aligned}$$

Základním pozorováním je následující skutečnost: posunu v kódovém slově odpovídá násobení odpovídajícího polynomu monomem x v okruhu R_n.

Cyklické kódy

Dále budeme počítat pouze v okruhu R_n binárních polynomů stupně nejvýše $n-1$ modulo polynom x^n-1. Tedy R_n se skládá z polynomů stupně $\leq n-1$ s koeficienty 0 a 1 tak, že platí následující pravidla pro sčítání a násobení polynomů:

$$\begin{aligned} a(x)+b(x) &= \sum_{i=0}^{n-1}(a_i+b_i)x^i \\ a(x)\cdot b(x) &= a(x)b(x) \bmod (x^n-1). \end{aligned}$$

Základním pozorováním je následující skutečnost: posunu v kódovém slově odpovídá násobení odpovídajícího polynomu monomem x v okruhu R_n. Totiž

$$\begin{aligned} & w(x)\cdot x \\ &= w_1x + w_2x^2 + w_3x^3 + \cdots + w_{n-1}x^{n-1} + w_nx^n \bmod (x^n-1) \\ &= w_1x + w_2x^2 + w_3x^3 + \cdots + w_{n-1}x^{n-1} + w_nx^0 \\ &= w_n + w_1x + w_2x^2 + w_3x^3 + \cdots + w_{n-1}x^{n-1}. \end{aligned}$$

Cyklické kódy

Platí pak následující lemma

Lemma 6.1

Je-li $w(x)$ polynomiální reprezentace kódového slova $\mathbf{w} \in \mathcal{C}$, je i $w(x)f(x)$ kódové slovo pro každý polynom f stupně nejvýše $n-1$.

Cyklické kódy

Platí pak následující lemma

Lemma 6.1

Je-li $w(x)$ polynomiální reprezentace kódového slova $\mathbf{w} \in \mathcal{C}$, je i $w(x)f(x)$ kódové slovo pro každý polynom f stupně nejvýše $n-1$.

Důkaz.

Protože $w(x) \in \mathcal{C}$, je i $xw(x) \in \mathcal{C}$ (posunutí o 1 místo doprava). Nutně tedy pro každé přirozené číslo k platí, že $x^k w(x) \in \mathcal{C}$. Ale protože je $\mathcal{C}$ lineární kód, je i libovolná lineární kombinace kódových slov tvaru $x^k w(x)$ opět v $\mathcal{C}$, zejména tedy je polynom $f(x)w(x)$ v $\mathcal{C}$. ∎

Cyklické kódy

Lemma 6.2

Je-li $g(x)$ nenulový polynom minimálního stupně v $\mathcal{C}$, pak $g(x)$ **generuje** *kód $\mathcal{C}$ v tom smyslu, že každé kódové slovo $w(x) \in \mathcal{C}$ je tvaru*

$$w(x) = f(x)g(x)$$

pro vhodný polynom $f(x)$.

Cyklické kódy

Lemma 6.2

Je-li $g(x)$ nenulový polynom minimálního stupně v $\mathcal{C}$, pak $g(x)$ **generuje** *kód $\mathcal{C}$ v tom smyslu, že každé kódové slovo $w(x) \in \mathcal{C}$ je tvaru*

$$w(x) = f(x)g(x)$$

pro vhodný polynom $f(x)$.

Důkaz.

Předpokládejme, že existuje $w(x) \in \mathcal{C}$, které nelze napsat ve výše uvedeném tvaru. Pak lze psát

$$w(x) = q(x)g(x) + r(x),$$

kde $r(x)$ je zbytek po dělení polynomem $g(x)$ tj. jeho stupeň je menší než stupeň polynomu $g(x)$.

Cyklické kódy

Lemma 6.2

Je-li $g(x)$ nenulový polynom minimálního stupně v $\mathcal{C}$, pak $g(x)$ **generuje** *kód $\mathcal{C}$ v tom smyslu, že každé kódové slovo $w(x) \in \mathcal{C}$ je tvaru*

$$w(x) = f(x)g(x)$$

pro vhodný polynom $f(x)$.

Důkaz.

Předpokládejme, že existuje $w(x) \in \mathcal{C}$, které nelze napsat ve výše uvedeném tvaru. Pak lze psát

$$w(x) = q(x)g(x) + r(x),$$

kde $r(x)$ je zbytek po dělení polynomem $g(x)$ tj. jeho stupeň je menší než stupeň polynomu $g(x)$. Ale $q(x)g(x), w(x) \in \mathcal{C}$, tj. $r(x) \in \mathcal{C}$ tj. nutně je $r(x)$ nulový polynom, spor. Tedy je každý polynom z $\mathcal{C}$ násobkem $g(x)$. ∎

Cyklické kódy

Mluvíme pak o polynomu $g(x)$ jakožto o ***generujícím polynomu*** kódu $\mathcal{C}$. Tím pak dostaneme velmi dobrou reprezentaci kódu $\mathcal{C}$. Připomeňme dále, že cyklický kód není nic jiného než ideál v okruhu polynomu a 6.2 plyne bezprostředně z toho, že každý ideál v okruhu polynomů je hlavní ideál.

Cyklické kódy

Mluvíme pak o polynomu $g(x)$ jakožto o ***generujícím polynomu*** kódu $\mathcal{C}$. Tím pak dostaneme velmi dobrou reprezentaci kódu $\mathcal{C}$. Připomeňme dále, že cyklický kód není nic jiného než ideál v okruhu polynomu a 6.2 plyne bezprostředně z toho, že každý ideál v okruhu polynomů je hlavní ideál.

Příklad 6.3

Předpokládejme, že $n = 3$ *a násobení je prováděno modulo polynom* $x^3 - 1$. *Pak kód*

$$\mathcal{C} = \{0, 1 + x, x + x^2, 1 + x^2\}$$

je cyklický a generován polynomem $1 + x$. *Standardní reprezentace kódu* $\mathcal{C}$ *sestává z vektorů*

$$\{(0, 0, 0), (1, 1, 0), (0, 1, 1), (1, 0, 1)\}.$$

Cyklické kódy

Lemma 6.4

Je-li $\mathcal{C}$ cyklický kód délky n s generujícím polynomem $g(x) = g_1 + g_2 x + \cdots + g_k x^{k-1}$, pak jeho generující matice typu $(n-k+1) \times n$ má tvar

$$G = \begin{bmatrix} g_1 & g_2 & g_3 & \dots & g_k & 0 & 0 & \dots & 0 \\ 0 & g_1 & g_2 & \dots & g_{k-1} & g_k & 0 & \dots & 0 \\ 0 & 0 & g_1 & \dots & g_{k-2} & g_{k-1} & g_k & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & \dots & \dots & \dots & g_{k-2} & g_{k-1} & g_k \end{bmatrix}.$$

Důkaz Lemmatu 6.4

Evidentně, řádky matice G jsou lineárně nezávislé. Ukažme, že každé kódové slovo lze reprezentovat pomocí těchto řádků. Je-li tedy $\mathbf{c} = (c_0, c_1, \ldots, c_{n-1})$ kódové slovo, je odpovídající polynom

$$c(x) = c_0 + c_1 x + \cdots + c_{n-1} x^{n-1}$$

tvaru

$$c(x) = g(x) f(x) \ (\mathrm{mod}(x^n - 1))$$

pro jistý polynom f stupně nejvýše $\leq n - 1$. Ale to neznamená nic jiného, než že

$$c(x) = \sum_{i=0}^{n-1} f_i x^i g(x) \ (\mathrm{mod}(x^n - 1)).$$

Důkaz Lemmatu 6.4

Položíme-li $g^{(i)}(x) = x^i g(x)$ a je-li $\mathbf{g}^{(i)}$ odpovídající reprezentace polynomu $g^{(i)}(x)$ ($i+1$-ní řádek matice G), máme

$$\mathbf{c} = \sum_{i=0}^{n-1} f_i \mathbf{g}^{(i)}.$$

■

Cyklické kódy

Lemma 6.5

Je-li g generující polynom cyklického kódu $\mathcal{C}$ délky n, pak g dělí polynom $(x^n - 1)$.

Cyklické kódy

Lemma 6.5

Je-li g generující polynom cyklického kódu $\mathcal{C}$ délky n, pak g dělí polynom $(x^n - 1)$.

Důkaz.

Předpokládejme, že tomu tak není. Můžeme pak psát

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

kde $r(x)$ je nenulový polynom se stupněm menším než je stupeň g. Protože $q(x)g(x) \in \mathcal{C}$ a $r = -qg$ v tomto okruhu, plyne z linearity, že i r je kódové slovo, tj. se nemůže jednat o polynom minimálního stupně a tedy $r = 0$, spor. ∎

Cyklické kódy

Lemma 6.6

Je-li dán polynom p stupně $< n$*, pak množina všech polynomů*

$$\mathcal{C} = \{qp(\mathrm{mod}(x^n - 1)) : q \text{ je polynom stupně } < n\}$$

je cyklický kód délky n.

Cyklické kódy

Lemma 6.6

Je-li dán polynom p stupně $< n$, pak množina všech polynomů

$$\mathcal{C} = \{qp(\mathrm{mod}(x^n - 1)) : q \text{ je polynom stupně } < n\}$$

je cyklický kód délky n.

Důkaz.

Evidentně, $\mathcal{C}$ je lineární kód. Zároveň, je-li $qp \in \mathcal{C}$, je i $xqp \in \mathcal{C}$ tj. $\mathcal{C}$ je cyklický kód. ∎

Cyklické kódy

Lemma 6.7

Je-li g generující polynom stupně k cyklického kódu délky n $\mathcal{C}$, pak polynom p stupně menšího než n je kódové slovo tehdy a jen tehdy, když

$$p(x)h(x) = 0 \ (\mathrm{mod}(x^n - 1)),$$

kde h je polynom stupně $n - k$ splňující $g(x)h(x) = (x^n - 1)$ z Lemmatu 6.5. Polynom h pak nazýváme **kontrolní polynom kódu** $\mathcal{C}$.

Důkaz Lemmatu 6.7

Je-li $c(x)$ kódové slovo, pak dle Lemmatu 6.2 platí $c(x) = f(x)g(x)$ pro vhodný polynom $f(x)$. Tudíž

$$c(x)h(x) = f(x)g(x)h(x) = f(x)(x^n - 1)) = 0 \ (\mathrm{mod}(x^n - 1)).$$

Důkaz Lemmatu 6.7

Je-li $c(x)$ kódové slovo, pak dle Lemmatu 6.2 platí $c(x) = f(x)g(x)$ pro vhodný polynom $f(x)$. Tudíž

$$c(x)h(x) = f(x)g(x)h(x) = f(x)(x^n - 1)) = 0 \ (\mathrm{mod}(x^n - 1)).$$

Obráceně, předpokládejme, že p je nenulový polynom splňující $p(x)h(x) = 0 \ (\mathrm{mod}(x^n - 1))$. Pak p musí být stupně alespoň k. Nechť

$$p(x) = g(x)q(x) + r(x),$$

kde $r(x)$ je polynom se stupněm menším než je stupeň g.

Důkaz Lemmatu 6.7

Je-li $c(x)$ kódové slovo, pak dle Lemmatu 6.2 platí $c(x) = f(x)g(x)$ pro vhodný polynom $f(x)$. Tudíž

$$c(x)h(x) = f(x)g(x)h(x) = f(x)(x^n - 1)) = 0 \ (\mathrm{mod}(x^n - 1)).$$

Obráceně, předpokládejme, že p je nenulový polynom splňující $p(x)h(x) = 0 \ (\mathrm{mod}(x^n - 1))$. Pak p musí být stupně alespoň k. Nechť

$$p(x) = g(x)q(x) + r(x),$$

kde $r(x)$ je polynom se stupněm menším než je stupeň g. Protože $p(x)h(x) = 0 \ (\mathrm{mod}(x^n - 1))$ a tedy $g(x)q(x)h(x) = 0 \ (\mathrm{mod}(x^n - 1))$, musí být i $r(x)h(x) = 0 \ (\mathrm{mod}(x^n - 1))$.

Důkaz Lemmatu 6.7

Je-li $c(x)$ kódové slovo, pak dle Lemmatu 6.2 platí $c(x) = f(x)g(x)$ pro vhodný polynom $f(x)$. Tudíž

$$c(x)h(x) = f(x)g(x)h(x) = f(x)(x^n - 1)) = 0 \ (\mathrm{mod}(x^n - 1)).$$

Obráceně, předpokládejme, že p je nenulový polynom splňující $p(x)h(x) = 0 \ (\mathrm{mod}(x^n - 1))$. Pak p musí být stupně alespoň k. Nechť

$$p(x) = g(x)q(x) + r(x),$$

kde $r(x)$ je polynom se stupněm menším než je stupeň g. Protože $p(x)h(x) = 0 \ (\mathrm{mod}(x^n - 1))$ a tedy $g(x)q(x)h(x) = 0 \ (\mathrm{mod}(x^n - 1))$, musí být i $r(x)h(x) = 0 \ (\mathrm{mod}(x^n - 1))$. Ale stupeň $r(x)h(x)$ je menší než n. Tedy $r(x)h(x) = 0$ tj. i $r(x) = 0$. Je tedy i

$$p(x) = g(x)q(x) \in \mathcal{C}. \ \blacksquare$$

Cyklické kódy

Cvičení 6.8

1. *Ukažte, že existují právě čtyři binární kódy délky 3 a najděte jejich generující polynom.*

Cyklické kódy

Cvičení 6.8

1. *Ukažte, že existují právě čtyři binární kódy délky 3 a najděte jejich generující polynom.*
2. *Dokažte, že Hammingovo kódování s parametry $n = 7$, $k = 4$ a $d = 3$ je cyklický kód s kontrolním polynomem $x^4 + x^2 + x + 1$. Jak vypadá jeho generující polynom?*

Obsah

FI V tomto odstavci se budeme věnovat dalším dvěma příkladům, kdy pomocí klasické moderní algebry byly zkonstruovány a vyvinuty nové třídy kódů.
Kódování $R(1, 5)$ použité v roce 1969 kosmickým korábem Mariner 9 pro přenos fotografií z Marsu je speciálním případem následujících obecných kódů.

FI V tomto odstavci se budeme věnovat dalším dvěma příkladům, kdy pomocí klasické moderní algebry byly zkonstruovány a vyvinuty nové třídy kódů.
Kódování $R(1, 5)$ použité v roce 1969 kosmickým korábem Mariner 9 pro přenos fotografií z Marsu je speciálním případem následujících obecných kódů.
Nejprve si připomeňme některé pojmy z moderní algebry.
Hadamardova matice je matice H typu $n \times n$, jejíž koeficienty jsou buď $+1$ nebo -1 tak, že

$$HH^T = nE_n, \tag{7.1}$$

kde E_n je jednotková matice typu $n \times n$.

Hadamardovy kódy

Jsou-li dále A a B čtvercové matice typu $m \times m$ a $n \times n$, definujeme jejich *Kroneckerův součin* jako matici $A \otimes B$ typu $mn \times mn$

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \dots & a_{1m}B \\ \vdots & \vdots & & \vdots \\ a_{m1}B & a_{m2}B & \dots & a_{mm}B \end{bmatrix}. \qquad (7.2)$$

Přímým výpočtem pak snadno dokážeme:

Lemma 7.1

Jsou-li H_1 a H_2 Hadamardovy matice, je jejich Kroneckerův součin $H_1 \otimes H_2$ Hadamardova matice.

Hadamardovy kódy - Důkaz Lemmatu 7.1

MA Počítejme $G = (H_1 \otimes H_2)(H_1 \otimes H_2)^T$. Pak g_{ij}, $i = \bar{i} + n \cdot (\hat{i} - 1)$, $j = \bar{j} + n \cdot (\hat{j} - 1)$.
Zejména $g_{ij} = \sum_{l=1}^{m} a_{\hat{i}l} a_{\hat{j}l} (\sum_{k=1}^{n} b_{\bar{i}k} b_{\bar{j}k})$.
Pokud $i = j$, je nutně $\hat{i} = \hat{j}$ a $\bar{i} = \bar{j}$ a tedy i $g_{ii} = \sum_{l=1}^{m} a_{\hat{i}l} a_{\hat{i}l} (\sum_{k=1}^{n} b_{\bar{i}k} b_{\bar{i}k})$. Ale $\sum_{k=1}^{n} b_{\bar{i}k} b_{\bar{i}k} = n$ a tedy $g_{ii} = \sum_{l=1}^{m} a_{\hat{i}l} a_{\hat{i}l} n = mn$.
Nechť tedy $i \neq j$. Pak buď $\bar{i} \neq \bar{j}$ nebo $\hat{i} \neq \hat{j}$. Nechť například $\hat{i} \neq \hat{j}$. Pak $g_{ij} = (\sum_{l=1}^{m} a_{\hat{i}l} a_{\hat{j}l})(\sum_{k=1}^{n} b_{\bar{i}k} b_{\bar{j}k}) = 0(\sum_{k=1}^{n} b_{\bar{i}k} b_{\bar{j}k}) = 0$.
Nechť tedy $\bar{i} \neq \bar{j}$. Podobně,

$$g_{ij} = (\sum_{l=1}^{m} a_{\hat{i}l} a_{\hat{j}l})(\sum_{k=1}^{n} b_{\bar{i}k} b_{\bar{j}k}) = (\sum_{l=1}^{m} a_{\hat{i}l} a_{\hat{j}l})0 = 0.$$

Hadamardovy kódy

FI Začneme-li tedy s nejmenší netriviální Hadamardovou maticí

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix},$$

můžeme postupně iterovat Kroneckerovým součinem, abychom obdrželi posloupnost Hadamardových matic s exponenciálně rostoucím typem. Chceme-li pak tuto posloupnost použít pro účely kódování, předpokládejme, že H je Hadamardova matice rozměru $n \times n$, přičemž n je sudé. Definujme pak A jakožto matici typu $2n \times n$

$$A = \begin{bmatrix} H \\ -H \end{bmatrix}.$$

Hadamardovy kódy

Pak definujme M jakožto matici, kterou získáme z matice A tak, že nahradíme každý výskyt -1 v A číslem 0.
Snadno se ověří následující tvrzení

Lemma 7.2

Hadamardovo kódování

1. *Jsou-li* $\mathbf{x}$ *a* $\mathbf{y}$ *dva různé řádky matice M, je pak vzdálenost* $d(\mathbf{x}, \mathbf{y})$ *vektorů* $\mathbf{x}$ *a* $\mathbf{y}$ *rovna číslu* $\frac{n}{2}$ *nebo* n.
2. *Řádky matice M tvoří binární* $(n, 2n, \frac{n}{2})$*-kód.*

Hadamardovy kódy - Důkaz Lemmatu 7.2

Vezmeme-li 2 různé řádky vzniklé z H, nutně je počet míst, kde se řádky liší, roven počtu míst, kde jsou oba řádky stejné, tj. $\mathrm{d}(\mathbf{x}, \mathbf{y}) = \frac{n}{2}$.

Hadamardovy kódy - Důkaz Lemmatu 7.2

Vezmeme-li 2 různé řádky vzniklé z H, nutně je počet míst, kde se řádky liší, roven počtu míst, kde jsou oba řádky stejné, tj. $\mathrm{d}(\mathbf{x}, \mathbf{y}) = \frac{n}{2}$.

Podobně, vezmeme-li 2 různé řádky, z nichž jeden vznikl z H a druhý z $-H$ a zároveň oba z různých vektorů z H, je nutně opět počet míst, kde se řádky liší, roven počtu míst, kde jsou oba řádky stejné, tj. opět $\mathrm{d}(\mathbf{x}, \mathbf{y}) = \frac{n}{2}$.

Hadamardovy kódy - Důkaz Lemmatu 7.2

Vezmeme-li 2 různé řádky vzniklé z H, nutně je počet míst, kde se řádky liší, roven počtu míst, kde jsou oba řádky stejné, tj. $\mathrm{d}(\mathbf{x}, \mathbf{y}) = \frac{n}{2}$.

Podobně, vezmeme-li 2 různé řádky, z nichž jeden vznikl z H a druhý z $-H$ a zároveň oba z různých vektorů z H, je nutně opět počet míst, kde se řádky liší, roven počtu míst, kde jsou oba řádky stejné, tj. opět $\mathrm{d}(\mathbf{x}, \mathbf{y}) = \frac{n}{2}$.

Případ, kdy oba řádky vznikly ze stejného vektoru, nám dává vzdálenost rovnu n. Poslední případ, kdy oba řádky vznikly z $-H$, se převede na první případ. Zbývající část tvrzení je triviální. ■

Hadamardovy kódy

Provedeme-li výše uvedené pětkrát za sebou na matici H_2, obdržíme $n = 32$ a to je přesně kódování použité Marinerem. Kódy získané výše uvedeným postupem se nazývají ***Hadamardovy kódy***.

Reed-Mullerovo kódování

FI Tato prakticky důležitá třída kódů byla objevena I.S. Reedem a D.E. Mullerem v roce 1954. Abychom mohli popsat tyto kódy, budeme nejprve prezentovat jednoduchý způsob zkonstruování nových kódování ze starých původních.

Reed-Mullerovo kódování

Lemma 7.3

Jsou-li dány (n, M_1, d_1) *binární kódování* $\mathcal{C}_1$ *a jiné* (n, M_2, d_2) *binární kódování* $\mathcal{C}_2$*, můžeme pak definovat třetí binární kódování* $\mathcal{C}_3 = \mathcal{C}_1 * \mathcal{C}_2$ *jakožto*

$$\mathcal{C}_3 = \{(\mathbf{u}, \mathbf{u} + \mathbf{v}) : \mathbf{u} \in \mathcal{C}_1, \mathbf{v} \in \mathcal{C}_2\}.$$

Přitom $\mathcal{C}_3$ *je* $(2n, M_1 M_2, d_3)$*-kód, kde*

$$d_3 = min\{2d_1, d_2\}. \tag{7.3}$$

Jsou-li navíc $\mathcal{C}_1$ *a* $\mathcal{C}_2$ *lineární kódy, je i* $\mathcal{C}_3$ *lineární kód.*

Reed-Mullerovo kódování - Důkaz Lemmatu 7.3

Délka kódových slov v $\mathcal{C}_3$ je nutně $2n$ a snadno se ověří, že jich je právě $M_1 M_2$. To plyne z toho, že pokud $(\mathbf{u}_1, \mathbf{u}_1 + \mathbf{v}_1) = (\mathbf{u}_2, \mathbf{u}_2 + \mathbf{v}_2)$ je nutně $\mathbf{u}_1 = \mathbf{u}_2$ a tedy i $\mathbf{v}_1 = \mathbf{v}_2$. Takovýchto uspořádaných dvojic $(\mathbf{u}, \mathbf{v})$ je právě $M_1 M_2$.

Reed-Mullerovo kódování - Důkaz Lemmatu 7.3

Délka kódových slov v $\mathcal{C}_3$ je nutně $2n$ a snadno se ověří, že jich je právě $M_1 M_2$. To plyne z toho, že pokud $(\mathbf{u}_1, \mathbf{u}_1 + \mathbf{v}_1) = (\mathbf{u}_2, \mathbf{u}_2 + \mathbf{v}_2)$ je nutně $\mathbf{u}_1 = \mathbf{u}_2$ a tedy i $\mathbf{v}_1 = \mathbf{v}_2$. Takovýchto uspořádaných dvojic $(\mathbf{u}, \mathbf{v})$ je právě $M_1 M_2$.

Zbývá ověřit, že minimální délka kódování $\mathcal{C}_3$, kterou značíme d_3, je rovna $\min\{2d_1, d_2\}$. To je ale zřejmé.

Reed-Mullerovo kódování - Důkaz Lemmatu 7.3

Délka kódových slov v $\mathcal{C}_3$ je nutně $2n$ a snadno se ověří, že jich je právě $M_1 M_2$. To plyne z toho, že pokud $(\mathbf{u}_1, \mathbf{u}_1 + \mathbf{v}_1) = (\mathbf{u}_2, \mathbf{u}_2 + \mathbf{v}_2)$ je nutně $\mathbf{u}_1 = \mathbf{u}_2$ a tedy i $\mathbf{v}_1 = \mathbf{v}_2$. Takovýchto uspořádaných dvojic $(\mathbf{u}, \mathbf{v})$ je právě $M_1 M_2$.

Zbývá ověřit, že minimální délka kódování $\mathcal{C}_3$, kterou značíme d_3, je rovna $\min\{2d_1, d_2\}$. To je ale zřejmé.

Totiž, je-li $(\mathbf{u}_1, \mathbf{u}_1 + \mathbf{v}_1) \neq (\mathbf{u}_2, \mathbf{u}_2 + \mathbf{v}_2)$, pak mohou nastat následující případy

1. $\mathbf{u}_1 = \mathbf{u}_2$: pak
   $\mathrm{d}(\mathbf{v}_1, \mathbf{v}_2) = \mathrm{d}((\mathbf{u}_1, \mathbf{u}_1 + \mathbf{v}_1), (\mathbf{u}_2, \mathbf{u}_2 + \mathbf{v}_2)) \geq d_2$.

Reed-Mullerovo kódování - Důkaz Lemmatu 7.3

Délka kódových slov v $\mathcal{C}_3$ je nutně $2n$ a snadno se ověří, že jich je právě $M_1 M_2$. To plyne z toho, že pokud $(\mathbf{u}_1, \mathbf{u}_1 + \mathbf{v}_1) = (\mathbf{u}_2, \mathbf{u}_2 + \mathbf{v}_2)$ je nutně $\mathbf{u}_1 = \mathbf{u}_2$ a tedy i $\mathbf{v}_1 = \mathbf{v}_2$. Takovýchto uspořádaných dvojic $(\mathbf{u}, \mathbf{v})$ je právě $M_1 M_2$.

Zbývá ověřit, že minimální délka kódování $\mathcal{C}_3$, kterou značíme d_3, je rovna $\min\{2d_1, d_2\}$. To je ale zřejmé.

Totiž, je-li $(\mathbf{u}_1, \mathbf{u}_1 + \mathbf{v}_1) \neq (\mathbf{u}_2, \mathbf{u}_2 + \mathbf{v}_2)$, pak mohou nastat následující případy

1. $\mathbf{u}_1 = \mathbf{u}_2$: pak
   $\mathrm{d}(\mathbf{v}_1, \mathbf{v}_2) = \mathrm{d}((\mathbf{u}_1, \mathbf{u}_1 + \mathbf{v}_1), (\mathbf{u}_2, \mathbf{u}_2 + \mathbf{v}_2)) \geq d_2$.
2. $\mathbf{v}_1 = \mathbf{v}_2$: pak
   $\mathrm{d}(\mathbf{u}_1, \mathbf{u}_2) + \mathrm{d}(\mathbf{u}_1, \mathbf{u}_2) = \mathrm{d}((\mathbf{u}_1, \mathbf{u}_1 + \mathbf{v}_1), (\mathbf{u}_2, \mathbf{u}_2 + \mathbf{v}_2)) \geq 2d_1$.

Reed-Mullerovo kódování - Důkaz Lemmatu 7.3

3. $\mathbf{u}_1 \neq \mathbf{u}_2$ a $\mathbf{v}_1 \neq \mathbf{v}_2$: pak označíme-li
$I_n = \{i : \pi_i(u_1) \neq \pi_i(u_2)\}$, $I_e = \{i : \pi_i(u_1) = \pi_i(u_2)\}$,
$J_n = \{i : \pi_i(v_1) \neq \pi_i(v_2)\}$, $J_e = \{i : \pi_i(v_1) = \pi_i(v_2)\}$, je
$\mathrm{d}((\mathbf{u}_1, \mathbf{u}_1 + \mathbf{v}_1), (\mathbf{u}_2, \mathbf{u}_2 + \mathbf{v}_2)) = |I_n| + |J_e \cap I_n| + |J_n \cap I_e| =$
$|J_n| + 2|J_e \cap I_n| \geq d_2$.

Přitom v prvním a druhém případě může pro vhodně vybrané dvojice nastat rovnost, tj. tvrzení platí.

Reed-Mullerovo kódování - Důkaz Lemmatu 7.3

3. $\mathbf{u}_1 \neq \mathbf{u}_2$ a $\mathbf{v}_1 \neq \mathbf{v}_2$: pak označíme-li $I_n = \{i : \pi_i(u_1) \neq \pi_i(u_2)\}$, $I_e = \{i : \pi_i(u_1) = \pi_i(u_2)\}$, $J_n = \{i : \pi_i(v_1) \neq \pi_i(v_2)\}$, $J_e = \{i : \pi_i(v_1) = \pi_i(v_2)\}$, je $\mathrm{d}((\mathbf{u}_1, \mathbf{u}_1 + \mathbf{v}_1), (\mathbf{u}_2, \mathbf{u}_2 + \mathbf{v}_2)) = |I_n| + |J_e \cap I_n| + |J_n \cap I_e| = |J_n| + 2|J_e \cap I_n| \geq d_2$.

Přitom v prvním a druhém případě může pro vhodně vybrané dvojice nastat rovnost, tj. tvrzení platí.

Buď $\mathcal{C}_1$ a $\mathcal{C}_2$ lineární kódy. Evidentně, $\mathbf{0}_{2n} = (\mathbf{0}_n, \mathbf{0}_n) \in \mathcal{C}_3$. Nechť $\mathbf{u}_1, \mathbf{u}_2 \in \mathcal{C}_1$ a $\mathbf{v}_1, \mathbf{v}_2 \in \mathcal{C}_2$. Pak $\mathbf{u}_1 + \mathbf{u}_2 \in \mathcal{C}_1$, $\mathbf{v}_1 + \mathbf{v}_2 \in \mathcal{C}_2$ a $(\mathbf{u}_1, \mathbf{u}_1 + \mathbf{v}_1) + (\mathbf{u}_2, \mathbf{u}_2 + \mathbf{v}_2) = (\mathbf{u}_1 + \mathbf{u}_2, (\mathbf{u}_1 + \mathbf{u}_2) + (\mathbf{v}_1 + \mathbf{v}_2)) \in \mathcal{C}_3$.

■

Reed-Mullerovo kódování

Definujme nyní rekurzivně *Reed-Mullerův kód* $\mathcal{C}(r, m)$ předpisem:

Pro všechna nezáporná celá čísla m a r taková, že $0 \leq r \leq m$, definujeme $\mathcal{C}(r, m)$ jakožto kód délky $n = 2^m$ takový, že

$$\mathcal{C}(0, m) = \{\mathbf{0}, \mathbf{1}\}, \tag{7.4}$$

kde $\mathbf{0} = (0, 0, \ldots, 0)$ a $\mathbf{1} = (1, 1, \ldots, 1)$, $\mathcal{C}(m, m)$ je množina všech binárních vektorů délky $n = 2^m$ tj. $\mathcal{C}(m, m) = \mathbf{2}^{2^m}$ a

$$\mathcal{C}(r + 1, m + 1) = \mathcal{C}(r + 1, m) * \mathcal{C}(r, m) \tag{7.5}$$

pro $r < m$.

Reed-Mullerovo kódování

Můžeme pak tyto kódy konstruovat následovně:

$$\begin{array}{lrcl} m = 1 & \mathcal{C}(0,1) & = & \{00, 11\} \\ & \mathcal{C}(1,1) & = & \{00, 01, 10, 11\} \end{array}$$

Reed-Mullerovo kódování

Můžeme pak tyto kódy konstruovat následovně:

$$
\begin{array}{lll}
m = 1 & \mathcal{C}(0,1) = & \{00, 11\} \\
 & \mathcal{C}(1,1) = & \{00, 01, 10, 11\} \\
 & & \\
m = 2 & \mathcal{C}(0,2) = & \{0000, 1111\} \\
 & \mathcal{C}(1,2) = & \mathcal{C}(1,1) * \mathcal{C}(0,1) \\
 & = & \{0000, 0011, 0101, 0110, \\
 & & \quad 1001, 1010, 1100, 1111\} \\
 & \mathcal{C}(2,2) = & \mathcal{C}(1,2) \cup \{1000, 0100, 0010, \\
 & & \quad 0001, 1110, 1101, 1011, 0111\}
\end{array}
$$

atd.

Reed-Mullerovo kódování

Aplikujeme-li nyní lemma 7.3, obdržíme následující větu:

Věta 7.4

Pro všechna nezáporná celá čísla m a r taková, že $0 \leq r \leq m$ je Reed-Mullerův $\mathcal{C}(r, m)$ lineární binární kód charakteristiky (n_r, M_r, d_r), kde

1. $M_r = 2^a$, *kde* $a = \sum_{i=0}^{r} \binom{m}{i} = 1 + \binom{m}{1} + \cdots + \binom{m}{r}$,
2. $n_r = 2^m$,
3. $d_r = 2^{m-r}$.

Důkaz věty 7.4

Důkaz povedeme indukcí vzhledem k definici $\mathcal{C}(r, m)$.
Totiž pro $\mathcal{C}(0, m) = \{\mathbf{0}, \mathbf{1}\}$ je počet slov M_0 roven dvěma a protože zároveň nutně $a = 1$, první část tvrzení pro $\mathcal{C}(0, m)$ platí.

Důkaz věty 7.4

Důkaz povedeme indukcí vzhledem k definici $\mathcal{C}(r, m)$.
Totiž pro $\mathcal{C}(0, m) = \{\mathbf{0}, \mathbf{1}\}$ je počet slov M_0 roven dvěma a protože zároveň nutně $a = 1$, první část tvrzení pro $\mathcal{C}(0, m)$ platí.

Protože délka n_r vektorů je dle definice 2^m, platí druhá část tvrzení. Protože kód obsahuje pouze dva vektory, které se liší na $n_r = 2^m$ místech, je vzdálenost kódu rovna $n_r = 2^{m-0} = d_r$.

Důkaz věty 7.4

Důkaz povedeme indukcí vzhledem k definici $\mathcal{C}(r, m)$.
Totiž pro $\mathcal{C}(0, m) = \{\mathbf{0}, \mathbf{1}\}$ je počet slov M_0 roven dvěma a protože zároveň nutně $a = 1$, první část tvrzení pro $\mathcal{C}(0, m)$ platí.

Protože délka n_r vektorů je dle definice 2^m, platí druhá část tvrzení. Protože kód obsahuje pouze dva vektory, které se liší na $n_r = 2^m$ místech, je vzdálenost kódu rovna $n_r = 2^{m-0} = d_r$.

Uvažme nyní kód $\mathcal{C}(m, m)$, což je množina všech binárních vektorů délky $n = 2^m = n_m$. Je tedy v našem případě $a = n$ a tedy i $M_m = 2^a$.Vzdálenost kódu je pak nutně $1 = 2^{m-m}$.

Důkaz věty 7.4 - pokračování

Věnujme se nyní případu $\mathcal{C}(r+1, m+1) = \mathcal{C}(r+1, m) * \mathcal{C}(r, m)$. Z indukčního předpokladu a lemmatu 7.3 víme, že $\mathcal{C}(r+1, m)$ a $\mathcal{C}(r, m)$ jsou lineární, tedy i $\mathcal{C}(r+1, m+1)$ je lineární, a

$$M_{r+1} = 2^{\sum_{i=0}^{r+1} \binom{m}{i}} \cdot 2^{\sum_{i=0}^{r} \binom{m}{i}} = 2^{\sum_{i=0}^{r+1} \binom{m}{i} + \sum_{i=0}^{r} \binom{m}{i}}$$

$$= 2^{\sum_{i=0}^{r} \left(\binom{m}{i+1} + \binom{m}{i} \right) + \binom{m}{0}} = 2^{\sum_{i=0}^{r+1} \binom{m+1}{i}}.$$

Důkaz věty 7.4 - pokračování

Věnujme se nyní případu $\mathcal{C}(r+1, m+1) = \mathcal{C}(r+1, m) * \mathcal{C}(r, m)$. Z indukčního předpokladu a lemmatu 7.3 víme, že $\mathcal{C}(r+1, m)$ a $\mathcal{C}(r, m)$ jsou lineární, tedy i $\mathcal{C}(r+1, m+1)$ je lineární, a

$$M_{r+1} = 2^{\sum_{i=0}^{r+1} \binom{m}{i}} \cdot 2^{\sum_{i=0}^{r} \binom{m}{i}} = 2^{\sum_{i=0}^{r+1} \binom{m}{i} + \sum_{i=0}^{r} \binom{m}{i}}$$

$$= 2^{\sum_{i=0}^{r} \left(\binom{m}{i+1} + \binom{m}{i} \right) + \binom{m}{0}} = 2^{\sum_{i=0}^{r+1} \binom{m+1}{i}}.$$

Dále víme, že $n_{r+1} = 2 \cdot 2^m = 2^{m+1}$ a $d_{r+1} = \min\{2 \cdot 2^{m-r-1}, 2^{m-r}\} = 2^{m-r} = 2^{m+1-(r+1)}$. ■

Reed-Mullerovo kódování s délkou 8

Generující matice:

$$\mathcal{C}(0,3): \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \qquad \mathcal{C}(1,3): \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\mathcal{C}(2,3): \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \qquad \mathcal{C}(3,3): \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Obsah

Problémy k řešení

Problémy 1

1. *Ukažte, že pokud platí*
   1. $2^k \sum_{i=0}^{d-2} \binom{n-1}{i} < 2^n,$

   pak existuje binární lineární $[n, k]$*-kód s minimální vzdáleností alespoň* d*. Tudíž odvoďte, že*
   1. $2^k \leq A_2(n, d),$

   kde k *je největší přirozené číslo splnující nerovnost (1).*
   Návod: Zkonstruujte matici H *typu* $(n - k) \times n$*, takovou, že každých jejích* $d - 1$ *sloupců je lineárně nezávislých* $d - 2$*.*

Problémy k řešení

Problémy 1

1. *Ukažte, že pokud platí*
   1. $2^k \sum_{i=0}^{d-2} \binom{n-1}{i} < 2^n,$

   pak existuje binární lineární $[n, k]$-kód s minimální vzdáleností alespoň d. Tudíž odvoďte, že
   1. $2^k \leq A_2(n, d),$

   kde k je největší přirozené číslo splnující nerovnost (1).
   Návod: Zkonstruujte matici H typu $(n - k) \times n$, takovou, že každých jejích $d - 1$ sloupců je lineárně nezávislých $d - 2$. Abyste tohoto dosáhli, vybírejte postupně vektory z $\mathbf{2}^{n-k}$, které se stanou sloupce matice H. Přitom vybírejte postupně sloupce tak, že každý nový sloupec není lineární kombinací nejvýše $d - 2$ předcházejících sloupců.

Problémy k řešení

Problémy 1

2. *Ukažte, že je-li H Hadamardova matice řádu n, je nutně $n = 1, 2$ nebo je n násobek 4. Poznamenejme, že existuje hypotéza, že pokud n je násobek čtyř, existuje Hadamardova matice řádu n.*

Problémy k řešení

Problémy 1

4. *Ukažte, že je-li H Hadamardova matice řádu n, je nutně* $n = 1, 2$ *nebo je n násobek* 4. *Poznamenejme, že existuje hypotéza, že pokud n je násobek čtyř, existuje Hadamardova matice řádu n.*
5. *Sestrojte binární kódování* $\mathcal{C}$ *typu* $[30, 11, 6]$. *Zjistěte kolik má* $\mathcal{C}$ *kódových slov a jaké jsou jeho schopnosti opravy chyb.*

Problémy k řešení

Problémy 1

6. *Nechť $\mathcal{C}$ je cyklický binární kód. Ukažte, že pokud $\mathcal{C}$ neobsahuje slovo* $11\ldots1$, *pak všechna kódová slova mají sudou váhu.*

Problémy k řešení

Problémy 1

6. *Nechť $\mathcal{C}$ je cyklický binární kód. Ukažte, že pokud $\mathcal{C}$ neobsahuje slovo* $11\ldots1$*, pak všechna kódová slova mají sudou váhu.*
7. *Ukažte, že Reed-Mullerovo kódování $\mathcal{C}(1, m)$ je tentýž kód, jakožto Hadamardovo kódování získané m-násobným použitím Kroneckerova součinu matice*

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

se sebou samou.