

Diskrétní matematika – 8. týden

Lineární kódy

Lukáš Vokřínek

Masarykova univerzita
Fakulta informatiky

podzim 2020

Obsah přednášky

- 1 (n, k) -kódy
- 2 Polynomiální kódy
- 3 Lineární kódy

Doporučené zdroje

- Jan Slovák, Martin Panák, Michal Bulant
Matematika drsně a svižně, e-text na
www.math.muni.cz/Matematika_drsne_svizne.

Doporučené zdroje

- Jan Slovák, Martin Panák, Michal Bulant
Matematika drsně a svižně, e-text na
www.math.muni.cz/Matematika_drsne_svizne.
- W. J. Gilbert, W. K. Nicholson, Modern algebra with applications, 2nd ed. John Wiley and Sons (Pure and applied mathematics) ISBN 0-471-41451-4

Plán přednášky

- 1 (n, k)-kódy
- 2 Polynomiální kódy
- 3 Lineární kódy

Při přenosu informace zpravidla dochází k její deformaci. Budeme pro jednoduchost pracovat s modelem, kdy jednotlivé částičky informace jsou buď nuly nebo jedničky (tj. prvky v \mathbb{Z}_2) a přenášíme slova o k bitech.

Při přenosu informace zpravidla dochází k její deformaci. Budeme pro jednoduchost pracovat s modelem, kdy jednotlivé částičky informace jsou buď nuly nebo jedničky (tj. prvky v \mathbb{Z}_2) a přenášíme slova o k bitech.

Přenosové chyby chceme

- 1 rozpoznávat
- 2 opravovat

a za tím účelem přidáváme dodatečných $n - k$ bitů informace pro pevně zvolené $n > k$. Mluvíme pak o (n, k) -kódu.

Při přenosu informace zpravidla dochází k její deformaci. Budeme pro jednoduchost pracovat s modelem, kdy jednotlivé částičky informace jsou buď nuly nebo jedničky (tj. prvky v \mathbb{Z}_2) a přenášíme slova o k bitech.

Přenosové chyby chceme

- 1 rozpoznávat
- 2 opravovat

a za tím účelem přidáváme dodatečných $n - k$ bitů informace pro pevně zvolené $n > k$. Mluvíme pak o (n, k) -kódu.

Všech slov o k bitech je 2^k a každé z nich má jednoznačně určovat jedno **kódové slovo** z 2^n možných. Máme tedy ještě

$$2^n - 2^k = 2^k(2^{n-k} - 1)$$

slov, které jsou chybové. Lze tedy tušit, že pro veliké k nám i malý počet přidaných bitů dává hodně redundantní informace.

Úplně jednoduchým příkladem je **kód kontrolující paritu**. Kódové slovo o $k + 1$ bitech je určeno tak, aby přidáním prvního bitu byl zaručen sudý počet jedniček ve slově.

Úplně jednoduchým příkladem je **kód kontrolující paritu**. Kódové slovo o $k + 1$ bitech je určeno tak, aby přidáním prvního bitu byl zaručen sudý počet jedniček ve slově.

Pokud při přenosu dojde k lichému počtu chyb, přijdeme na to. Dvě různá kódová slova se při tomto kódu vždy liší alespoň ve dvou pozicích, chybové slovo se ale od dvou různých kódových slov liší pouze v pozici jedné. Nemůžeme proto umět chyby opravovat ani kdybychom věděli, že došlo k právě jedné.

Úplně jednoduchým příkladem je **kód kontrolující paritu**. Kódové slovo o $k + 1$ bitech je určeno tak, aby přidáním prvního bitu byl zaručen sudý počet jedniček ve slově.

Pokud při přenosu dojde k lichému počtu chyb, přijdeme na to. Dvě různá kódová slova se při tomto kódu vždy liší alespoň ve dvou pozicích, chybové slovo se ale od dvou různých kódových slov liší pouze v pozici jedné. Nemůžeme proto umět chyby opravovat ani kdybychom věděli, že došlo k právě jedné.

Navíc neumíme detekovat tak obvyklé chyby, jako je záměna dvou sousedních hodnot ve slově.

Definice

Hammingova vzdálenost dvou slov je rovna počtu bitů, ve kterých se liší.

Definice

Hammingova vzdálenost dvou slov je rovna počtu bitů, ve kterých se liší.

Věta

- 1 *Kód odhaluje r a méně chyb právě, když je minimální Hammingova vzdálenost kódových slov právě $r + 1$.*
- 2 *Kód opravuje r a méně chyb právě, když je minimální Hammingova vzdálenost kódových slov právě $2r + 1$.*

Plán přednášky

- 1 (n, k) -kódy
- 2 Polynomiální kódy
- 3 Lineární kódy

Jak konstruovat kódová slova, abychom je snadno rozpoznali?
Kontrolu parity jsme už viděli, další triviální možnost je prosté opakování bitů – např. $(3, 1)$ -kód bere jednotlivé bity a posílá je třikrát po sobě.

Jak konstruovat kódová slova, abychom je snadno rozpoznali?
Kontrolu parity jsme už viděli, další triviální možnost je prosté opakování bitů – např. $(3, 1)$ -kód bere jednotlivé bity a posílá je třikrát po sobě.

Docela systematickou cestou je využití dělitelnosti polynomů.

Zpráva $b_0b_1 \dots b_{k-1}$ je reprezentována jako polynom

$$m(x) = b_0 + b_1x + \dots + b_{k-1}x^{k-1} \in \mathbb{Z}_2[x].$$

Jak konstruovat kódová slova, abychom je snadno rozpoznali? Kontrolu parity jsme už viděli, další triviální možnost je prosté opakování bitů – např. $(3, 1)$ -kód bere jednotlivé bity a posílá je třikrát po sobě.

Docela systematickou cestou je využití dělitelnosti polynomů.

Zpráva $b_0b_1 \dots b_{k-1}$ je reprezentována jako polynom

$$m(x) = b_0 + b_1x + \dots + b_{k-1}x^{k-1} \in \mathbb{Z}_2[x].$$

Definice

Nechť $p(x) = a_0 + \dots + a_{n-k}x^{n-k} \in \mathbb{Z}_2[x]$ je polynom s $a_0 = 1$, $a_{n-k} = 1$. **Polynomiální kód generovaný polynomem $p(x)$** je (n, k) -kód jehož slova jsou polynomy stupně menšího než n dělitelné $p(x)$.

Jak konstruovat kódová slova, abychom je snadno rozpoznali? Kontrolu parity jsme už viděli, další triviální možnost je prosté opakování bitů – např. $(3, 1)$ -kód bere jednotlivé bity a posílá je třikrát po sobě.

Docela systematickou cestou je využití dělitelnosti polynomů.

Zpráva $b_0b_1 \dots b_{k-1}$ je reprezentována jako polynom

$$m(x) = b_0 + b_1x + \dots + b_{k-1}x^{k-1} \in \mathbb{Z}_2[x].$$

Definice

Nechť $p(x) = a_0 + \dots + a_{n-k}x^{n-k} \in \mathbb{Z}_2[x]$ je polynom s $a_0 = 1$, $a_{n-k} = 1$. **Polynomiální kód generovaný polynomem $p(x)$** je (n, k) -kód jehož slova jsou polynomy stupně menšího než n dělitelné $p(x)$.

Zpráva $m(x)$ je zakódována jako $v(x) = r(x) + x^{n-k}m(x)$, kde $r(x)$ je zbytek po dělení polynomu $x^{n-k}m(x)$ polynomem $p(x)$.

Z definice víme

$$v(x) = x^{n-k}m(x) + r(x) = q(x)p(x) + r(x) + r(x) = q(x)p(x).$$

Budou tedy všechna kódová slova dělitelná $p(x)$.

Z definice víme

$$v(x) = x^{n-k}m(x) + r(x) = q(x)p(x) + r(x) + r(x) = q(x)p(x).$$

Budou tedy všechna kódová slova dělitelná $p(x)$.

Původní zpráva je obsažena přímo v polynomu $v(x)$, takže dekódování správného slova je snadné.

Z definice víme

$$v(x) = x^{n-k}m(x) + r(x) = q(x)p(x) + r(x) + r(x) = q(x)p(x).$$

Budou tedy všechna kódová slova dělitelná $p(x)$.

Původní zpráva je obsažena přímo v polynomu $v(x)$, takže dekódování správného slova je snadné.

Příklad

- 1 Polynom $p(x) = 1 + x$ generuje $(n, n - 1)$ -kód kontroly parity pro všechna $n \geq 3$.
- 2 Polynom $p(x) = 1 + x + x^2$ generuje $(3, 1)$ -kód opakování bitů.

První tvrzení plyne z toho, že $1 + x$ dělí polynom $v(x)$ tehdy a jen tehdy, když $v(1) = 0$ a to nastane tehdy, když je ve $v(x)$ sudý počet nenulových koeficientů. Druhé je zřejmé.

Přenos slova $v \in \mathbb{Z}_2[x]$ dopadne přijmem polynomu

$$u(x) = v(x) + e(x)$$

kde $e(x)$ je tzv. **chybový polynom** reprezentující vektor chyby přenosu.

Přenos slova $v \in \mathbb{Z}_2[x]$ dopadne příjmem polynomu

$$u(x) = v(x) + e(x)$$

kde $e(x)$ je tzv. **chybový polynom** reprezentující vektor chyby přenosu.

Chyba je rozpoznatelná pouze, když generátor kódu $p(x)$ nedělí $e(x)$. Máme proto zájem o polynomy, které nevystupují jako dělitelé zbytečně často.

Přenos slova $v \in \mathbb{Z}_2[x]$ dopadne příjmem polynomu

$$u(x) = v(x) + e(x)$$

kde $e(x)$ je tzv. **chybový polynom** reprezentující vektor chyby přenosu.

Chyba je rozpoznatelná pouze, když generátor kódu $p(x)$ nedělí $e(x)$. Máme proto zájem o polynomy, které nevystupují jako dělitelé zbytečně často.

Definice

Ireducibilní polynom $p(x) \in \mathbb{Z}_2[x]$ stupně m se nazývá **primitivní**, jestliže $p(x)$ dělí polynom $(1 + x^\ell)$ pro $\ell = 2^m - 1$ ale nedělí jej pro žádná menší ℓ .

Věta

Je-li $p(x)$ primitivní polynom stupně m , pak pro všechna $n \leq 2^m - 1$ rozpoznává příslušný $(n, n - m)$ -kód všechny jednoduché a dvojité chyby.

Věta

Je-li $p(x)$ primitivní polynom stupně m , pak pro všechna $n \leq 2^m - 1$ rozpoznává příslušný $(n, n - m)$ -kód všechny jednoduché a dvojité chyby.

Důsledek

Je-li $q(x)$ primitivní polynom stupně m , pak pro všechna $n \leq 2^m - 1$ rozpoznává $(n, n - m - 1)$ -kód generovaný polynomem $p(x) = q(x)(1 + x)$ všechny dvojité chyby a všechna slova s lichým počtem chyb.

Tabulka dává o informace o výsledcích předchozích dvou vět pro několik polynomů:

Tabulka dává o informace o výsledcích předchozích dvou vět pro několik polynomů:

primitivní polynom	kontrolní bity	délka slova
$1 + x + x^2$	2	3
$1 + x + x^3$	3	7
$1 + x + x^4$	4	15
$1 + x^2 + x^5$	5	31
$1 + x + x^6$	6	63
$1 + x^3 + x^7$	7	127
$1 + x^2 + x^3 + x^4 + x^8$	8	255
$1 + x^4 + x^9$	9	511
$1 + x^3 + x^{10}$	10	1023

Nástroje pro konstrukci primitivních polynomů dává teorie konečných polí. Souvisí s tzv. primitivními prvky v Galoisových polích $G(2^m)$.

Nástroje pro konstrukci primitivních polynomů dává teorie konečných polí. Souvisí s tzv. primitivními prvky v Galoisových polích $G(2^m)$.

Ze stejné teorie lze také dovodit příjemnou realizaci dělení se zbytkem (tj.) ověřování, zda je přijaté slovo kódové, pomocí zpožděvacích registrů. Jde o jednoduchý obvod s tolika prvky, kolik je stupeň polynomu.

Plán přednášky

- 1 (n, k) -kódy
- 2 Polynomiální kódy
- 3 **Lineární kódy**

Definice

Lineární kód je injektivní lineární zobrazení $g : (\mathbb{Z}_2)^k \rightarrow (\mathbb{Z}_2)^n$. Matice G typu n/k reprezentující toto zobrazení v standardních bazích se nazývá generující **matice kódu**.

Definice

Lineární kód je injektivní lineární zobrazení $g : (\mathbb{Z}_2)^k \rightarrow (\mathbb{Z}_2)^n$. Matice G typu n/k reprezentující toto zobrazení v standardních bazích se nazývá generující **matice kódu**.

Pro každé slovo u je

$$v = G \cdot u$$

příslušné kódové slovo.

Věta

Každý polynomiální (n, k) -kód je lineární kód.

Věta

Každý polynomiální (n, k)-kód je lineární kód.

Matice příslušná k polynomu $p(x) = 1 + x + x^3$ a jím určenému (7, 4)-kódu je

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

Věta

Je-li $g : (\mathbb{Z}_2)^k \rightarrow (\mathbb{Z}_2)^n$ lineární kód s (blokově zapsanou) maticí

$$G = \begin{pmatrix} P \\ \mathbb{I}_k \end{pmatrix},$$

potom zobrazení $h : (\mathbb{Z}_2)^n \rightarrow (\mathbb{Z}_2)^{n-k}$ s maticí

$$H = \begin{pmatrix} \mathbb{I}_{n-k} & P \end{pmatrix}$$

má následující vlastnosti

- 1 $\text{Ker } h = \text{Im } g$
- 2 Přijaté slovo $v = G \cdot u$ je kódové slovo právě, když je $H \cdot v = 0$.

Věta

Je-li $g : (\mathbb{Z}_2)^k \rightarrow (\mathbb{Z}_2)^n$ lineární kód s (blokově zapsanou) maticí

$$G = \begin{pmatrix} P \\ \mathbb{I}_k \end{pmatrix},$$

potom zobrazení $h : (\mathbb{Z}_2)^n \rightarrow (\mathbb{Z}_2)^{n-k}$ s maticí

$$H = (\mathbb{I}_{n-k} \quad P)$$

má následující vlastnosti

- 1 $\text{Ker } h = \text{Im } g$
- 2 Přijaté slovo $v = G \cdot u$ je kódové slovo právě, když je $H \cdot v = 0$.

Matici H z věty se říká **matice kontroly parity** přílušného (n, k) -kódu.

Jak jsme viděli, přenos zprávy u dává výsledek

$$v = u + e,$$

kde ale neznáme u , e a hledáme takový “rozklad”, kde e obsahuje co nejméně jedniček (oprava chyby za předpokladu co nejmenšího počtu chyb).

Jak jsme viděli, přenos zprávy u dává výsledek

$$v = u + e,$$

kde ale neznáme u , e a hledáme takový “rozklad”, kde e obsahuje co nejméně jedniček (oprava chyby za předpokladu co nejmenšího počtu chyb).

Je-li $v = \begin{pmatrix} x \\ y \end{pmatrix}$, pak jednou z možností na odeslanou zprávu je

$Gy = \begin{pmatrix} Py \\ y \end{pmatrix}$ (ne nutně optimální), tedy

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} Py \\ y \end{pmatrix} + \begin{pmatrix} x + Py \\ 0 \end{pmatrix}$$

Jak jsme viděli, přenos zprávy u dává výsledek

$$v = u + e,$$

kde ale neznáme u , e a hledáme takový “rozklad”, kde e obsahuje co nejméně jedniček (oprava chyby za předpokladu co nejmenšího počtu chyb).

Je-li $v = \begin{pmatrix} x \\ y \end{pmatrix}$, pak jednou z možností na odeslanou zprávu je

$Gy = \begin{pmatrix} Py \\ y \end{pmatrix}$ (ne nutně optimální), tedy

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} Py \\ y \end{pmatrix} + \begin{pmatrix} x + Py \\ 0 \end{pmatrix}$$

kde $s = x + Py$ je právě syndrom slova v a jedná se tedy o chybu za předpokladu, že k ní došlo pouze na kontrolních bitech (z informačních bitů lze proto přechíst původní slovo).

Protože kódová slova jsou právě součty sloupců matice G , lze všechny další možnosti obdržet přičítáním sloupců g_i ke kódovému slovu i chybě:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \left(\begin{pmatrix} Py \\ y \end{pmatrix} + g_i \right) + \left(\begin{pmatrix} x + Py \\ 0 \end{pmatrix} + g_i \right)$$

atd.

Protože kódová slova jsou právě součty sloupců matice G , lze všechny další možnosti obdržet přičítáním sloupců g_i ke kódovému slovu i chybě:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \left(\begin{pmatrix} Py \\ y \end{pmatrix} + g_i \right) + \left(\begin{pmatrix} x + Py \\ 0 \end{pmatrix} + g_i \right)$$

atd.

Přítom přičtení každého sloupce vyrobí jednu 1 v informačních bitech chyby, snažíme se jejich počet kompenzovat snížením počtu 1 v kontrolních bitech. Toto budeme zkoušet pouze pro malý počet chyb, viz cvičení.