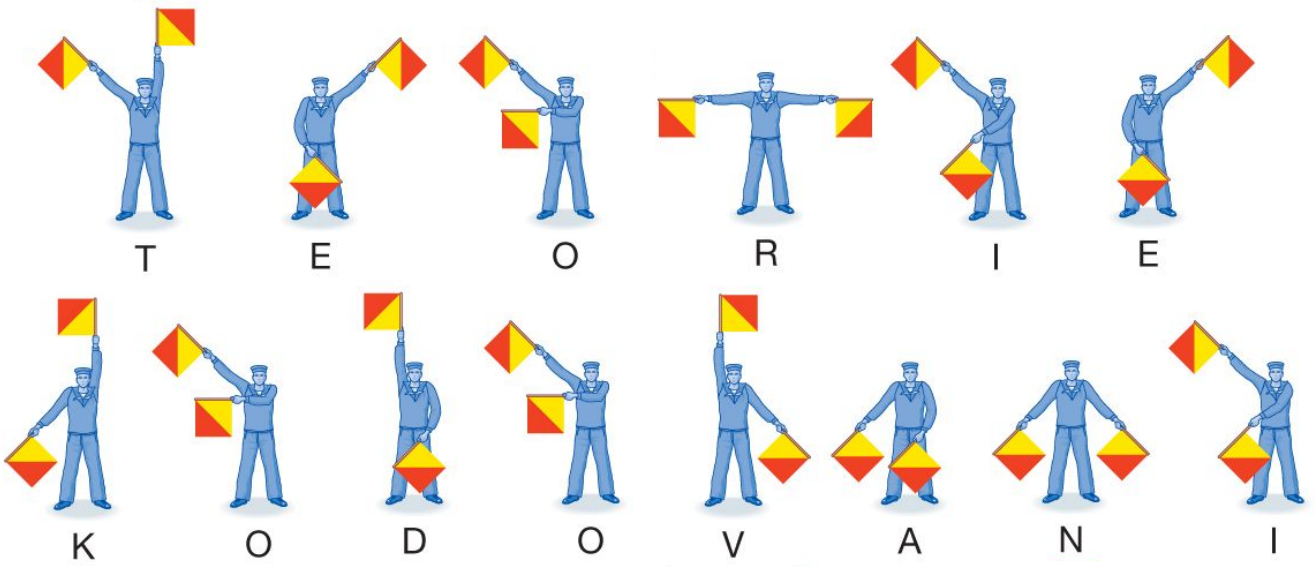
Teorie kódování



Petr Kovář

2024

Petr Kovář
Teorie kódování

Úvodem

Text, který právě čtete, vzniká jako systematická příprava přednášek pro magisterské i doktorské studium na technické vysoké škole. Při výběru témat a přípravě textu jsem vycházel z osnov předmětu, z knihy *A First Course in Coding Theory* od R. Hilla [H], z knihy *Kódování* od Jiřího Adámka a celé řady dalších zdrojů.

Pokud máte pocit, že v textu je nějaká nesrovnalost, dejte mi vědět. Je velmi pravděpodobné, že se jedná o chybu nebo překlep.

Jak číst tento text

Na konci každé podkapitoly najdete příklady k procvičení probrané látky. Protože není smysluplné vybudovat nejdříve celou teorii a teprve potom řešit příklady, tak některá cvičení se odvolávají na pojmy, které budou zavedeny teprve v pozdějších kapitolách. Věřím, že není problém nalistovat si příslušnou definici (na konci textu najdete rejstřík) a pak takové příklady vyřešit.

Poděkování

Děkuji Tereze Kovářové, která připravila první rozšířenou osnovu tohoto textu a se kterou jsem spolupracoval na přípravě textu i většiny cvičení.

K použitým symbolům

Příklady označené „*“ patří k náročnějším. Jejich řešení obvykle vyžaduje delší výpočet nebo pečlivější rozbor. Pro řešení příkladů označených „**“ je třeba nějaký nápad nebo výsledek z jiné oblasti matematiky. Zdůrazněme ale, že hvězdička neznamená nutně „to nikdy nevyřeším“. Naproti tomu příklady označené „$^\heartsuit$“ jsou tak lehké, že jejich řešení je možné zpaměti a jen s užitím základních pojmů.

V Krmelíně 20. května 2024.

Obsah

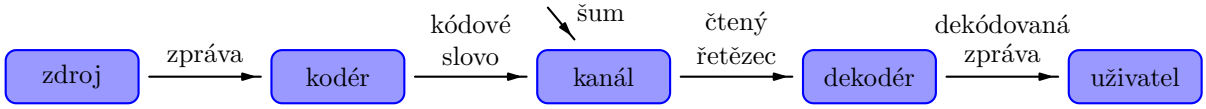
Kapitola 1. Úvod do samoopravných kódů

Samoopravné kódy se používají téměř ve všech moderních digitálních elektronických systémech, datových sítích i záznamových médiích. Kódování s ohledem na opravu chyb umožňuje efektivně posílat, ukládat, spravovat i získávat digitální informace. Samoopravné kódy využíváme zejména tam, kde bezchybnost hraje vyšší roli než rychlost přenosu či objem dat záznamu. Teorie kódování je pěknou aplikací na rozhraní teoretické algebry, lineární algebry, kombinatoriky a teorie designů. Detekce a oprava chyb spojuje matematické disciplíny, informatiku, telekomunikace i elektrotechniku. Řada příkladů uvedených v textu toto propojení ukazuje.

1.1. Motivace pro samoopravné kódy

Samoopravné kódy se používají pro přenos rušeným komunikačním kanálem nebo pro ukládaní dat odolné proti ztrátám části dat. Rušený kanál může být telefonní linka, kanál televizního vysílání či satelitního vysílání, nebo kanály bezdrátového pozemního přenosu dat nebo satelitní přenos signálu nebo dokonce komunikace s vesmírnými sondami. Může se jednat o záznamové médium jako magnetická páska nebo pevný disk. Rušení může být způsobeno tepelným šumem, rušením elektromagnetického pole nebo nedokonalostí přístroje, které způsobí, že přijatá data se liší od vyslaných dat.

Samoopravný kód
Smyslem samoopravných kódů je takové zakódování dat, aby původní zpráva mohla být s malou námahou získána i v případě, že dojde k porušení části dat. Obvykle se tak děje přidáním jisté redundance, prodloužením zprávy tak, aby obsahovala nejen výchozí data, ale další část navíc. Původní zpráva pak může být obnovena, pokud dojde ke ztrátě nebo poškození části dat.



Obrázek 1.1.: *Přenos zašuměným kanálem.*

Cílem není přenos informace skrýt před nepovolanými čtenáři, ale naopak umožnit správné přečtení nebo dekódování informace i v případě, že informační kanál je rušen nebo paměťové médium je mírně poškozeno. Správné určení kódových slov by mělo být možné i bez nutnosti nového posílání nebo čtení, neboť takový postup může být velmi zdlouhavý, případně nežádoucí nebo nemožný. Dostáváme se k následující jednoduché definici.

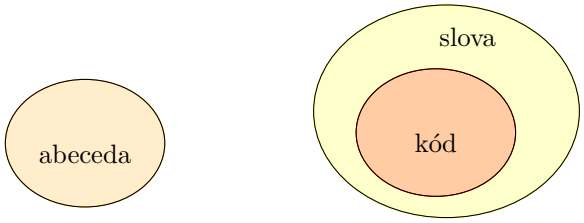
Cvičení

1.1.1. Astronomická observatoř zachytila signál, který přicházel ze souhvězdí Lyry. Signál se skládá z periodicky se opakující sekvence 35 bitů 00101000001000111111110101010010100. *Vědci se domnívají, že signál vyslaly bytosti podobné netopýrům. Proč?*

1.2. Abeceda, slovo, kód

Nejprve zavedeme ústřední pojmy celého textu: kód a kódové slovo.

Definice Kód
Abeceda je konečná množina *symbolů* nebo *znaků*. *Slovo* je posloupnost znaků dané abecedy. *Kódové slovo* je posloupnost znaků dané abecedy, která je v množině kódu. *Kód* je neprázdnou podmnožinou množiny slov, jedná se o množinu kódových slov.



Obrázek 1.2.: *Abeceda, množina slov a kód.*

Všimněte si, že každé kódové slovo je posloupnost znaků. Zpravidla mají všechna kódová slova stejnou délku, ale není to nutné, definice to nevyžaduje (Obrázek 1.2.).

Slovo nebo kódové slovo budeme značit $\vec{x}$. Jedná se o konečnou posloupnost n symbolů abecedy, proto $\vec{x} = (x_1, x_2, \ldots, x_n)$, kde $x_1, x_2, \ldots, x_n$ jsou symboly dané abecedy. Pokud nebude hrozit mýlka, tak využijeme kratší zápis $\vec{x} = x_1 x_2 \ldots x_n$. Symbol x_i budeme nazývat *souřadnice* nebo *znak* slova $\vec{x}$. Můžeme tak například mluvit o první souřadnici x_1, poslední souřadnici x_n nebo i-té souřadnici x_i slova $\vec{x}$, $\vec{x} = = x_1 x_2 \ldots x_n$.

Příklad 1.1. Uveďme několik příkladů abeced a kódů.

1) Anglická abeceda obsahuje 27 znaků (26 písmen a mezeru). Anglická slova můžeme chápat jako kód, přičemž slova kódu jsou platná anglická slova, zatímco slova jsou posloupnosti písmen, která anglická slova netvoří.
2) Česká abeceda obsahuje 43 znaků (42 písmen a mezeru). posloupnosti písmen tvoří slova, ale jen česká slova tvoří kód.
3) Binární abeceda obsahuje jen dva znaky 0 a 1. Označme $F_2 = \{0, 1\}$. Binární řetězce dané délky n tvoří množinu binárních slov $(F_2)^n$. *Binárním kódem* budeme rozumět nějakou podmnožinu C množiny binárních slov, tedy podmnožinu množiny posloupností bitů (nul a jedniček). Těmto vybraným posloupnostem říkáme *kódová slova*.
4) Hexadecimální abeceda, která například určuje MAC adresy v počítačové síti, je kód sestavený z abecedy 16 znaků s kódovými slovy délky 12. Až na jistá omezení může být každé slovo množiny $(F_{16})^{12}$ kódovým slovem. MAC adresy nemají implementován žádný samoopravný kód.
5) Morseova abeceda může být chápána jako binární nebo jako ternární kód. Morseova abeceda není samoopravný kód, slouží pro přenos informace jednoduchým binárním kanálem. Abeceda obsahuje tři symboly: tečku, čárku a mezeru, která odděluje tečky, čárky, ale také znaky a slova. Bez krátké mezery, která odděluje tečky a čárky, bez střední mezery, která odděluje znaky, a bez dlouhé mezery, která odděluje slova, by Morseova abeceda nebyla srozumitelná. Naproti tomu Morseova abeceda není plnohodnotným ternárním kódem, neboť mezera plní jen roli oddělovače symbolů a oddělovače slov. Morseovu abecedu je možno popsat binárním kódem, který má 5 kódových slov:
   - tečka: 1
   - čárka: 111
   - mezera mezi tečkami a čárkami: 0
   - krátká mezera mezi písmeny: 000
   - střední mezera mezi slovy: 0000000.
6) Huffmannův kód je způsob bezztrátové komprese dat do bitových řetězců. Využívá různé četnosti znaků v obvyklém textu daného jazyka, případně v aktuálním textu. Abeceda Huffmanova kódu závisí na jazyku aktuálně přenášené zprávy. Znaky s vysokým počtem výskytů se kódují pomocí kratších sekvencí, zatímco relativně málo se vyskytující znaky mohou mít delší reprezentaci. Nejedná se o samoopravný kód.

Všimněte si, že rozlišujeme množinu slov jazyka a jazyk. Zatímco množinu slov nějakého jazyka můžeme považovat za kód, tak jazyk spíše ne. Význam slov závisí na gramatice, na slovosledu. Aby věta měla nějaký smysl, musí dodržet určitá pravidla, což je vlastnost, kterou od kódu nepožadujeme.

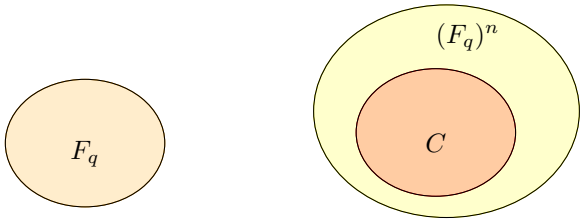
q-ární kód

V předchozích odstavcích jsme zmínili binární kódy. V digitální technice se vskutku nejčastěji setkáme s binárními kódy, které pracují jen se dvěma symboly 0 a 1, ale v analogových systémech může mít smysl pracovat i více symboly. Dostáváme se přirozeně k definici q-árního kódu.

Definice ***q*-ární kód**

q-ární kód je kód s kódovými slovy sestavenými z q různých symbolů $s_1, s_2, \ldots, s_q$. Abeceda kódu je množina symbolů $F_q = \{s_1, s_2, \ldots, s_q\}$, zpravidla používáme abecedu $F_q = \mathbb{Z}_q = \{0, 1, \ldots, q-1\}$.

Ve speciálním případě, kdy $F_q = \{0, 1\}$, mluvíme o *binárním kódu*. Jestliže abeceda obsahuje tři symboly, například $F_q = \{0, 1, 2\}$, mluvíme o *ternárním kódu*.



Obrázek 1.3.: *q-ární abeceda, množina slov délky n a kód délky n.*

Označení F_q vychází z anglického termínu „field“. Pokud je q mocninou prvočísla, tak prvky množiny F_q chápeme jako prvky číselného tělesa $(\mathbb{Z}_q, +, \cdot)$. Pokud ale q není mocnina prvočísla, tak nemá smysl mluvit o F_q jako o „číselném tělese“, neboť v předmětu Algebra bylo ukázáno, že takové těleso neexistuje. I v takovém případě pracujeme s množinou $F_q = \{0, 1, \ldots, q-1\}$.

Operace se symboly (prvky číselného tělesa) pro sestavení kódu mohou ale nemusí hrát roli. V prvních kapitolách operace sčítání „+“ a násobení „$\cdot$“ symbolů v číselném tělese využijeme jen omezeně. Významnější roli budou operace hrát zejména v dalších kapitolách, například u lineárních kódů.

Cvičení

1.2.1.$^{\heartsuit}$ *Kolik existuje různých binárních slov délky n?*

1.2.2.$^{\heartsuit}$ *Kolik existuje různých binárních kódů délky n?*

1.2.3.$^{\heartsuit}$ *Každé rodné číslo musí být dělitelné 11 a můžeme je považovat za kód v množině deseticiferných čísel. a) Umí kód rodných čísel detekovat chybu? b) Umí kód rodných čísel opravit chybu?*

1.2.4. Umí ISBN-10 kód detekovat chybu? Umí ISBN-10 kód opravit chybu?

Značení

Symbolem $(F_q)^n$ značíme množinu všech posloupností n-prvků $(a_1, a_2, \ldots, a_n) = a_1 a_2 \ldots a_n$, kde $a_i \in F_q$. Množina $(F_q)^n$ má q^n prvků, n-prvkových posloupností. Těmto posloupnostem říkáme *slova* nebo *vektory* délky n. Všimněte si, že ne každé „slovo“ je kódovým slovem nějakého kódu.

Připomeňme, že formálně bychom slova délky n měli značit $(a_1, a_2, \ldots, a_n)$, neboť se jedná o posloupnosti n symbolů abecedy.

Příklad 1.2. Uveďme několik příkladů q-árních kódů.

1) Binární řetězce dané délky n tvoří množinu binárních slov $(F_2)^n$. Vhodné podmnožiny C množiny binárních slov $(F_2)^n$ budou tvořit binární kódy. Každý binární řetězec délky n je slovem. Ta slova, která patří do C, jsou kódová slova.
2) Slova anglické abecedy nepovažujeme za kódová slova posloupností $(F_{27})^n$, přestože F_{27} je těleso.
3) Devítimístná telefonní čísla můžeme chápat jako slova desítkové abecedy F_{10} z množiny $(F_{10})^9$. Množina F_{10} však těleso netvoří. Každé telefonní číslo je slovem, avšak telefonní čísla operátor nevybírá tak, aby tvořila samoopravný kód.

Jestliže všechna kódová slova daného kódu mají stejnou délku n, tak kód nazýváme *q-ární kód délky n*. Takový kód je podmnožinou množiny slov $(F_q)^n$.

Jedním z nejjednodušších kódů, které umožňují detekci a opravu chyb, je opakovací kód.

Definice Opakovací kód

Mějme množinu $(F_q)^n$. *Opakovací q-ární kód* délky n je kód, jehož každé kódové slovo je posloupnost n kopií stejného symbolu abecedy F_q. Kódových slov je q.

$$C = \left\{ \begin{array}{l} 00\ldots0 \\ 11\ldots1 \\ \quad\vdots \\ \underbrace{(q-1)(q-1)\ldots(q-1)}_{n} \end{array} \right.$$

Příklad 1.3. Uveďme několik příkladů opakovacích kódů.

1) Binární opakovací kód délky n obsahuje jen dvě kódová slova $00\ldots0$, $11\ldots1$.
2) Hexadecimální opakovací kód délky 5 obsahuje 16 kódových slov 00000, 11111, ..., 99999, AAAAA, ..., FFFFF.
3) Opakovací kód anglické abecedy s délkou slova 3 obsahuje 26 kódový slov AAA, BBB, ..., ZZZ a jedno kódové slovo sestávající ze tří mezer „␣␣␣“.

Příklad 1.4.

1) Kód C_1 z Příkladu 1.9. ani kódy C_2 z Příkladu 1.10. nejsou opakovací kódy.
2) Kód C_3 z Příkladu 1.11. není opakovací kód.

Příklad 1.5. Kód dva z pěti
Kód *dva z pěti* obsahuje deset kódových slov 11000, 10100, 10010, 10001, 01100, 01010, 01001, 00110, 00101, 00011. Nejedná se o opakovací kód.

1.3. Redundance, detekce a korekce chyb

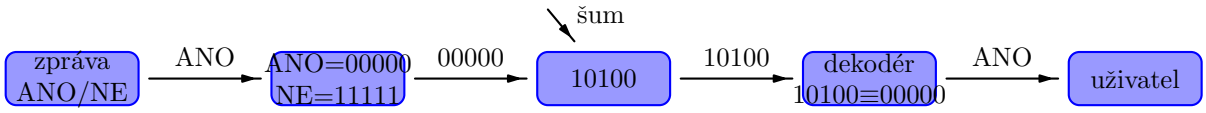
Nutnou vlastností kódu, který umožní detekci chyby přenosu, případně opravu chyby, je *redundance*. Redundance kódu znamená, že posílaná data obsahují více znaků, než posílaná zpráva. Přitom kódová slova jsou volena tak, aby umožnila detekci a opravu chyb. Hlavní myšlenku ukážeme nejprve na následujících příkladech.

Příklad jednoduchého binárního kódu
Uvedeme několik příkladů binárních kódů různé délky a s různým počtem kódových slov.

Příklad 1.6. Mějme binární opakovací kód délky 5. Jeho kódová slova jsou

$$00000, \quad 111111.$$

Tento kód slouží k přenesení jen dvou možných zpráv. Můžeme si je představit jako zprávy ANO a NE které odpovídají bitům 0 a 1. Kód vznikl tak, že každý bit je jednoduše pětkrát zopakován. Takovému kódu budeme říkat „opakovací kód délky 5“. Opakování bitů je jeden z nejjednodušších způsobů, jak nadbytečnost informace může zajistit správnost i v případě, že část zprávy bude ztracena/porušena.



Obrázek 1.4.: *Oprava chyby při přenosu zašuměným kanálem.*

Představme si, že zpráva „ano“ je přenášena zašuměným kanálem. Je odesláno kódové slovo 00000, ale místo něj je přijato slovo 10100. Budeme předpokládat, že počet chyb, které při přenosu nastaly, není velký, že pravděpodobnost chyby je poměrně malá. Proto nejpravděpodobnější se jeví, že místo slova 10100 mělo být odesláno kódové slovo 00000.

Všimněte si, že nikdy nemůžeme mít úplnou jistotu, že dekódované slovo je jistě správné. Pokud předpokládáme, že chyb v přenosu nastalo málo, můžeme určit nejpravděpodobnější slovo. Pokud by byl kanál příliš zatížen šumem a každý přečtený bit byl v podstatě náhodný, tak dekódovaná slova i sebelepšího kódu budou náhodná a zpráva bude ztracena. Více o předpokládané kvalitě kanálu je psáno na straně 6.

Poznámka 1.1. Ztracené bity
Při přenosu zašuměným kanálem se často mluví o ztracených bitech. Rozumíme tím zpravidla situaci, že přečteme jiné stejně dlouhé kódové slovo, kde některé symboly (zpravidla bity) jsou zatíženy chybou. Říkáme, že bity jsou neplatné nebo ztracené, nikoliv že by zpráva obsahovala méně symbolů.

Příklad 1.7. Názvy ulic

Názvy ulic ve městě můžeme chápat jako kód. Abeceda kódu sestává z až 42 písmen a mezery, délka kódu není konstantní. Název každé ulice je kódové slovo, ale takový kód není příkladem dobrého kódu. Může stačit změna jediného znaku, například „Holasova ulice“ a „Halasova ulice“, která změní kódové slovo a dostaneme jinou adresu ve městě.

Příklad 1.8. Telefonní čísla
Telefonní čísla v České republice odpovídají desítkovému kódu délky 9. Takový kód umožňuje miliardu telefonních čísel. Ovšem opět kód nemá implementovánu žádnou detekci ani korekci chyb.

Lze sestavit takový kód systému telefonních čísel s 82 milióny telefonních čísel, že pokud bude jedna cifra telefonního čísla chybně zvolena, stále bude možno rozpoznat a udělat správné spojení. Naproti tomu nebude možno zajistit telefonní čísla na přání.

Navigování
V následující trojici navazujících příkladů ukážeme, jak můžeme upravit kód, který bude detekovat malé množství chyb a dále, jak je možno malé množství chyb opravit bez nutnosti opakovaného zasílání.

Představme si, že navigujeme robota náročným terénem. Například na jiné planetě může jedna sonda na orbitě pečlivě proměřovat terén a vozítku, které se pohybuje po povrchu planety navrhovat směr, kterým se pohybovat. Pro jednoduchost předpokládejme, že pracujeme jen se čtyřmi základními směry sever/jih/východ/západ, které si pro pro jednoduchost označíme S, J, V, Z.

Následující kód umožňuje předání jednoduchých pokynů, kterým směrem se má v souřadnicové síti posunout.

Příklad 1.9. Mějme jednoduchý binární kód C_1

$$\text{kód } C_1 = \begin{cases} 00 & \text{S} \\ 01 & \text{Z} \\ 10 & \text{V} \\ 11 & \text{J}. \end{cases}$$

Nejjednodušší a nejrychlejší způsob, jak předat pokyny vozítku, kterým směrem se pohybovat, znamená odeslat příslušné kódové slovo kódu C_1.

Detekce chyby
Každá chyba přenosu v náročném prostředí může být fatální. Stačí změnit jediný bit kódového slova 00 a vozítko se místo na sever vydá na západ nebo na východ.

Příklad 1.10. Ke každému kódovému slovu přidáme navíc jeden bit následujícím způsobem. Mějme binární kód C_2

$$\text{kód } C_2 = \begin{cases} 000 & \text{S} \\ 011 & \text{Z} \\ 101 & \text{V} \\ 110 & \text{J}. \end{cases}$$

Tento kód je sice delší a jeho přenos zabere o polovinu více prostředků, avšak pokud při přenosu nastane jediná chyba, tak budeme schopni rozpoznat, že k chybě došlo. Snadno ověříme, že změnou jednoho libovolného bitu kódového slova nedostaneme platné kódové slovo, neboť všechna kódová slova mají sudý počet bitů s hodnotou 1.

Změna jednoho libovolného bitu musí dát slovo s lichým počtem bitů 1, a takové slovo není platným kódovým slovem. Vozítko tak po přijetí slova s jednou chybou pozná, že zpráva byla porušena a může si vyžádat opakování přenosu.

Korekce chyby
Nyní předpokládejme, že podmínky pro přenos jsou náročné, zejména, že přenos z vozítka na orbitální sondu je náročná, protože energetické zdroje vozítka jsou omezené. Můžeme představit následující kód, který místo dvou bitů bude mít kódová slova s pěti bity.

Příklad 1.11. Mějme binární kód C_3

$$\text{kód } C_3 = \begin{cases} 00000 & \text{S} \\ 01110 & \text{Z} \\ 10101 & \text{V} \\ 11011 & \text{J}. \end{cases}$$

Tento kód je výrazně delší delší než kód C_1, avšak umožňuje jednak rozpoznat až dvě chyby, které mohou při přenosu nastat, ale hlavně za předpokladu, že chyba při přenosu nastala jediná, chybu opravit tím, že rozpozná příslušné správné kódové slovo.

Všimněte si, že pokud změníte jediný bit kteréhokoliv kódového slova, tak bude výsledné slovo nejvíce podobné původnímu kódovému slovu. Žádné jiné kódové slovo se nebude lišit v jediném bitu.

Co to znamená být „podobné“ a „lišit se“ upřesníme v další části textu. Podrobněji tento kód rozebereme ve Cvičení.

Hammingova vzdálenost

Existuje více způsobů míry podobnosti slov, jednou z nich je Hammingova vzdálenost

Definice Mějme množinu slov $(F_q)^n$ a dvě slova $\vec{x}, \vec{y} \in (F_q)^n$. *Hammingova vzdálenost* dvou slov $\vec{x}$ a $\vec{y}$ v $(F_q)^n$ je počet souřadnic (znaků), ve kterých se slova $\vec{x}$ a $\vec{y}$ liší. Vzdálenost značíme $\mathrm{dist}(\vec{x}, \vec{y})$.

Příklad 1.12. Uvedeme několik příkladů Hammingových vzdáleností.

1) V množině $(F_3)^4$ je vzdálenost $\mathrm{dist}(0122, 1102) = 2$.
2) Opakovací q-ární kód délky n má vzdálenost libovolných dvou kódových slov rovnu n.
3) V kódu „dva z pěti“ mají kódová slova 01010 a 01010 vzdálenost 4, neboť $\mathrm{dist}(01010, 01010) = 4$. Naproti tomu kódová slova 01010 a 11000 mají vzdálenost 2, neboť $\mathrm{dist}(01010, 11000) = 2$.

Je snadné ukázat, že Hammingova vzdálenost tvoří metriku na množině $(F_q)^n$.

Lemma 1.1. Hammingova vzdálenost je metrika
Hammingova vzdálenost je metrika na množině $(F_q)^n$.

Důkaz. Ověříme splnění všech tří podmínek metriky.

(i) $\mathrm{dist}(\vec{x}, \vec{y}) = 0$ právě tehdy když $\vec{x} = \vec{y}$.
(ii) $\mathrm{dist}(\vec{x}, \vec{y}) = \mathrm{dist}(\vec{y}, \vec{x})$ pro všechna slova $\vec{x}, \vec{y} \in (F_q)^n$.
(iii) $\mathrm{dist}(\vec{x}, \vec{y}) \leq \mathrm{dist}(\vec{x}, \vec{z}) + \mathrm{dist}(\vec{z}, \vec{y})$ pro všechna slova $\vec{x}, \vec{y}, \vec{z} \in (F_q)^n$.

Mějme libovolná slova $\vec{x}, \vec{y}, \vec{z} \in (F_q)^n$.
(i) Jestliže $\mathrm{dist}(\vec{x}, \vec{y}) = 0$, tak se slova $\vec{x}$ a $\vec{y}$ neliší v žádném znaku, proto $\vec{x} = \vec{y}$. Naopak, pokud $\vec{x} = \vec{y}$, tak jejich vzdálenost je 0.
(ii) Protože „lišit se“ nezáleží na pořadí slov $\vec{x}$ a $\vec{y}$, tak $\mathrm{dist}(\vec{x}, \vec{y}) = \mathrm{dist}(\vec{y}, \vec{x})$.
(iii) Abychom ukázali trojúhelníkovou nerovnost $\mathrm{dist}(\vec{x}, \vec{y}) \leq \mathrm{dist}(\vec{x}, \vec{z}) + \mathrm{dist}(\vec{z}, \vec{y})$, stačí si uvědomit, že ve slovu $\vec{x}$ můžeme udělat nejprve $\mathrm{dist}(\vec{x}, \vec{z})$ změn, abychom ze slova $\vec{x}$ dostali slovo $\vec{z}$ a potom dalších $\mathrm{dist}(\vec{z}, \vec{y})$ změn, abychom dostali slovo $\vec{y}$. Počet změn $\mathrm{dist}(\vec{x}, \vec{y})$ slova $\vec{x}$, abychom dostali slovo $\vec{y}$ nebude větší než součet počtu postupných změn. Proto platí $\mathrm{dist}(\vec{x}, \vec{y}) \leq \mathrm{dist}(\vec{x}, \vec{z}) + \mathrm{dist}(\vec{z}, \vec{y})$. □

Binární a q-ární symetrický kanál

q-ární symetrický kanál (Obrázek 1.5.) je komunikační kanál, který má následující vlastnosti:

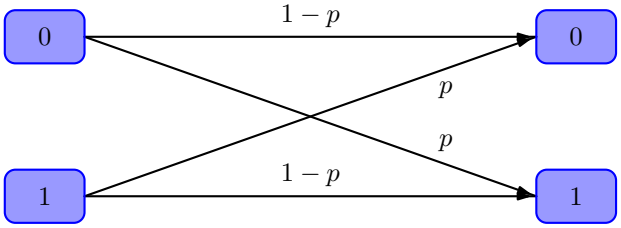
(i) každý přenášený symbol/znak má stejnou pravděpodobnost p, kde $p < \frac{1}{2}$, že bude přijat s chybou,
(ii) pokud je nějaký znak přenesen s chybou, tak každý z $q - 1$ chybných symbolů se může vyskytnout se stejnou pravděpodobností.

Číslo p se nazývá *pravděpodobnost chyby znaku* daného q-árního kanálu.

V obecném případě nemusí být pravda, že daný komunikační kanál je symetrický, jak ukazuje další příklad.

Příklad 1.13. Na analogové telefonní lince jsou „vytáčena čísla“, nejpravděpodobnější chyba znamená, že vytočíme jiný blízký, zpravidla nižší symbol tím, že nedotočíme číselník až k zarážce. Takový kanál pak není symetrický, neboť různé chyby mají různou pravděpodobnost.

Příklad 1.14. Na Obrázku 1.5. je schématicky znázorněn binární symetrický kanál, kde p je pravděpodobnost chyby a $1 - p$ je pravděpodobnost, že symbol bude kanálem přenesen bez chyby.



Obrázek 1.5.: *Pravděpodobnost chyb symbolů.*

V ternárním symetrickém kanálu označme pravděpodobnost chyby p, to je pravděpodobnost, že místo vysланého symbolu dostaneme jiný symbol. Vzhledem k symetrii, bude mít každý ze dvou zbývajících symbolů stejnou pravděpodobnost $\frac{p}{2}$, že bude přijat místo vysланého symbolu. S pravděpodobností $1-p$ chyba nenastane.

Obecně, bude-li kanál q-ární, tak každý symbol bude přijat správně s pravděpodobností $1-p$ a každý jiný symbol bude přijat se stejnou pravděpodobností $\frac{p}{q-1}$.

Příklad 1.15. Mějme binární opakovací kód délky 3. Předpokládejme, že pravděpodobnost chyby přenosu v symetrickém kanálu je $p = 0,01$ a předpokládejme, že přenášíme kódové slovo 111. Určíme s jakou pravděpodobností přijmeme správné kódové slovo, tj. že nastane bezchybný přenos, nebo přijmeme kódové slovo s jedinou chybou, kterou opravíme.
Kódové slovo 111 bez chyby přijmeme s pravděpodobností $(1-p)^3$. S jedinou chybou s pravděpodobností $3p(1-p)^2$. Celkem dostaneme správné kódové slovo s pravděpodobností $(1-p)^3+3p(1-p)^2 = (1+2p)(1-p)^2$. Pro $p = 0,01$ dostáváme hodnotu pravděpodobnosti $(1+2p)(1-p)^2 = 0,99702$, že kódové slovo bude přijato správně. ✓

Pravděpodobnost chyby kódu
Pravděpodobnost chyby kódu C je pravděpodobnost, že přijaté slovo bude dekódováno špatně, tj. nebude dekódováno jako kódové slovo, které bylo vysláno. Jedná se o doplněk pravděpodobnosti P, že přijaté slovo bude dekódováno správně. Značí se $P_{err}(C)$ a platí $P_{err}(C) = 1 - P$.

Příklad 1.16. V předchozím Příkladu 1.15. pro $p = 0,01$ je pravděpodobnost chyby $P_{err}(C) = 1 - (1 + 2p)(1-p)^2 = 0,00029702$. Znamená to, že přibližně 3 z 10 000 slov budou přijaty s chybou.

Všimněte si, že vzhledem k symetrii v uvedeném příkladu pravděpodobnost chyby kódu nezávisí na přenášené kódovém slovu. To v obecnosti nemusí být pravda, neboť může záviset na tom, jak moc se dané kódové slovo liší (jakou má Hammingovu vzdálenost) od ostatních kódových slov. Avšak v celé řadě kódů, například pro lineární kódy probírané v Kapitole 4. a 5. je pravděpodobnost chyby kódu nezávislá na konkrétní volbě kódového slova.

Dekódování nejbližším kódovým slovem
V symetrickém kanálu předpokládáme, že každá chyba je stejně pravděpodobná. Bude-li například vysláno slovo $\vec{x}$ a přijato bude slovo $\vec{y}$, tak v rušeném kanálu může být $\vec{x} \neq \vec{y}$. Slovo $\vec{y}$ bude dekódováno jako takové kódové slovo $\vec{x}'$, aby vzdálenost $\mathrm{dist}(\vec{x}', \vec{y})$ byla co nejmenší. Takovému dekódování se říká „dekódování nejbližším kódovým slovem“. V symetrickém q-árním kanálu s malou pravděpodobností chyby to znamená, že dekódované slovo $\vec{x}'$ současně odpovídá s největší pravděpodobností vyslanému slovu $\vec{x}$.

Příklad 1.17. Hammingova vzdálenost je jen jedna z možných metrik. Například pro telefonní čísla na tlačítkovém telefonu může být chyba způsobena stisknutím sousedního tlačítka, přičemž stisknout 6 místo 5 je mnohem pravděpodobnější než stisknout 9 místo 1.

Pro telefonní čísla vytáčená na starších telefonech s číselníkem bylo nejčastější chybou „nedotočit“ ciferník až k zarážce a tak vytočit chybnou cifru 3 místo 4 bylo mnohem pravděpodobnější než vytočit 4 místo 3.

V některých případech proto může mít smysl zavést i jiné míry vzdálenosti, než Hammingovu vzdálenost. V tomto textu však budeme pracovat převážně s Hammingovou vzdáleností.

Minimální vzdálenost kódu
Kvalitu kódu lze posuzovat více měřítky. Jedním z nejpřirozenějších je nejmenší vzdálenost mezi dvěma kódovými slovy.

Definice Minimální vzdálenost kódu
Minimální vzdálenost kódu C nad $(F_q)^n$ značíme $\mathrm{dist}(C)$ a je rovna

$$\mathrm{dist}(C) = \min\{\mathrm{dist}(\vec{x}, \vec{y}) : \vec{x}, \vec{y} \in C, \vec{x} \neq \vec{y}\}.$$

Minimální vzdálenost kódu určuje, jak dobrý je kód při detekci opravování chyb.

Příklad 1.18. Uveďme několik jednoduchých příkladů minimální vzdálenosti kódu.

1) Názvy ulic zmíněné v Příkladu 1.7. na straně 5 jsou příkladem kódu s malou minimální vzdáleností. Ve městě jsou ulice, které se jmenují Halasova ulice a Holasova ulice, které mají Hammingovu vzdálenost 1. Stačí záměna jediného písmena a dostaneme jiné platné kódové slovo. Takový kód není vhodný pro detekci chyb.
   Podobně ulice Bendova a Bendlova se liší jediným znakem navíc.
2) Telefonní čísla tvoří kód v množině slov $(F_{10})^9$, který má minimální vzdálenost 1 a neumí proto žádnou chybu detekovat, ani opravit. Například děkan naší fakulty má telefonní číslo 597 326 000 a jeho sekretářka má telefonní číslo 597 326 001.
3) Hláskovací abeceda je kódem s 42 kódovými slovy, která vyjadřují první písmena. Slova jsou volena tak, aby se všechna dostatečně lišila a aby i při silném rušení se dalo každé slovo dobře rozpoznat.
4) Kód C_1 má minimální vzdálenost $\text{dist}(C_1) = 1$.
5) Kód C_2 má minimální vzdálenost $\text{dist}(C_2) = 2$.
6) Kód C_3 má minimální vzdálenost $\text{dist}(C_3) = 3$.

Dekódování nejpravděpodobnějším kódovým slovem

Pokud jsou všechny chyby stejně pravděpodobné a současně pravděpodobnost chyby je malá, tak můžeme předpokládat, že nejpravděpodobnější je současně nejbližší ve smyslu Hammingovy vzdálenosti.

Tady chybí část textu, kterou je třeba doplnit.

Věta 1.2. Detekce a oprava chyb

Mějme kód C.

(i) Kód C s minimální vzdáleností $t+1$ umí detekovat až t chyb v jakémkoliv kódovém slově kódu C.
(ii) Kód C s minimální vzdáleností $2t+1$ umí opravit až t chyb v jakémkoliv kódovém slově kódu C.

Důkaz. Dokážeme obě části tvrzení.
(i) Postupujeme přímo. Předpokládejme, že minimální vzdálenost kódu C je $t+1$. Předpokládejme, že vysláno bylo kódové slovo $\vec{x}$ a pokud při přenosu nastane t nebo méně chyb, tak přijaté slovo nemůže být jiným kódovým slovem. To znamená, že může být detekováno až t libovolných chyb.
(ii) Opět postupujeme přímo. Předpokládejme, že minimální vzdálenost kódu C je $2t+1$. Předpokládejme, že vysláno bylo kódové slovo $\vec{x}$ a při přenosu nastalo nejvýše t chyb. Přijato bylo slovo $\vec{u}$, přičemž $\text{dist}(\vec{x},\vec{u}) \leq t$. Pro každé kódové slovo $\vec{x}'$ kódu C, $\vec{x}' \neq \vec{x}$, platí $\text{dist}(\vec{x}',\vec{u}) \geq t+1 > t$. Kdyby totiž $\text{dist}(\vec{x}',\vec{u}) \leq t$, tak z trojúhelníkové nerovnosti (vlastnost (iii) Lemmatu 1.1.) by plynulo $\text{dist}(\vec{x},\vec{x}') \leq \text{dist}(\vec{x},\vec{u}) + \text{dist}(\vec{x}',\vec{u}) \leq t + t < 2t+1$, což by byl spor s minimální vzdáleností kódu $\text{dist}(C) = 2t+1$. Proto vyslané kódové slovo $\vec{x}$ je nejbližší přijatému slovu $\vec{u}$ a kód C opraví až t chyb. □

Ihned dostáváme následující důsledek.

Důsledek 1.3. *Mějme kód C s minimální vzdáleností d. Potom kód C umí*

(i) detekovat až $d-1$ chyb,
(ii) opravit až $\lfloor (d-1/2) \rfloor$ chyb v libovolném kódovém slově.

Je dobré připomenout, že tvrzení Věty 1.2. a tvrzení Důsledku 1.3. vychází z předpokladu, že přenos probíhá symetrickým kanálem. Bez něj nemá smysl se spoléhat na dekódování nejbližším kódovým slovem.

(n, M, d)-kód

Délka kódu, počet slov kódu a minimální vzdálenost patří k nejdůležitějším parametrům, které daný kód charakterizují.

Definice Kód C délky n s M kódovými slovy a minimální vzdáleností d budeme nazývat (n, M, d)-*kód*.

Všimněte si, že definice nespecifikuje počet znaků abecedy. Ten zpravidla popíšeme pomocí přídavného jména, například binární $(2, 4, 1)$-kód, nebo q-ární (n, q, n)-kód. Z kontextu bude jasné, s jakou abecedou pracujeme. Ve většině případů budeme uvažovat binární kódy.

Příklad 1.19.

1) C_1 je binární $(2, 4, 1)$-kód.
2) C_2 je binární $(3, 4, 2)$-kód.
3) C_3 je binární $(5, 4, 3)$-kód.
4) Binární opakovací kód C_4 délky 4 je $(4, 2, 4)$-kód.

$$\text{kód } C_4 = \begin{cases} 0000 \\ 1111 \end{cases}$$

5) q-ární opakovací kód C délky n je (n, q, n)-kód.
6) Sonda Mariner 9 posílala černobílé fotografie Marsu prostřednictvím binárního $(32, 64, 16)$-kódu [H]. Každý pixel fotografie rozlišoval 64 odstínů, přičemž každý odstín byl zakódován pomocí některého z 64 kódových slov. Třebaže pro rozlišení 64 různých bitových slova stačí 6 bitů, samoopravný Reed-Mullerův kód pracoval s kódovými slovy délky 32 bitů (měl tedy 26 redundantních bitů).

Parametry binárního (n, M, d)-kódu nejsou nezávislé. Čím větší požadujeme větší počet slov, tím delší musí být kódová slova. Podobně, čím požadujeme větší minimální vzdálenost kódu, tím delší musí být kódová slova.

Příklad 1.20. Ukážeme, že binární $(4, M, 2)$-kód může mít nejvíce 8 slov.
Všech kódových slov délky 4 je $2^4 = 16$. Vybereme všechna kódová slova se sudým počtem 1, takových je právě polovina, osm. Dostaneme kód C.

$$\text{kód } C = \begin{cases} 0000 \\ 1100 \\ 1010 \\ 1001 \\ 0110 \\ 0101 \\ 0011 \\ 1111 \end{cases}$$

Ukážeme, že kód C má požadované vlastnosti. Žádná dvě kódová slova nejsou ve vzdálenosti 1, neboť kódové slovo ve vzdálenosti 1 od kódového slova se sudým počtem jedniček má lichý počet jedniček.

Dále ukážeme, že žádný kód nemůže mít více kódový slov. Ke každému kódovému slovu jednoznačně přiřadíme do dvojice kódové slovo, které se liší právě na prvním souřadnici. Takových dvojic jistě existuje $16/2 = 8$. Z každé dvojice může do kódu patřit nejvýše jedno slovo, proto kódových slov je nejvýše 8. ✓

Cvičení

1.3.1. Ukažte, že v kódu „dva z pěti“ mají každá dvě kódová slova sudou vzdálenost.

1.3.2. Mějme (n, M, d)-kód C. Sestavíme $(2n, M, d')$-kód C_k, který každý symbol vstupního slova obsahuje dvakrát za sebou. Takovému kódu říkáme koktavý. *Určete nejmenší vzdálenost kódu C_k délky $2n$.*

1.3.3. Zvolme libovolné binární kódové slovo $\vec{w}$ délky 4. Sestavte co největší binární $(4, M, 2)$-kód, který obsahuje kódové slovo $\vec{w}$.

1.3.4. Kolik nejvíce slov obsahuje binární $(3, M, 2)$-kód?

1.3.5. Kolik nejvíce slov obsahuje ternární $(3, M, 2)$-kód?

1.3.6. Mějme binární $(5, 4, 3)$-kód C_3 z Příkladu 1.11. Ukažte, že žádný jiný binární $(5, 4, 3)$-kód nemůže mít více slov.

Kapitola 2. Hlavní úloha teorie kódování

Dobrý (n, M, d)-kód má malé n, aby jej bylo možnost rychle přenášet, případně úsporně ukládat. Dále má velké M, aby bylo možno vyjádřit dostatečné množství zpráv a současně velké d, aby uměl rozpoznat, případně opravit velké množství chyb. Tyto požadavky jsou protichůdné. Proto se zpravidla jeden z parametrů optimalizuje dle požadavků ostatních dvou. Obvykle hledáme kód, který má co největší M, přičemž délka n a minimální vzdálenost d jsou pevně zvoleny.

Symbolem $A_q(n, d)$ značíme nejvyšší známou(!) hodnotu M, pro kterou existuje q-ární (n, M, d)-kód. Hodnota $A_q(n, d)$ je známa pro řadu parametrů n, d a q.

Je jasné, že minimální vzdálenost kódu d nabývá některé hodnoty z intervalu $[1, n]$. Následující věta ukazuje, že pro obě extrémní hodnoty d je hodnota $A_q(n, d)$ známa přesně a příslušné kódy umíme setrojit.

Věta 2.1. *Mějme q-ární (n, M, d)-kód. Pro největší počet slov kódu platí. Platí*

(i) $A_q(n, 1) = q^n$ *a*
(ii) $A_q(n, n) = q$.

Důkaz. Dokážeme obě části tvrzení.
(i) Pokud je minimální vzdálenost kódu $d = 1$, tak stačí, aby kódová slova byla různá. Proto největší $(n, M, 1)$-kód obsahuje všechna slova $(F_q)^n$ a platí $M = q^n$.
(ii) Mějme q-ární (n, M, d)-kód s minimální vzdáleností $d = n$. Každá dvě kódová slova se proto musí lišit v každé z n souřadnic. Pokud jedno kódové slovo má na první souřadnici symbol $s_1 \in F_q$, druhé kódové slovo musí mít na první souřadnici jiný symbol s_2, třetí kódové slovo opět jiný symbol s_3, atd. Proto počet kódových slov M je nejvýše q a platí $A_q(n, n) \leq q$. Na druhou stranu stačí vzít q-ární opakovací kód délky n se symboly z F_q a dostaneme (n, M, n)-kód s počtem slov $M = q$. Proto $A_q(n, n) = q$. □

Příklad 2.1. Jaká je nejmenší délka ternárního kódu, který má a) 10 b) 100 kódových slov?
a) Podle Věty 2.1. víme, že $A_q(n, 1) = q^n$. V našem případě musí platit $A_3(n, 1) = 3^n \geq 10$. Ihned vidíme, že $n \geq 3$.

Takový kód jistě existuje, neboť $(F_3)^3$ má dokonce 27 kódových slov.
b) Podobně si uvědomíme, že musí platit $A_3(n, 1) = 3^n \geq 100$. Ihned vidíme, že $n \geq 5$ (neboť $3^4 = 9^2 = 81$) a kód $(F_3)^5$ má dokonce 243 kódových slov. ✓

Příklad 2.2. Platí $A_2(5, 3) = 4$.
V Příkladu 1.11. jsme ukázali příklad $(5, 4, 3)$-kódu. Přirozeně vzniká otázka, zda existuje kód délky $n = 5$ s minimální vzdáleností $d = 3$ a větším počtem slov. Existuje $\binom{32}{5} = 201\,376$ možných binárních kódů délky 5 s pěti slovy. Hrubou silou prověřit, že ani jedna pětice nemá minimální vzdálenost 3 je pracné.

V Příkladu 2.3. ukážeme, že takový kód s minimální vzdáleností 3 a více než čtyřmi kódovými slovy neexistuje, a proto $A_2(5, 3) \leq 4$. ✓

V Příkladu 2.3. dokážeme dokonce silnější tvrzení: ukážeme, že pokud se omezíme na „ekvivalentní kódy“, tak takový kód existuje jediný.

2.1. Ekvivalence kódů

Připomeňme, že permutace konečné množiny A je bijektivní zobrazení $\sigma : A \to A$. Permutace můžeme zapisovat pomocí dvouřádkové matice. Například permutaci σ, která každému číslo z množiny $[0, 8]$ přiřadí dvojnásobek modulo 9, je

$$\sigma = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 0 & 2 & 4 & 6 & 8 & 1 & 3 & 5 & 7 \end{pmatrix}.$$

Stejnou permutaci σ můžeme zapsat pomocí cyklů, například

$$\sigma = (0)(1\,2\,4\,8\,7\,5)(3\,6).$$

Permutace prvků abecedy na některé souřadnici kódu znamená, že pro všechna kódová slova kódu C provedeme na pevně zvolené souřadnici záměnu. Prvek a nahradíme prvkem $\sigma(a)$. Podobně můžeme pouze změnit pořadí souřadnic každého kódového slova. Z kódu můžeme dostat jiný kód, který však bude mít stejnou

délku, stejný počet kódových slov i stejnou minimální vzdálenost. Toto pozorování zobecníme v následující větě.

Definice Ekvivalence kódů
Mějme dva q-ární kódy C, C' stejné délky se stejným počtem slov. Řekneme, že kódy C a C' jsou *ekvivalentní*, jestliže kód C' může být získán z kódu C pomocí dvou typů operací:

(i) stejnou permutací souřadnic $a_1, a_2, \ldots, a_n$ každého slova kódu,
(ii) stejnou permutací symbolů na pevně zvolené souřadnici každého slova kódu.

Pokud si představíme, že všechna kódová slova uložíme do matice B typu $M \times n$ tak, že každý řádek odpovídá jednomu kódovému slovu, tak operace (i) odpovídá permutaci sloupců matice B a operace (ii) odpovídá permutaci symbolů v pevně zvoleném sloupci matice B. Zmíníme ještě, že permutace řádků matice B odpovídá jen změně pořadí kódových slov.

Příklad 2.3. Mějme binární kód C_5.

$$\text{kód } C_5 = \begin{cases} 10000 \\ 11110 \\ 00011 \\ 01101 \end{cases}$$

Ukážeme, že C_5 je ekvivalentní kódu C_3 z Příkladu 1.11.
Stačí prohodit třetí a čtvrtý sloupce a v prvním sloupci udělat (jedinou možnou) permutaci symbolů. ✓

Je zřejmé, že každé dva ekvivalentní kódy mají stejnou délku, stejný počet slov. Snadno ukážeme (Cvičení 2.1.3.), že mají i stejnou minimální vzdálenost. Všechny ekvivalentní (n, M, d)-kódy proto detekují stejný počet chyb a opraví stejný počet chyb. Všechny ekvivalentní q-ární kódy můžeme považovat za rovnocenné vzhledem k detekci a opravě chyb při přenosu v symetrickém q-árním kanálu. Pokud by však přenosový kanál nebyl symetrický, tak to pravda být nemusí. Záměnou symbolů můžeme dostat jinou pravděpodobnost chyb.

Příklad? *Tady chybí část textu, kterou je třeba doplnit.*

Příklad 2.4. Kód C_2 z Příkladu 1.10. a opakovací kód délky 3 nejsou ekvivalentní, neboť nemají stejný počet slov ani nemají stejnou vzdálenost.

Následující příklad ukazuje, že i když dva kódy mají stejnou abecedou, stejnou délkou, stejným počtem slov a stejnou minimální vzdáleností, tak nemusí být ekvivalentní.

Příklad 2.5. Mějme dva kódy C a C'.

$$\text{kód } C = \begin{cases} 000 \\ 001 \\ 010 \\ 011 \end{cases} \qquad \text{kód } C' = \begin{cases} 000 \\ 001 \\ 110 \\ 111 \end{cases}$$

Ukážeme, že kódy C a C' nejsou ekvivalentní.
Oba kódy C i C' jsou binární kódy délky $n = 3$ s minimální vzdáleností $\operatorname{dist} C = \operatorname{dist} C' = 1$. Kód C má pouze dvě souřadnice, které obsahují různé symboly. Žádnou permutací symbolů na první souřadnici žádnou permutací sloupců nemůžeme zajistit, aby se na každé ze tří souřadnic vyskytovaly oba symboly abecedy. Analogicky bychom mohli argumentovat i naopak, že každá permutace symbolů i sloupců kódu C' zachová oba symboly na každé souřadnici. Proto kódy C a C' nejsou ekvivalentní. ✓

Otázky:

- Kolik existuje binárních kódů délky 5 s $M = 2$?
- Jsou všechny takové kódy ekvivalentní?
- Kolik existuje binárních kódů, které jsou ekvivalentní s binárním opakovacím kódem délky 5?
- Kolik existuje ekvivalentních binárních $(5, 4, 1)$-kódů?

Kódové slovo $\vec{0}$
K danému kódu vždy můžeme najít ekvivalentní kód, který obsahuje některé pevně zvolené slovo. Zpravidla požadujeme, aby kód obsahoval slovo $00\ldots0$.

Lemma 2.2. *Každý q-ární (n, M, d)-kód s abecedou $F_q = \{0, 1, \ldots, q-1\}$ je ekvivalentní nějakému q-árnímu (n, M, d)-kódu, který obsahuje kódové slovo $00\ldots0$.*

Důkaz. Tvrzení ukážeme přímo. Mějme q-ární (n, M, d)-kód s abecedou $F_q = \{0, 1, \ldots, q-1\}$ a zvolme libovolné kódové slovo $\vec{x} = x_1 x_2 \ldots x_n$. Pro každé $i = 1, 2, \ldots, n$ buď ponecháme symboly v příslušné souřadnici stejné, pokud $x_i = 0$. V opačném případě provedeme permutaci symbolů

$$\begin{pmatrix} 0 & x_i & j \\ \downarrow & \downarrow & \downarrow \\ x_i & 0 & j \end{pmatrix} \text{ pro každé } j \neq 0,\ j \neq x_i .$$

□

Příklad 2.6. Navážeme na Příklad 2.3. Ukážeme, že každý binární $(5, 4, 3)$-kód C je ekvivalentní kódu C_3 z Příkladu 1.11.

Mějme libovolný binární $(5, 4, 3)$-kód C. Podle Lemmatu 2.2. víme, že C je ekvivalentní s kódem C', který obsahuje kódové slovo 00000. Kód C' neobsahuje žádné kódové slovo, které obsahuje pouze jednu nebo dvě jedničky, neboť dvě taková slova se liší na nejvýše dvou souřadnicích a neplatilo by $\operatorname{dist}(C') = 3$.

Dále kód C' obsahuje nejvýše jedno kódové slovo, které obsahuje 4 nebo 5 jedniček, neboť dvě taková slova se liší na nejvýše dvou souřadnicích a neplatilo by $\operatorname{dist}(C') = 3$.

A konečně, protože $M \geq 4$, musí kód C' obsahovat alespoň dvě slova, která obsahují právě tři jedničky. Permutací souřadnic (operace (ii) ve větě 2.1.) dostaneme, že kód C' obsahuje slova

00000
11100
00111

Jiné kódové slovo se třemi jedničkami sdílí alespoň tři symboly s každým ze dvou posledních slov. Jediné slovo, které se liší ode všech tří slov v alespoň třech souřadnicích je slovo 11011. Tím jsme ukázali, nejen, že $A_2(5, 3) = 4$, ale také, že až na ekvivalenci existuje jediný takový kód. ✓

Velikosti binárních kódů

Ve zbývající části podkapitoly se omezíme na binární kódy. Pečlivým rozborem pro délky $n \leq 20$ a netriviální vzdálenosti $d \leq 11$ se podařilo pro binární kódy sestavit Tabulku 2.1. Tabulka je převzata ze https://www.win.tue.nl/~aeb/codes/binary-1.html

n	$d = 3$	$d = 5$	$d = 7$	$d = 9$	$d = 11$
5	4	2	1	1	1
6	8	2	1	1	1
7	16	2	2	1	1
8	20	4	2	1	1
9	40	6	2	2	1
10	72	12	2	2	1
11	144	24	4	2	2
12	256	32	4	2	2
13	512	64	8	2	2
14	1 024	128	16	4	2
15	2 048	256	32	4	2
16	2 816–3 276	258–340	36	6	2
17	5 632–6 552	512–673	64	10	4
18	10 496–13 104	1 024–1 237	128	20	4
19	20 480–26 168	2 048–2 279	256	40	6
20	40 960–43 688	2 560–4 096	512	42–47	8

Tabulka 2.1.: *Tabulka hodnot* $A_2(n, d)$.

Například první hodnotu tabulky $A_2(5,3) = 4$ jsme určili v Příkladu 2.6. Poslední tři hodnoty v prvním řádku $A_2(5,7) = 1$, $A_2(5,9) = 1$, $A_2(5,9) = 1$ říkají, že takový kód nemá smysl. Kód s jediným kódovým slovem nemůže při vysílání jediného kódového slova přenášet žádnou užitečnou informaci, je schopen pouze rozlišit, zda vysílá, nebo nikoliv. Rozsah 2 816–3 276 v řádku $n = 16$ znamená, že velikost $A_2(16,3)$ největšího binárního $(16, M, 3)$-kódu není známa, ale jsou známy odhady. Takový kód má alespoň 2 816 slov, ale ne více než 3 276 slov.

Sloupec pro $d = 1$ v Tabulce 2.1. není nutno uvádět, protože z Věty 3.5. (i) víme, že $A_2(n,1) = 2^n$. Z Věty 3.5. (ii) pak vyplývá, že v řádku n a sloupci $d = n$ bude vždy hodnota 2. Kdybychom sestavovali podobnou tabulku pro jiné než binární kódy, tak $A_q(n,n) = q$. Pro hodnoty $d > n$ musí tabulka vždy obsahovat hodnotu 1.

Všimněte si, že v tabulce jsou uvedeny pouze hodnoty pro d liché. V další podkapitole ukážeme, že to není na úkor obecnosti, ale že pro sudé hodnoty n dostaneme analogické výsledky jako důsledek.

Některé z dalších hodnot uvedených v tabulce dokážeme v průběhu textu.

Cvičení

2.1.1.$^{\heartsuit}$ *Sestavte následující binární kódy: a)* $(5,2,5)$*-kód, b)* $(4,16,1)$*-kód*

2.1.2. Najděte dva kódy stejné délky, stejné velikosti, se stejnou minimální vzdáleností, které nejsou ekvivalentní.

2.1.3. Ukažte, že operace (i) a (ii) z definice ekvivalence kódů na straně 11 nezmění minimální vzdálenost kódu.

2.1.4. Ukažte, že ternární kód

$$C = \begin{cases} 012 \\ 120 \\ 201 \end{cases}$$

je ekvivalentní s ternárním opakovacím kódem délky 3*.*

2.1.5. Najděte dva kódy se stejným počtem slov a se stejnou co nejkratší délkou slov, které nejsou ekvivalentní.

2.1.6. Najděte dva kódy se stejnou abecedou, se stejnou délkou $n = 5$*, se stejným počtem slov a se stejnou minimální vzdáleností* 2*, které nejsou ekvivalentní.*

2.1.7. Najděte dva binární kódy se stejným počtem slov alespoň 4 *a s minimální vzdáleností* 2*, které nejsou ekvivalentní tak, aby měly co nejmenší délku. Ukažte, že kratší takové kódy neexistují.*

2.2. Binární kódy

V této podkapitole se omezíme na binární kódy, budeme pracovat s abecedou $F_2 = \{0,1\}$ a binárními vektory z množiny $(F_2)^n$. Nejprve zavedeme pojem váhy binárního slova.

Definice **Váha binárního slova**
Mějme slovo $\vec{x} \in (F_2)^n$. *Váha binárního slova* je počet jedniček, která se ve slově vyskytují. Váhu značíme $\mathrm{w}(x)$.

Příklad 2.7. Uveďme několik příkladů vah binárních slov.

1) $\mathrm{w}(000) = 0$,
2) $\mathrm{w}(111) = 3$,
3) $\mathrm{w}(01010) = 2$.

Operace s binárními kódovými slovy
Dále zavedeme dvě operace s binárními kódovými slovy.

Definice Mějme dvě binární slova $\vec{x}, \vec{y} \in (F_2)^n$, kde $\vec{x} = x_1 x_2 \dots x_n$ a $\vec{y} = y_1 y_2 \dots y_n$. *Součet* dvou slov definujeme předpisem

$$\vec{x} + \vec{y} = (x_1 + y_1, x_2 + y_2, \dots, x_n + y_n).$$

Průnik dvou slov definujeme předpisem

$$\vec{x} \cap \vec{y} = (x_1 y_1, x_2 y_2, \dots, x_n y_n).$$

$+$	0	1
0	0	1
1	1	0

$\cdot$	0	1
0	0	0
1	0	1

Tabulka 2.2.: *Cayleyho tabulky operací v binárním tělese* $(F_2, +, \cdot)$.

Operace jsou definovány po složkách jako obvyklé binární operace sčítání a násobení bez přenášení.

Operace můžeme snadno popsat Cayleyho tabulkami operací v tělese $(F_2, +, \cdot)$.

Příklad 2.8. Uveďme několik příkladů operací s binárními slovy.

1) $1010 + 0101 = 11113$,
2) $1010 \cap 0101 = 0000$,
3) $11100 + 00111 = 11011$,
4) $11100 \cap 00111 = 00100$.

Následující lemma ukazuje, že platí rovnost mezi vzdáleností dvou slov a váhou jejich součtu

Lemma 2.3. *Mějme $\vec{x}, \vec{y} \in (F_2)^n$, tj. binární slova délky n. Platí* $\mathrm{dist}(\vec{x}, \vec{y}) = \mathrm{w}(\vec{x} + \vec{y})$.

Důkaz. Součet dvou kódových slov $\vec{x} + \vec{y}$ nabývá hodnoty 1 v té složce, kde se kódová slova liší a 0 jinak. Proto počet jedniček součtu $\vec{x} + \vec{y}$ udává nejen váhu součtu, ale současně i Hammingovu vzdálenost slov $\vec{x}$, $\vec{y}$. □

Další lemma ukazuje vztah mezi vzdáleností dvou slov, jejich vahou a vahou jejich součinu.

Lemma 2.4. *Mějme $\vec{x}, \vec{y} \in (F_2)^n$, tj. binární slova délky n. Platí* $\mathrm{dist}(\vec{x}, \vec{y}) = \mathrm{w}(\vec{x}) + \mathrm{w}(\vec{y}) - 2\,\mathrm{w}(\vec{x} \cap \vec{y})$.

Důkaz. Podle Lemmatu 2.3. víme, že $\mathrm{dist}(\vec{x}, \vec{y}) = \mathrm{w}(\vec{x} + \vec{y})$. Avšak váha součtu je dána váhou jednotlivých slov kromě souřadnic, které jsou započítané dvakrát. V binárním kódu to jsou však právě souřadnice, kde obě slova $\vec{x}$, $\vec{y}$ mají hodnotu 1, který je právě $\vec{x} \cap \vec{y}$. Tyto souřadnice musíme dvakrát odečíst, proto $\mathrm{dist}(\vec{x}, \vec{y}) = \mathrm{w}(\vec{x}) + \mathrm{w}(\vec{y}) - 2\,\mathrm{w}(\vec{x} \cap \vec{y})$. □

Spojením tvrzení obou lemmat ihned vidíme, že váha součtu je dána součtem vah mínus dvojnásobek váhy průniku.

$$\mathrm{w}(\vec{x} + \vec{y}) = \mathrm{w}(\vec{x}) + \mathrm{w}(\vec{y}) - 2\,\mathrm{w}(\vec{x} \cap \vec{y})$$

Nyní můžeme ukázat jedno z nejdůležitějších tvrzení týkající se existence kódů se sudou a lichou minimální vzdáleností.

Věta 2.5. *Mějme liché číslo d. Binární (n, M, d)-kód existuje právě tehdy, když existuje $(n+1, M, d+1)$-kód.*

Důkaz. Jedná se o ekvivalenci, ukážeme obě implikace.
„⇒“ Tvrzení ukážeme přímo. Mějme (n, M, d)-kód C, kde d je liché číslo. Sestavíme $(n+1, M, d+1)$-kód C' tak, že ke každému kódovému slovu kódu C „přidáme součtový paritní bit“. Mějme kódové slovo $\vec{x} \in C$, kde $\vec{x} = x_1 x_2 \dots x_n$. Sestavíme kódové slovo

$$\vec{x}' = \begin{cases} x_1 x_2 \dots x_n 0 & \text{jestliže } \mathrm{w}(\vec{x}) \text{ je sudá,} \\ x_1 x_2 \dots x_n 1 & \text{jestliže } \mathrm{w}(\vec{x}) \text{ je lichá.} \end{cases}$$

(Ekvivalentně bychom mohli definovat $\vec{x}' = x_1 x_2, \dots x_n x_{n+1}$, kde $x_{n+1} = \sum_{i=1}^{n} x_i$ a sčítání probíhá modulo 2.) Dostaneme tak kód C', kde váha každého kódového slova je sudá. Podle Lemmatu 2.4. je vzdálenost každých dvou slov sudá, neboť je součtem nebo rozdílem sudých čísel. Minimální vzdálenost kódu C' je proto také sudé číslo. Je zřejmé, že $d \le \mathrm{dist}(C') \le d + 1$, neboť přidáním paritního součtu vzdálenost nemůžeme snížit ani zvýšit o více než o 1. Protože číslo d je liché, tak musí nastat $\mathrm{dist}(C') = d + 1$. Tím je implikace dokázána.
„⇐“ Důkaz opačné implikace je jednodušší. Odebráním jednoho bitu (znaku) ze všech slov kódu snížíme minimální vzdálenost nejvýše o 1. Protože $\mathrm{dist}(C') = d+1$, tak v kódu C' existují dvě slova $\vec{x}$, $\vec{y}$ ve vzdálenosti $d + 1$. Ze všech kódových slov kódu C' odebereme některý bit (souřadnici), ve kterém se slova $\vec{x}$, $\vec{y}$ liší. Tak dostaneme binární (n, M, d)-kód. Tím je tvrzení dokázáno. □

Postupu, kdy ke každému kódovému slovu přidáme paritní bit, se říká *rozšíření kódu*.

Otázka: Platí tvrzení Věty 2.5. i pro jiné než binární kódy?

Z tvrzení Věty 2.5. ihned dostáváme, že při hledání hodnot $A_2(n,d)$ se stačí omezit pouze na liché nebo pouze na sudé hodnoty d. Proto se při konstrukci binárních kódů soustředíme převáženě na kódy s lichou délkou n.

Důsledek 2.6. *Jestliže d je liché číslo, tak $A_2(n+1,d+1) = A_2(n,d)$. A pokud d je sudé číslo, tak $A_2(n,d) = A_2(n-1,d-1)$.*

Příklad 2.9. Mějme $(5,4,3)$-kód C_3 z Příkladu 1.11. Sestavíme $(6,4,4)$-kód C_3' přidáním paritního bitu. Dostaneme kód C_3'.

$$\text{kód } C_3' = \begin{cases} 000000 \\ 011101 \\ 101011 \\ 110110 \end{cases}$$

Kód C_3' stále umí (podle Důsledku 1.3.) detekovat dvě chyby a opravit jedinou chybu. ✓

Pokud bychom však chtěli ukázat, že $(6,M,4)$-kód má nejvíce M slov, kde $M \leq 4$, tak podobným rozborem jako v Příkladu 2.6. by diskuze případů byla mnohem složitější. V následující podkapitole 2.3. ukážeme, že existuje další nástroje pro horní odhad velikosti kódu.

Paritní bit

Postup ukázaný v důkazu Věty 2.5. se nazývá *přidání paritního součtu*. Jestliže máme nějaký binární (n,M,d)-kód C, tak přidání paritního součtu vždy

1) zvýší délku kódu o 1,
2) zachová počet kódových slov,
3) zachová nebo zvýší minimální vzdálenost kódu (pro liché d vzdálenost zvýší).

Zkrácený kód

V předchozím textu jsme ukázali, jak daný kód s lichou vzdáleností prodloužit a minimální vzdálenost zvýšit. Můžeme však postupovat i opačně a kód zkrátit. Přitom nemusí nutně dojít ke snížení nejnižší vzdálenosti kódu. Je možno místo toho snížit počet kódových slov.

Mějme nějaký binární (n,M,d)-kód C. *Zkrácený kód* dostaneme z kódu C tak, že vybereme všechna kódová slova, která na pevně zvolené souřadnici mají stejný symbol. Bez újmy na obecnosti má množina těchto slov A alespoň polovina všech slov kódu C, tj. platí $|A| \geq M/2$. Nyní tuto souřadnici ze všech kódových slov množině A odebereme a dostaneme slov množinu A', která tvoří zkrácený $(n-1,M',d')$-kód C'. Jistě platí $M' = |A| \geq M/2$ a dále kódová slova kódu C' stejnou minimální vzdálenost, jako měla kódová slova v množině A kódu C. Platí proto $d' \geq d$.

Jestliže máme nějaký binární (n,M,d)-kód C, tak popsané zkrácení kódu vždy

1) sníží délku kódu o 1,
2) sníží počet kódových slov (výsledný kód má polovinu nebo více kódových slov),
3) zachová minimální vzdálenost kódu, ve výjimečných případech je minimální vzdálenost kódu vyšší.

Cvičení

2.2.1. Sestavte následující kódy, pokud existují. a) binární $(2,4,1)$-kód, b) binární $(3,4,2)$-kód, c) binární $(7,2,7)$-kód, d) binární $(5,3,4)$-kód. Pokud takový kód neexistuje, vysvětlete proč.

2.2.2. Sestavte $(7,8,3)$-kód. Pokud takový kód neexistuje, vysvětlete proč.

2.2.3. Mějme binární (n,M,d)-kód C. Ukažte, že existuje binární $(n-1,M',d)$-kód C', pro který platí $M' \geq M/2$.

2.2.4. Najdete příklad binárního (n,M,d)-kódu C tak, aby pro zkrácený $(n-1,M',d')$-kód C' platilo $M' > M/2$ a současně $d' > d$? Pokud takový kód neexistuje, dokažte to.

2.2.5. Mějme množinu B všech binárních vektorů délky n. Ukažte, že počet vektorů v B s lichou váhou i počet vektorů v B se sudou váhou je stejný, tj. 2^{n-1}.

2.2.6. Ukažte, že Věta 2.5. nemusí platit pro (n,M,d)-kódy, které nejsou binární, tj. že přidáním kontrolního součtu do kódu s lichou minimální vzdáleností, kdy kontrolní součet získáme jako součet souřadnic modulo q, nemusí zvýšit minimální vzdálenost.

2.3. Sféra kódového slova

Připomeňme symbol pro kombinační číslo $\binom{n}{k}$ a jeho význam: kombinační číslo udává počet k prvkových podmnožin n prvkové množiny (n různých prvků). Víme, že platí

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}. \tag{1}$$

Řada softwarů i kalkulaček má přímo funkci pro výpočet kombinačního čísla. Pro ruční počítání spíš než (1) je výhodnější následující vztah.

$$\binom{n}{k} = \frac{n(n-1)\cdots(n-k+1)}{(n-k)!} \tag{2}$$

Kombinačnímu číslo se také říká *binomický koeficient* podle binomické věty, kde kombinační čísla tvoří koeficienty příslušného polynomu.

$$(a+b)^n = \binom{n}{0}a^0b^n + \binom{n}{1}a^1b^{n-1} + \cdots + \binom{n}{n}a^nb^0$$

Dále připomeňme, že pro kombinační čísla platí dobře známé vztahy

Lemma 2.7. *Mějme nezáporná přirozená čísla n, k, kde $k \leq n$. Platí*

$$\binom{n}{k} = \binom{n}{n-k}, \qquad \binom{n}{k} + \binom{n}{k+1} = \binom{n+1}{k+1}.$$

Příklad 2.10. Kolik kódových slov by měl a) binární kód „tři z pěti“, b) binární kód „tři z šesti“, c) ternární kód „tři ze sedmi“?

a) Kód „tři z pěti“ by měl $\binom{5}{3} = 10$ kódových slov. Každé by obsahovalo právě tři jedničky a dvě nuly, stejně jako kód „dva z pěti“ obsahuje tři nuly a dvě jedničky.

b) Kód „tři z šesti“ by měl $\binom{6}{3} = 20$ kódových slov. Každé by obsahovalo právě tři jedničky a tři nuly.

c) Kód „tři ze sedmi“ by měl $\binom{7}{3} \cdot 2^4 = 35 \cdot 16 = 560$ kódových slov. Každé by obsahovalo právě tři jedničky a na zbývajících čtyřech souřadnicích jiné hodnoty 0 nebo 2. ✓

Sféra slova

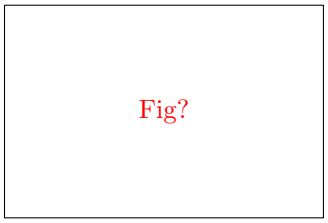
Nyní se opět podíváme na slova v množině $(F_q)^n$. Formálně pro dané slovo $\vec{u}$ popíšeme množinu „blízkých slov“ a ukážeme, jak určit jejich počet. Sféru definujeme pro libovolné slovo, zajímavé však jsou zejména sféry kódových slov.

Definice Sféra

Mějme libovolné slovo $\vec{u} \in (F_q)^n$ a libovolné celé číslo $r \geq 0$. *Sféra* o poloměru r se středem v $\vec{u}$ je množina

$$S(\vec{u}, r) = \{\vec{v} \in (F_q)^n : \mathrm{dist}(\vec{u}, \vec{v}) \leq r\}.$$

Sféra pevně zvoleného slova $\vec{u}$ je tedy množina všech slov množiny $(F_q)^n$, která jsou ve vzdálenosti nejvýše r od slova $\vec{u}$.



Obrázek 2.1.: *Sféra kódového slova, slova ve vzdálenosti menší než poloměr sféry.*

Tady chybí část textu, kterou je třeba doplnit.

Terminologie i obrázek vycházejí z jakési geometrické analogie se vzdáleností v euklidovském prostoru. Kódové slovo chápeme jako bod (konečného prostoru $(F_q)^n$ a sféra slova je množina slov (bodů), které jsou ve vzdálenosti nejvýše poloměru r sféry, přičemž vzdálenost slov (bodů) neměříme euklidovskou metrikou, ale Hammingovou vzdáleností.

Příklad 2.11. Mějme binární opakovací $(5, 2, 5)$-kód. Sféra $S(00000, 1)$ obsahuje kromě kódového slova 00000 ještě všechna slova, která obsahují jednu jedničku. Takových slov je celkem 6. Sféra $S(00000, 2)$ obsahuje všechna kódové slovo 00000, dále slova, která obsahují jednu nebo dvě jedničky. Takových slov je celkem 1+5+10=16.

Poznámka 2.1. Sféry o poloměru t slov kódu C s minimální vzdáleností $\operatorname{dist}(C) \geq 2t+1$ jsou disjunktní

Všechna slova $\vec{y}$ uvnitř sféry $S(\vec{x}, t)$ jsou slova $\vec{x}'$ s Hammingovou vzdáleností nejvýše t. Tato slova mohou dekódována jako slovo $\vec{x}$, pokud žádné dvě sféry $S(\vec{x}, t)$ a $S(\vec{y}, t)$ kódových slov $\vec{x}$, $\vec{y}$ mají prázdný průnik.

Naopak, pokud by dvě slova měla neprázdný průnik sfér, tak přijeté slovo nemusíme umět správě dekódovat, pokud nastane alespoň t chyb.

Objem sféry

Následující lemma ukazuje, kolik slov patří do sféry o poloměru r libovolně zvoleného slova.

Lemma 2.8. *Sféra o poloměru r libovolného slova množiny $(F_q)^n$ obsahuje právě*

$$\binom{n}{0} + \binom{n}{1}(q-1) + \binom{n}{2}(q-1)^2 + \cdots + \binom{n}{r}(q-1)^r$$

slov.

Důkaz. Mějme pevně zvolené slovo $\vec{u}$ délky n z množiny $(F_q)^n$. Takové slovo ve vzdálenosti 0 od slovo $\vec{u}$ je jediné, tento počet můžeme zapsat jako $\binom{n}{0} = 1$. Všechna slova ve vzdálenosti r se liší v právě r souřadnicích. Množinu q souřadnic můžeme vybrat právě $\binom{n}{r}$ způsoby. Každá souřadnice může obsahovat $q-1$ jiných symbolů, než který obsahuje slovo $\vec{u}$. Ve vzdálenosti q se proto nachází právě $\binom{n}{r}(q-1)^r$ slov. Sečteme počet slov ve vzdálenosti $0, 1, \ldots, q$ od $\vec{u}$ a dostaneme celkem

$$\binom{n}{0} + \binom{n}{1}(q-1) + \binom{n}{2}(q-1)^2 + \cdots + \binom{n}{r}(q-1)^r$$

slov. □

Odtud dostáváme (jedno z více možných) omezení počtu slov, která může obsahovat kód délky n s minimální vzdáleností d obsahovat. Všimněte si, že se omezíme na liché minimální vzdálenosti kódu.

Věta 2.9. Hammingova hranice

Pro každý q-ární $(n, M, 2t+1)$-kód platí

$$M\left(\binom{n}{0} + \binom{n}{1}(q-1) + \binom{n}{2}(q-1)^2 + \cdots + \binom{n}{t}(q-1)^t\right) \leq q^n. \tag{3}$$

Důkaz. Kód má M slov. Sféra každého slova obsahuje

$$\left(\binom{n}{0} + \binom{n}{1}(q-1) + \binom{n}{2}(q-1)^2 + \cdots + \binom{n}{t}(q-1)^t\right)$$

slov.

Žádné dvě sféry o poloměry t nemohou mít podle Poznámky 2.1. společné slovo, proto celkový počet slov $(F_q)^n$ musí být alespoň tak velký, jako je počet slov ve všech sférách.

$$M\left(\binom{n}{0} + \binom{n}{1}(q-1) + \binom{n}{2}(q-1)^2 + \cdots + \binom{n}{t}(q-1)^t\right) \leq q^n$$

□

Hammingova hranice (3) dává poměrně jednoduchý a přirozený horní odhad na velikost kódu $A_q(n,d)$. Z nerovnosti (3) například můžeme pro $n=5$, $d=3$ a $q=2$ odvodit

$$\begin{aligned} M\,(1+5) &\leq 2^5 \\ 6M &\leq 32 \\ M &\leq 5. \end{aligned}$$

To znamená, že žádný binární $(5,M,3)$-kód nemůže mít více než 5 slov. Na druhou stranu z nerovnosti (3) neplyne, že takový kód musí existovat. Vždyť v Příkladu 2.6. jsme ukázalo, že pro binární $(5,M,3)$-kód musí platit $M \leq 4$.

Cvičení

2.3.1.$^{\heartsuit}$ *Jaký odhad dává Hammingova hranice (3) pro počet slov binárního opakovacího $(5,M,5)$-kódu?*

2.3.2.$^{\heartsuit}$ *Jaký odhad dává Hammingova hranice pro počet slov q-árního opakovacího $(5,M,5)$-kódu?*

2.4. Perfektní kódy

Kód může obsahovat více slov, než je obsaženo ve sférách jednotlivých slov. Pokud v nerovnosti (3) nastává rovnost takovému kódu říkám perfektní.

Definice Perfektní kód
Mějme (n,M,d)-kód, pro který v nerovnosti (3) nastává rovnost, se nazývá *perfektní* kód.

V perfektním kódu, který opraví t chyb, se nachází *všechna* slova uvnitř sfér kódových slov o poloměru t. Každé slovo prostoru $(F_q)^t$ se tak nachází ve vzdálenosti nejvýše t od nějakého kódového slova.

Příklad 2.12. Uveďme několik jednoduchých příkladů perfektních kódů.

1) Libovolný binární $(n,2,n)$-kód pro liché n je perfektní kód.
2) Libovolný q-ární $(n,2,n)$-kód pro liché n je perfektní kód.

Oba kódy z Příkladu 2.12. jsou *triviální* perfektní kódy.

Příklad 2.13. Uveďme několik jednoduchých příkladů kódů, které nejsou perfektní.

1) Libovolný binární $(n,2,n)$-kód pro sudé n není perfektní, neboť existují slova, která jsou stejně vzdálena od dvou (nebo více) kódových slov. Například slovo 0011 v binárním opakovacím $(4,2,4)$-kódu nespadá do žádné sféry s poloměrem 2, ale spadá do obou sfér s poloměrem 2.
2) Libovolný q-ární $(n,2,n)$-kód pro sudé n není perfektní kód.

Hledání perfektních kódů bylo jednou z velkých výzev teorie kódování. Otázka existence a konstrukce perfektních kódů pro řády a vzdálenosti, které takovou konstrukci umožňují, byla postupně zodpovězena. Další perfektní kódy ukážeme v následující kapitole a zejména v Kapitole ??, kde otázku existence a konstrukce zodpovíme.

Příklad 2.14. Ověříme, zda kód C_3 z Příkladu 1.11. je perfektní.
Kód C_3 je binární $(5,4,3)$-kód, platí tedy $n=5$, $M=4$, $d=3$ a $q=2$. Protože $d=3=2t+1$, tak spočítáme, kolik slov patří do sféry o poloměru $t=1$ každého kódového slova. Dostaneme

$$\binom{n}{0}+\binom{n}{1}(q-1)^t = \binom{5}{0}+\binom{5}{1}(2-1) = 1+5 = 6.$$

V Hammingově nerovnosti dostáváme

$$\begin{aligned} M\left(\binom{5}{0}+\binom{5}{1}(q-1)\right) &\leq q^5 \\ 4\,(1+5(2-1)) &\leq 2^5 \\ 4\cdot 6 &\leq 32 \\ 24 &< 32. \end{aligned}$$

To znamená, že binární $(5, 4, 3)$-kód C_3 není perfektní.

Vskutku, například pro slovo 01110 platí $\mathrm{dist}(11100, 01110) = 2$, stejně tak platí $\mathrm{dist}(00111, 01110) = 2$. Dále platí a $\mathrm{dist}(11011, 01110) = 3$ a $\mathrm{dist}(00000, 01110) = 3$. Pro slovo 01110 nelze rozhodnout, které kódové slovo je nejbližší, protože nespadá do sféry o poloměru 1 žádného kódového slova. ✓

Cvičení

2.4.1. Určete, zda je kód C_3' z Příkladu 2.9. perfektní.

Kapitola 3. Blokové designy, konečná tělesa a vektorové prostory

V kapitole zavedeme nejprve nový pojem: pojem kombinatorického designu. Dále v druhé části připomeneme některé vlastnosti konečných těles a ve třetí části připomeneme vybrané vlastnosti vektorových prostorů.

3.1. Blokový design

Kombinatorický design
Důležitým matematickým objektem jsou systémy podmnožin, pro které máme předepsány vlastnosti, jako velikost podmnožin, počet společných prvků a počet zastoupení prvků nebo dvojic prvků. Takovým objektům se říká kombinatorické designy (čti „dizajny“).

Definice Kombinatorický design
Mějme konečnou nosnou množinu V, která obsahuje v *prvků* (někdy zvaných *variety*). Označme $\mathcal{B}$ systém b podmnožin množiny V, kterým říkáme *bloky*. Mějme přirozená čísla k, r a λ. Jestliže pro bloky systému $\mathcal{B}$ platí, že

(i) každý blok obsahuje právě k prvků,
(ii) každý prvek je obsažen v právě r blocích,
(iii) každá dvojice prvků patří současně právě do λ bloků,

tak systém $\mathcal{B}$ se nazývá (b, v, r, k, λ)-*design*, nebo *vyvážený blokový design*.

V anglické literatuře se vyvážený blokový design označuje také jako *vyvážený neúplný blokový design* (anglicky „balanced incomplete block design“, zkráceně *BIBD*). Slovo „vyvážený“ znamená, že každá dvojice prvků se vyskytuje ve stejném počtu bloků. Slovo „neúplný“ někdy zdůrazňuje že pro bloky s k prvky v systému $\mathcal{B}$ platí $k < v$, tedy žádný blok není roven celé množině V.

Protože všechny bloky $B_1, B_2, \ldots, B_b \in \mathcal{B}$ mají stejnou velikost k, tak se říká, že systém $\mathcal{B}$ je k-uniformní. Označení počtu r, v kolika blocích se každý prvek $x \in V$ vyskytuje, pochází z anglického „replication number“. Protože r je stejné pro každý prvek $x \in V$, tak se říká, že design je r-pravidelný (anglicky „r-regular)“.

Dále je dobré si uvědomit, že definice připouští možnost, aby design měl více stejných bloků.

Příklad 3.1. Najděte (triviální) příklad $(3, 3, 2, 2, 1)$-designu.
Podle zadání máme tříprvkovou množinu, například $V = \{1, 2, 3\}$, a hledáme tři 2-prvkové bloky tak, aby každý prvek byl ve dvou blocích a každá dvojice v jediném bloku. Stačí vzít bloky dvojic

$$\mathcal{B} = \{\{1, 2\}, \{2, 3\}, \{3, 1\}\}.$$

Systém $\mathcal{B}$ tvoří hledaný $(3, 3, 2, 2, 1)$-design. ✓

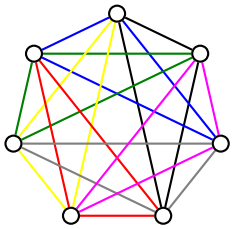
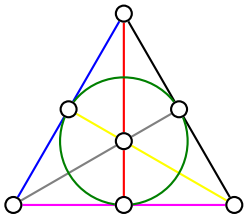
Příklad 3.2. Fanova rovina
Ukažte, že pro nosnou množinu $V = \{1, 2, 3, 4, 5, 6, 7\}$ je systém

$$\mathcal{B} = \{\{1, 2, 4\}, \{2, 3, 5\}, \{3, 4, 6\}, \{4, 5, 7\}, \{5, 6, 1\}, \{6, 7, 2\}, \{7, 1, 3\}\}$$

vyváženým blokovým designem.
Ověříme postupně všechny vlastnosti designu. Evidentně je každý blok $B_i \subseteq V$ pro každé $i = 1, 2, \ldots, 7$. Platí $v = |V| = 7$ a $b = |\mathcal{B}| = 7$.

Design je 3-uniformní, neboť každý blok obsahuje právě tři prvky. Platí $k = 3$. Design je 3-pravidelný, neboť každý prvek je obsažen v právě třech blocích. Platí $r = 3$. A konečně každá dvojice patří do právě jednoho bloku, design je vyvážený s násobností $\lambda = 1$. Jedná se o $(7, 7, 3, 3, 1)$-design. ✓

Designu z Příkladu 3.2. se říká *Fanova rovina*. Jedná se o speciální případ *konečné projektivní roviny* řádu 2. Bloky chápeme jako přímky konečného prostoru (roviny), každé dva body jednoznačně určují nějakou

Obrázek 3.1.: *Dvě různá znázornění Fanovy roviny.*

přímku, každé dvě přímky mají právě jeden bod společný. Existuje pěkné grafické znázornění Fanovy roviny (Obrázek 3.1. vpravo).

Příklad 3.3. Mějme přirozené číslo n a nosnou množinu prvků $V = [1, n]$. Označme $\mathcal{B}$ systém všech dvouprvkových podmnožin množiny V. Systém $\mathcal{B}$ tvoří vyvážený blokový design s parametry $b = n(n-1)/2$, $v = n$, $r = n-1$, $k = 2$, $\lambda = 1$, tedy $(n(n-1)/2, n, n-1, 2, 1)$-design.

Samozřejmě ne pro všechny hodnoty parametrů v, b, r, k a λ musí nějaký design existovat. Například jistě platí $k \leq v$ a dále triviálně musí platit $\lambda = 0$, pokud $k = 1$.

Naproti tomu $b \leq \binom{v}{k}$ platit nemusí. Systém $\mathcal{B} = \{\{1,2\},\{1,2\},\{2,3\},\{2,3\},\{3,1\},\{3,1\}\}$ tvoří $(6,3,4,2,2)$-design, přičemž $b = 6$ a $\binom{v}{k} = \binom{3}{2} = 3$, tj. $b > \binom{v}{k}$.

Konstrukcím designů se věnuje samostatná část diskrétní matematiky. V tomto textu se omezíme jen na některé vybrané výsledky.

Incidenční matice blokového designu

Každému kombinatorickému designu (nejen vyváženému nebo pravidelnému) můžeme jednoznačně přiřadit matici, která popisuje, ve kterých blocích se který prvek nachází.

Definice Mějme blokový (b, v, r, k, λ)-design $\mathcal{B}$ s nosnou množinou $V = \{x_1, x_2, \ldots, x_v\}$ a s bloky $\mathcal{B} = \{B_1, B_2, \ldots, B_b\}$. Obdélníková matice $M = (m_{ij})$ o rozměru $v \times b$, kde každý prvek je definován

$$a_{ij} = \begin{cases} 1 & \text{pokud } x_i \in B_j, \\ 0 & \text{pokud } x_i \notin B_j. \end{cases}$$

se nazývá *incidenční matice* blokového designu $\mathcal{B}$.

Incidenční matice blokového designu je určena jednoznačně až na pořadí sloupců a pořadí řádků. V dalším textu budeme z incidenční matice designu sestavovat kód. Jak ukážeme, pořadí řádků nehraje při sestavení kódu žádnou roli vůbec a různé pořadí sloupců sice povede ke konstrukci různých kódů, které však budou ekvivalentní.

Příklad 3.4. Ukážeme několik jednoduchých příkladů incidenčních matic.

1) Jednoduchý design z Příkladu 3.1. má incidenční matici

	B_1	B_2	B_3
1	1	0	1
2	1	1	0
3	0	1	1

2) Design Fanovy roviny z Příkladu 3.2. má incidenční matici

	B_1	B_2	B_3	B_4	B_5	B_6	B_7
1	1	0	0	0	1	0	1
2	1	1	0	0	0	1	0
3	0	1	1	0	0	0	1
4	1	0	1	1	0	0	0
5	0	1	0	1	1	0	0
6	0	0	1	0	1	1	0
7	0	0	0	1	0	1	1

3) Mějme množinu $V = [1,5]$ a sestavme blokový design $\mathcal{B}$, který obsahuje všechny dvouprvkové množiny V. Design $\mathcal{B} = \binom{V}{2}$ je $(10,5,4,2,1)$-design s incidenční matici (například)

	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	B_9	B_{10}
1	1	1	1	1	0	0	0	0	0	0
2	1	0	0	0	1	1	1	0	0	0
3	0	1	0	0	1	0	0	1	1	0
4	0	0	1	0	0	1	0	1	0	1
5	0	0	0	1	0	0	1	0	1	1

Otázky:

- Kolik jedniček najdeme v každém řádku incidenční matice blokového (b,v,r,k,λ)-designu?
- Kolik jedniček najdeme v každém sloupci incidenční matice blokového (b,v,r,k,λ)-designu?

Součin incidenční matice a transponované matice

Ještě doplníme informaci, že součin incidenční matice M blokového (b,v,r,k,λ)-designu s transponovanou maticí M^T dá čtvercovou matici řádu v. Prvky součinu $(n_{ij}) = MM^T$ jsou podle důkazu Věty 3.2.

$$n_{ij} = \begin{cases} r & \text{pro } i = j, \\ \lambda & \text{pro } i \neq j. \end{cases}$$

Sestavení kódu z incidenční matice

Nyní ukážeme, jak z incidenční matice sestavit netriviální kód. Každý řádek incidenční matice bude jedno kódové slovo. Obecně získáme tak (b,v,d)-kód, kde b je počet bloků designu a v počet prvků nosné množiny. Hodnota d závisí na struktuře designu.

V následujícím příkladu sestavíme dalšími úpravami z incidenční matice designu Fanovy roviny z Příkladu 3.2. kód, který bude dokonce perfektní.

Příklad 3.5. Mějme incidenční sestavenou z designu z Příkladu 3.2. Incidenční matici M jsme sestavili v Příkladu 3.4. Položíme $A = M$ a každý řádek matice A označme jako kódové slovo $\vec{a}_1, \vec{a}_2, \ldots, \vec{a}_7$. Dále sestavíme matici $B = \overline{M}$, ve které uděláme permutaci $(0\,1)$ symbolů 0 a 1 (prohodíme nuly a jedničky). Každý řádek matice B označme jako kódové slovo $\vec{b}_1, \vec{b}_2, \ldots, \vec{b}_7$. Dále označme $\vec{0} = 0000000$ a $\vec{1} = 1111111$. Dostaneme tak celkem 16 kódových slov kódu C_F. Určíme minimální vzdálenost kódu C_F.

Rozlišíme čtyři typy kódových slov: $\vec{0}$, $\vec{1}$, a_i, b_i pro $i = 1,2,\ldots,7$. Nejprve určíme vzdálenosti mezi dvěma kódovými slovy $\vec{a}_i$, $\vec{a}_j$ pro $i \neq j$. Podle vlastností designu $\mathcal{B}$ z Příkladu 3.2. víme, že každé dva bloky mají společný právě jeden prvek. Proto s Využitím Lemmatu 2.4. dostáváme $\operatorname{dist}(\vec{a}_i, \vec{a}_j) = \operatorname{w}(\vec{w}_i) + \operatorname{w}(\vec{w}_j) - 2\operatorname{w}(\vec{a}_i \cap \vec{a}_j) = 3 + 3 - 2 \cdot 1 = 4$ pro $i \neq j$.

Analogicky dostaneme pro kódová slova $\vec{b}_i$, pro $i \neq j$, vzdálenost $\vec{b}_j$ $\operatorname{dist}(\vec{b}_i, \vec{b}_j) = 4$, neboť slova vznikla permutací symbolů 0 a 1. To současně znamená, že $\operatorname{dist}(\vec{a}_i, \vec{b}_j) = 7$ pro $i = j$.

Dále je snadné si rozmyslet, že pro $i = 1,2,\ldots,7$ platí $\operatorname{dist}(\vec{0}, \vec{a}_i) = 3$, $\operatorname{dist}(\vec{0}, \vec{b}_i) = 4$ a $\operatorname{dist}(\vec{0}, \vec{1}) = 7$. Ze symetrie pak vyplývá, že pro $i = 1,2,\ldots,7$ platí $\operatorname{dist}(\vec{1}, \vec{a}_i) = 4$, $\operatorname{dist}(\vec{1}, \vec{b}_i) = 3$ a opět $\operatorname{dist}(\vec{1}, \vec{0}) = 7$.

Zbývá určit vzdálenost $\operatorname{dist}(\vec{a}_i, \vec{b}_j)$ pro $i \neq j$. Zde je šikovné si všimnout, že slova $\vec{a}_i$ a $\vec{b}_j$ se *liší* právě v těch souřadnicích, kde se slova $\vec{a}_i$ a $\vec{a}_j$ *shodují*. Můžeme proto psát $\operatorname{dist}(\vec{a}_i, \vec{b}_j) = 7 - \operatorname{dist}(\vec{a}_i, \vec{a}_j) = 7 - 4 = 3$.

Tím jsme ukázali, že sestavený kód C_F s 16 kódovými slovy

$$\begin{aligned} \text{kód } C_F = \{&0000000, 1000101, 1100010, 0110001, 1011000, 0101100, 0010110, 0001011, \\ &1111111, 0111010, 0011101, 1001110, 0100111, 1010011, 1101001, 1110100.\} \end{aligned}$$

má minimální vzdálenost $\operatorname{dist} C_F = 3$ a jedná se o binární $(7,16,3)$-kód. ✓

Sestavili jsme kód délky 7 se 16 kódovými slovy a s minimální vzdáleností 3. Navážeme dalším příkladem.

Příklad 3.6. Ukážeme, že množina C_F 16 kódových slov z Příkladu 3.5. tvoří perfektní $(7,16,3)$-kód.

Je zřejmé, že kód C_F má délku 7 a že obsahuje 16 kódových slov. Podle Příkladu 3.5. už víme, že minimální vzdálenost kódu je $\operatorname{dist} C_F = 3$. Spočítáme počet kódových slov ve sféře o poloměru každého slova

$$16\left(\binom{7}{0} + \binom{7}{1}(2-1)^1\right) = 16(1+7) = 128 = 2^7.$$

Protože v nerovnosti Hammingovy hranice nastává rovnost, tak kód C_F je perfektní. ✓

Z dvojice Příkladů 3.5. a 3.6. vyplývá, že $A_2(7,3) = 16$. Všimněte si, že jsme tak ověřili další hodnotu v Tabulce 2.1.

Designy vhodné pro sestavení kódu

Jaké designy můžeme použít pro sestavení (binárního) kódu podobně jako v Příkladu 3.5.? Použít můžeme libovolný design, ale ne vždy dostaneme binární kód s vhodnými parametry. V následujícím příkladu sestavíme kód délky 5, s pěti kódovými slovy a s minimální vzdáleností 2.

Příklad 3.7. Mějme $(5,5,4,4,3)$-design $\mathcal{B}$ sestavený ze všech čtyprvkových podmnožin pětiprvkové množiny $[1,5]$. Sestavíme odpovídající incidenční matici a kód.

Bloky designu $\mathcal{B}$ jsou

$$\mathcal{B} = \{\{1,2,3,4\},\{1,2,3,5\},\{1,2,4,5\},\{1,3,4,5\},\{2,3,4,5\}\}$$

Incidenční matice designu $\mathcal{B}$ je

	B_1	B_2	B_3	B_4	B_5
1	1	1	1	1	0
2	1	1	1	0	1
3	1	1	0	1	1
4	1	0	1	1	1
5	0	1	1	1	1

Z incidenční matice sestavíme kód, dostaneme $(5,5,2)$-kód C.

$$\text{kód } C_5 = \begin{cases} 11110 \\ 11101 \\ 11011 \\ 10111 \\ 01111 \end{cases}$$

Víme však, že kód C_5 z Příkladu 1.5. je $(5,10,2)$-kód, který má dvojnásobek kódových slov. ✓

Další příklad kódu: stačí vzít jen kódová slova $a_1, a_2, \ldots, a_7$. Množina vybraných kódových slov tvoří kód $(7,7,3)$-kód C_7, který pochopitelně není perfektní.

Tady chybí část textu, kterou je třeba doplnit.

Lineární kombinace kódových slov

Všimněte si, že sečtením dvou kódových slov $(7,16,3)$-kódu C_F dostaneme opět kódové slovo kódu C_F. Tato vlastnost bude velmi důležitá v Kapitole 4.

Dále si všimněte, že stejná vlastnost platí i pro kód C_3 z Příkladu 1.11., o kterém podle Příkladu 2.14. víme že není perfektní.

Další vlastnosti designů

Všech pět parametrů b, v, r, k a λ vyváženého kombinatorického designu není nezávislých. Následující lemma ukazuje, že platí dvě základní jednoduché rovnosti.

Lemma 3.1. *Pro každý vyvážený (b,v,r,k,λ)-design platí*

(i) $bk = vr$,
(ii) $r(k-1) = \lambda(v-1)$.

Důkaz. Postupně ukážeme obě rovnosti metodou dvojího počítání.

(i) První rovnost vyjadřuje, kolik prvků má multimnožina, kterou získáme sjednocením všech bloků. Tento počet označme x. Máme b boků a každý blok obsahuje k prvků, proto $x = bk$. Současně víme, že nosná množina má v prvků a každý prvek se v designu vyskytuje v r kopiích. Proto $x = vr$.

Porovnáním obou počtů dostáváme první rovnost $bk = vr$.

(i) Každý prvek nosné množiny designu se vyskytuje s každým jiným prvkem v právě λ blocích. Pro libovolný pevně zvolený prvek proto můžeme spočítat, v kolika dvojicích se vyskytuje se všemi $v-1$ zbývajícími prvky. Počet dvojic obsahující pevně zvolený prvek označíme y a ihned vidíme, že $y = \lambda(v-1)$. Na tyto dvojice se můžeme podívat i z pohledu všech bloků. Pevně zvolený prvek se nachází v právě r blocích a v každém takovém bloku tvoří dvojici s každým z $k-1$ zbývajících prvků bloku. Celkem máme $y = r(k-1)$ dvojic.

Porovnáním obou počtů dostáváme druhou rovnost $r(k-1) = \lambda(v-1)$. □

V roce 1940 ukázal britský matematik a statistik R.A. Fisher, že v každém designu platí jednoduchá nerovnost.

Lemma 3.2. Fisherova nerovnost
V každém (b, v, r, k, λ)-designu, ve kterém $k < v$, platí $v \leq b$.

Důkaz. Mějme incidenční matici M (b, v, r, k, λ)-designu. Matice M má rozměr $v \times b$. Označme N matici $N = MM^T$, kde M T je transponovaná matice. Matice N je čtvercová matice řádu v, pro kterou platí

$$n_{ij} = \begin{cases} r & \text{pro } i = j, \\ \lambda & \text{pro } i \neq j. \end{cases}$$

První rovnost $n_{ii} = r$ nastává, protože každý prvek se nachází v r blocích a v součinu s maticí transponovanou bude r-krát sečtena hodnota 1, ostatní součiny budou nulové. Druhá rovnost $n_{ij} = \lambda$ nastává, protože nenulový součin dostáváme pouze v případě, že konkrétní dvojice prvků x_i, x_j se nachází ve stejném bloku, což podle definice kombinatorického designu nastane právě λ-krát.

Pokud je $r \neq \lambda$, tak matice N má jistě lineárně nezávislé řádky a její hodnost je v. Dále z vlastností maticového součinu plyne, že hodnost matice N nemůže být vyšší než hodnost matice A. Dostáváme následující řetěz rovností a nerovností (s využitím vlastnosti, že hodnost součinu matice je rovna minimu z hodností jednotlivých matic)

$$v = \operatorname{rank}(M) = \operatorname{rank}(AA^T) \leq \min\{\operatorname{rank}(A), \operatorname{rank}(A^T\} \leq \operatorname{rank}(A) \leq b.$$

Poslední nerovnost je pak splněna triviálně. □

Zpravidla požadujeme splnění nerovnosti $k < v$, jinak triviálně $b = 1$, zatímco v může být libovolně veliké.

Ještě doplňme, že jistě platí $\lambda \leq r$, neboť počet výskytů konkrétní dvojice prvků nemůže být vyšší, než kolikrát se prvek v designu opakuje. Rovnost nastává pouze v triviálním případě, kdy každý blok designu $B_i = V$.

Symetrický design
Kombinatorický (b, v, r, k, λ)-design je *symetrický*, jestliže nastává rovnost $v = b$ (a proto současně z rovnosti (i) Lemmatu 3.1. plyne také $r = k$). V takovém případě stačí mluvit o symetrickém (v, k, λ)-designu.

Například oba designy z Příkladů i a 3.2. jsou symetrické, zatímco design z Příkladu 3.3. obecně symetrický není, neboť pro $v > k > 2$ je $b = \binom{v}{k} > v$.

Cvičení

3.1.1. Mějme přirozené číslo n. Sestavme systém všech k-prvkových podmnožin množiny V, kde $V = [1, n]$. Jedná se o kombinatorický design? Pokud ano, jaké jsou jeho parametry?

3.1.2.♡ Mějme blokový design ze Cvičení 3.1.1. Jaký rozměr má jeho incidenční matice?

3.1.3.♡ Vezměte incidenční matici A designu Fanovy roviny ($(7, 7, 3, 3, 1)$-designu z Příkladu 3.2.) a vypočítejte AA^T. Ověřte, že diagonální prvky mají hodnotu r a mimodiagonální hodnotu λ.

3.1.4. Sestavte systém všech 3-prvkových podmnožin pětiprvkové množiny. a) Jaké parametry má příslušný design? b) Sestavte bloky designu. c) Jak vypadá kód sestavený dle postupu v Příkladu 3.7. a jaké jsou parametry tohoto kódu?

3.1.5. Podle designu ze Cvičení 3.1.1. sestavíme (n, M, d)-kód. Jaké jsou parametry tohoto kódu?

3.1.6. Podobně jako v Příkladu 3.7. sestavme design jako systém všech $(n-1)$ prvkových podmnožin n prvkové množiny. a) Jaké bude mít design parametry? b) jak bude vypadat incidenční matice designu? c) Jak bude vypadat kód sestavený z incidenční matice?

3.1.7. Mějme $(11, 5, 2)$-design $\mathcal{B}$, kde

$$\mathcal{B} = \{\{1,3,4,5,9\},\{2,4,5,6,10\},\{3,5,6,7,11\},\{1,4,6,7,8\},\{2,5,7,8,9\},\{3,6,8,9,10\},\{4,7,9,10,11\},\{1,5,8,10,11\},\{1,2,6,9,11\},\{1,2,3,7,10\},\{2,3,4,8,11\}\}.$$

Použijte design $\mathcal{B}$ ke konstrukci $(11, 24, 5)$-kódu C_H podobně jako v Příkladu 3.5.

3.2. Konečná tělesa

Nejjednodušší algebraickou strukturou je množina, na které máme definovanou operaci. Připomeňme, že operací na (neprázdné) množině A rozumíme takové zobrazení $A \times A \to A$, které každé dvojici prvků $a, b \in A$ přiřadí prvek $c \in A$. Automaticky tak předpokládáme, že operace je *uzavřená* na množině A, tj. výsledek operace nemůže padnout mimo množinu A. V tomto textu budeme navíc téměř výhradně pracovat s konečnými množinami.

Struktury s jednou operací
Nejprve se podíváme na struktury, kde nad danou množinou máme jedinou operaci. Připomeňme následující klasickou definice z algebry.

Definice Grupoid
Mějme neprázdnou množinu A s operací „$\circ$“ na A. Uspořádaná dvojice $(A, \circ)$ se nazývá *grupoid*. Množině A říkáme *nosič* nebo *nosná množina*.

Pokud je navíc operace „$\circ$“ grupoidu $(G, \circ)$ komutativní, tak máme *komutativní* grupoid.

Příklad 3.8. Uveďme několik jednoduchých příkladů grupoidů (množin s operací).

1) Množina celých čísel s operací obvyklého sčítání $(\mathbb{Z}, +)$ je grupoid, protože sčítání je operace na $\mathbb{Z}$.
2) Množina celých čísel s operací obvyklého odčítání $(\mathbb{Z}, -)$ je grupoid, protože odčítání je operace na $\mathbb{Z}$.
3) Množina celých čísel s operací obvyklého násobení $(\mathbb{Z}, \cdot)$ je grupoid, protože násobení je operace na $\mathbb{Z}$.
4) Množina sudých celých čísel s operací obvyklého sčítání čísel $(\mathbb{S}, +)$ je grupoid, protože součet každých dvou sudých čísel je opět sudé číslo.

Příklad 3.9. Uveďme také několik příkladů, které grupoidem nejsou.

1) Množina přirozených čísel s operací běžného odečítání $(\mathbb{N}, -)$ není grupoid, neboť operace „$-$“ není na množině $\mathbb{N}$ uzavřená. Rozdíl dvou přirozených čísel nemusí být kladné celé číslo.
2) Množina lichých celých čísel s operací obvyklého sčítání čísel $(\mathbb{L}, +)$ není grupoid, protože operace „$+$“ není na množině $\mathbb{L}$ uzavřená. Součet žádných dvou lichých čísel není liché číslo.

Důležitým případem grupoidu $(G, \circ)$ jsou takové grupoidy, jejichž operace „$\circ$“ je navíc asociativní.

Definice Pologrupa
Grupoid $(A, \circ)$ se nazývá *pologrupa* právě tehdy, když operace „$\circ$“ je *asociativní*, tj. pro každé $a, b, c \in A$ platí $a \circ (b \circ c) = (a \circ b) \circ c$.

Příklad 3.10. Uvedeme několik příkladů pologrup.

1) $(\mathbb{N}, +)$ je komutativní pologrupa, neboť sčítání přirozených čísel je asociativní.
2) $(\mathbb{Z}, +)$ je komutativní pologrupa, neboť sčítání celých čísel je asociativní.
3) Grupoid s jednoprvkovým nosičem A a s „libovolnou“ operací „$\circ$“ je pologrupa, neboť operace je triviálně asociativní.
4) $(\mathbb{Z}_n, +)$ je komutativní pologrupa pro libovolné přirozené číslo n, neboť sčítání celých čísel modulo n je asociativní.
5) Mějme grupoid $(A, \circ)$, ve kterém pro každé $a, b \in A$ položíme $a \circ b = c$, kde c je libovolný pevně zvolený prvek A. Potom $(A, \circ)$ je pologrupa (Cvičení 3.2.1.).

Příklad 3.11. Uveďme několik příkladů grupoidů, které asociativní nejsou.

1) Množina celých čísel s operací obvyklého odčítání $(\mathbb{Z}, -)$ pologrupou není, protože například $1 - (2 - 3) = 1 - (-1) = 2$, avšak $(1 - 2) - 3 = (-1) - 3 = -4$.
2) Množina celých čísel s operací obvyklého odčítání $(\mathbb{Z}_n, -)$ modulo n obecně pologrupou není. Pro $n = 1$ a $n = 2$ se jedná o triviální pologrupu a binární pologrupu, avšak pro $n \geq 3$ se o pologrupu nejedná, protože například $1 - (2 - 1) = 1 - (1) = 0$, avšak $(1 - 2) - 1 = (-1) - 1 = -2$.

Neutrální prvek
Důležitým prvkem, pokud v grupoidu nebo v pologrupě existuje, je neutrální prvek.

Definice Neutrální prvek
Mějme grupoid $(A, \circ)$ a prvek $e \in A$. Prvek e se nazývá *neutrální prvek* (vzhledem k operaci „$\circ$“) právě tehdy, když pro každé $a \in A$ platí $e \circ a = a \circ e = a$.

Příklad 3.12. U následujících příkladů grupoidů uvedeme, zda jsou komutativní a jaký je jejich neutrální prvek, pokud existuje.

1) V grupoidu $(\mathbb{Z}, +)$ je neutrální prvek číslo 0.
2) Grupoid $(\mathbb{Z}, -)$ *nemá* neutrální prvek.
3) Dvojice $(\mathbb{S}, +)$ je grupoidem, neboť operace je uzavřená (součtem dvou sudých celých čísel je opět sudé číslo). Víme, že při obvyklém sčítání je neutrálním prvkem číslo 0.
4) Grupoid $(\mathbb{N}, +)$ nemá neutrální prvek, protože číslo 0 není přirozené.
5) Grupoid s jednoprvkovým nosičem a „libovolnou“ operací $(\{a\}, \circ)$ je triviálně komutativní. Jediný prvek grupoidu je současně neutrálním prvkem.
6) Množina vektorů v $\mathbb{R}^3$ spolu s vektorovým součinem vektorů „$\times$“ je nekomutativní grupoid $(\mathbb{R}^3, \times)$. Neutrální prvek v tomto grupoidu není, neboť $\vec{0}$ jistě není neutrální prvek a $\forall \vec{v} \in \mathbb{R}^3, \vec{v} \neq \vec{0} : \vec{v} \times \vec{v} = \vec{0} \neq \vec{v}$.

Následující lemma říká, že neutrální prvek, pokud existuje, je určen jednoznačně.

Věta 3.3. O jednoznačnosti neutrálního prvku
V grupoidu existuje nejvýše jeden neutrální prvek.

Důkaz. Postupujeme přímo. Pokud v grupoidu žádný neutrální prvek neexistuje, tak tvrzení platí.

Předpokládejme dále, že máme dva neutrální prvky e_1, e_2. Protože e_1 je neutrální prvek grupoidu, tak platí $e_1 \circ e_2 = e_2$. Podobně $e_1 \circ e_2 = e_1$, protože e_2 je neutrální prvek grupoidu. Odtud porovnáním ihned plyne $e_1 = e_1 \circ e_2 = e_2$ a neutrální prvek je jediný. □

Grupoid, který je současně asociativní a má neutrální prvek, nazýváme *monoid*.

Inverzní prvek
V dalším textu využijeme, pokud pro dané prvky a, b budeme umět řešit rovnice typu $a \circ x = b$. To znamená, že k prvku a budeme hledat inverze.

Definice Inverzní prvek
Mějme grupoid $(A, \circ)$ s neutrálním prvkem e vzhledem k operaci „$\circ$“. *Inverzním prvkem k prvku* $a \in A$ (vzhledem k operaci „$\circ$“) rozumíme takový prvek $b \in A$, pro který platí obě následující rovnosti.

$$a \circ b = e, \quad b \circ a = e$$

Příklad 3.13. Uveďme několik jednoduchých příkladů grupoidů a inverzí jejich prvků.

1) Máme-li množinu celých čísel s operací obvyklého sčítání $(\mathbb{Z}, +)$, tak neutrálním prvkem je číslo 0 a ke každému celému číslu $a \in \mathbb{Z}$ existuje inverzní prvek $-a$. Zejména číslo 0 je inverzí samo k sobě.
2) Množina celých čísel s operací obvyklého násobení $(\mathbb{Z}, \cdot)$ je monoid s neutrálním prvek 1; inverze existují pouze ke dvěma prvkům, k neutrálnímu prvku $1^{-1} = 1$ a k číslu -1 je inverzí také samotné číslo -1.
3) Množina nenulových celých čísel modulo 10 s operací obvyklého násobení $(\mathbb{Z}_{10} \setminus \{\overline{0}\}, \cdot)$ je monoid s neutrálním prvek $\overline{1}$; inverze existuje pouze k některým lichým prvkům: $\overline{1}^{-1} = \overline{1}$, $\overline{3}^{-1} = \overline{7}$, $\overline{7}^{-1} = \overline{3}$, $\overline{9}^{-1} = \overline{9}$. Sudé prvky a prvek $\overline{5}$ inverzi nemají.

Příklad 3.14. Uveďme několik jednoduchých příkladů grupoidů, ve kterých neexistují inverze.

1) V pologrupě $(\mathbb{S}, \cdot)$ neexistuje neutrální prvek, a proto nemá smysl definovat inverzní prvky.
2) Obvyklé odčítání celých čísel tvoří neasociativní grupoid $(\mathbb{Z}, -)$, ve kterém nemá smysl definovat inverzní prvky, neboť nemá neutrální prvek. Číslo 0 není neutrálním prvkem, neboť pro nenulový prvek a je $0 - a = -a \neq a$.

Inverzní prvek

Grupoid, který je současně asociativní, má neutrální prvek a ke každému prvku existuje inverze, nazýváme grupa.

Definice Grupa
Grupoid $(G, \circ)$ se nazývá *grupa* právě tehdy, když

(i) $\forall a, b, c \in G : a \circ (b \circ c) = (a \circ b) \circ c$, (asociativita)
(ii) $\exists e \in G \, \forall a \in G : e \circ a = a \circ e = a$, (existence neutrálního prvku)
(iii) $\forall a \in G \, \exists b \in G : a \circ b = b \circ a = e$, kde e je neutrální prvek vzhledem k operaci „$\circ$“. (existence inverze)

Prvek b se nazývá *inverzní k prvku* a (vzhledem k operaci „$\circ$“) a značíme jej a^{-1}, případně $-a$.

Příklad 3.15. Uvedeme několik jednoduchých příkladů grup včetně určení neutrálního prvku.

1) $(\mathbb{Z}, +)$ je komutativní grupa. Už víme, že se jedná o monoid s neutrálním prvkem 0, inverzní k prvku $a \in \mathbb{Z}$ je opačný prvek $-a$.
2) $(\mathbb{Q}, +)$ a $(\mathbb{R}, +)$ jsou komutativní grupy. Opačným prvkem k prvku x je vždy prvek $-x$.
3) $(\mathbb{R} \setminus \{0\}, \cdot)$ je komutativní grupa. Násobení nenulových čísel je grupoid, operace je komutativní i asociativní, neutrálním prvkem je číslo 1 a inverzí k (nenulovému) prvku a je prvek $\frac{1}{a}$.
4) $((0, \infty), \cdot)$ je komutativní grupa. Násobení kladných čísel je jistě grupoid, operace je (stejně jako u předchozího příkladu) komutativní i asociativní, neutrálním prvkem je číslo 1 a inverzí ke kladnému prvku a je (kladný) prvek $\frac{1}{a}$.
5) Grupoid $(\mathbb{Z}_n, +)$ sčítání modulo n je grupa s neutrálním prvkem $\overline{0}$.
6) Grupoid $(\mathbb{Z}_p \setminus \{\overline{0}\}, \cdot)$ násobení modulo n je grupa s neutrálním prvkem $\overline{1}$ pouze, pokud p je prvočíslo.

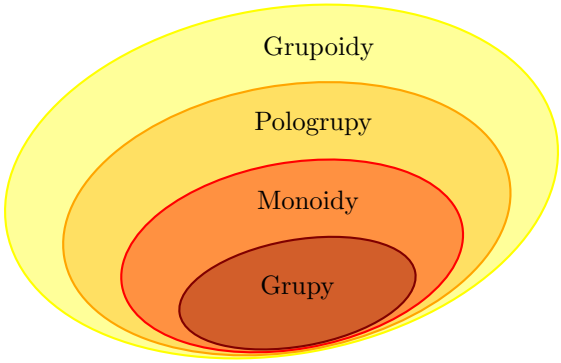
Příklad 3.16. Uvedeme také několik příkladů, které grupou nejsou.

1) $(\mathbb{N}, +)$ není grupou, protože není ani monoidem, neboť v pologrupě $(\mathbb{N}, +)$ není neutrální prvek.
2) $(\mathbb{Z}, -)$ není grupa protože operace „$-$“ není asociativní.
3) $(\mathbb{N}, \cdot)$ není grupa. Už víme, že se jedná o monoid, neutrálním prvek je číslo 1, avšak k číslům větším než 1 neexistují inverze.
4) $(\mathbb{R}, \cdot)$ není grupa, protože k prvku 0 neexistuje inverze.
5) $(\mathbb{R}^*, \cdot)$, kde $\mathbb{R}^*$ je rozšířená množina reálných čísel včetně ∞ a $-\infty$, není grupa. Dokonce, i pokud bychom dodefinovali $0 \cdot \infty = 1$, dostaneme $0^{-1} = \infty$ a $\infty^{-1} = 0$, tak stále není jasná inverze k $-\infty$. Inverzí už nemůže být 0, protože inverze jsou určeny jednoznačně.
6) Množina vektorů v $\mathbb{R}^3$ spolu s vektorovým součinem vektorů je grupoid $(\mathbb{R}^3, \times)$, který není grupou, neboť operace není asociativní (Příklad 3.11.), a navíc ani nemá neutrální prvek (Cvičení 3.2.2.), a proto nemá smysl k vektorům hledat inverzní prvky.
7) $(\mathbb{Z}_4 \setminus \{0\}, \cdot)$ (násobení nenulových čísel modulo 4) netvoří grupu. Prvek $\overline{2}$ nemá inverzi.
8) $(M_{n,n}, \cdot)$ (čtvercové matice řádu n s operací násobení matic) netvoří grupu, protože k singulárním maticím neexistují inverzní matice.

Příklad 3.17. Množina nenulových celých čísel modulo 11 s operací obvyklého násobení $(\mathbb{Z}_{11} \setminus \{\overline{0}\}, \cdot)$ je grupa s neutrálním prvek $\overline{1}$; ke každému prvku existuje inverze: $\overline{1}^{-1} = \overline{1}$, $\overline{2}^{-1} = \overline{6}$, $\overline{3}^{-1} = \overline{4}$, $\overline{4}^{-1} = \overline{3}$, $\overline{5}^{-1} = \overline{9}$, $\overline{6}^{-1} = \overline{2}$, $\overline{7}^{-1} = \overline{8}$, $\overline{8}^{-1} = \overline{7}$, $\overline{9}^{-1} = \overline{5}$ a $\overline{10}^{-1} = \overline{10}$. Inverze násobení modulo 11 přehledně shrnuje Tabulka 3.1.

a	0	1	2	3	4	5	6	7	8	9	10
a^{-1}	–	1	6	4	3	9	2	8	7	5	10

Tabulka 3.1.: *Tabulka inverzí násobení modulo 11.*



Obrázek 3.2.: *Hierarchie grupoidů, pologrup, monoidů a grup.*

V každé grupě je inverzní prvek ke každému prvku grupy určen jednoznačně. Následující věta tak ukazuje, že formulace v definici grupy je korektní: inverzní prvek k prvku a je v grupě jediný, a proto jej můžeme označit a^{-1}.

Věta 3.4. O existenci a jednoznačnosti inverzního prvku v grupě
Mějme grupu $(G, \circ)$ s neutrálním prvkem e. Pro každé $a \in G$ existuje jediný inverzní prvek $b \in G$ tak, že $a \circ b = b \circ a = e$.

Důkaz. Postupujeme přímo. Podle definice grupy však víme, že $\forall a \in G$ existuje alespoň jedna inverze. Označme b_1 a b_2 inverzní prvky k prvku a v grupě $(G, \circ)$.

$$b_1 = b_1 \circ e = b_1 \circ (a \circ b_2) = (b_1 \circ a) \circ b_2 = e \circ b_2 = b_2.$$

Při úpravě jsme postupně využili: existenci neutrálního prvku e, využili jsme, že b_2 je inverzí k a, asociativitu operace, vlastnosti inverze b_1 k prvku a a vlastnosti neutrálního prvku. Proto neexistuje více různých neutrálních prvků a existuje alespoň jeden, a tak ke každému prvku grupy existuje právě jeden inverzní prvek vzhledem k operaci „$\circ$“. □

Používáme-li pro operaci „$\circ$“ multiplikativní notaci „$\cdot$“, mluvíme zpravidla o inverzních prvcích a^{-1}. Používáme-li aditivní notaci „$+$“, tak zpravidla mluvíme o opačných prvcích $-a$.

Struktury se dvěma operacemi
V předmětu Algebra byla zavedena struktura se dvěma operacemi, přičemž operace jsou navíc provázané požadavkem na splnění distributivních zákonů (levého a pravého distributivního zákona).

Definice Mějme neprázdnou množinu R a mějme na R dvě binární operace „$+$“ a „$\cdot$“. Uspořádaná trojice $(R, +, \cdot)$ je *okruh* právě tehdy, když platí

(i) $\forall a, b \in R : a + b = b + a$ (komutativita „$+$“)
(ii) $\forall a, b, c \in R : (a + b) + c = a + (b + c)$ (asociativita „$+$“)
(iii) $\exists 0 \in R \; \forall a \in R : a + 0 = a$ (existence neutrálního prvku vzhledem k „$+$“)
(iv) $\forall a \in R \; \exists - a \in R : a + (-a) = 0$ (existence opačného prvku vzhledem k „$+$“)
(v) $\forall a, b, c \in R : (a \cdot b) \cdot c = a \cdot (b \cdot c)$ (asociativita „$\cdot$“)
(vi) $\forall a, b, c \in R : a \cdot (b + c) = a \cdot b + a \cdot c$, $(b + c) \cdot a = b \cdot a + c \cdot a$ (distributivita zleva, zprava vhledem k „$+$“)

Definici okruhu je možno vyslovit stručněji. Okruhem nazveme takovou trojici $(R, +, \cdot)$, kdy dvojice $(R, +)$ tvoří komutativní grupu, dvojice $(R, \cdot)$ tvoří pologrupu a navíc jsou splněny oba distributivní zákony násobení vzhledem ke sčítání.

Příklad 3.18. Uveďme několik klasických příkladů okruhů.

1) Množina celých čísel s operacemi obvyklého sčítání a násobní $(\mathbb{Z}, +, \cdot)$ je nejjednodušším klasickým příklad okruhu. Pojem okruhu byl zaveden, aby zobecnil právě počítání s celými čísly.
2) Množina reálných čísel s operacemi obvyklého sčítání a násobní $(\mathbb{R}, +, \cdot)$ je okruh.
3) Množina zbytkových tříd s operacemi sčítání a násobní $(\mathbb{Z}_n, +, \cdot)$ je okruh.

$+$	0	1
0	0	1
1	1	0

$\cdot$	0	1
0	0	0
1	0	1

Tabulka 3.2.: *Operace v binárním okruhu* $(\{0,1\},+,\cdot)$.

4) Označme množinu $R=\{0,1\}$. Trojice $(R,+,\cdot)$, kde operace jsou určeny Tabulkami 3.2., je okruh (Cvičení 3.2.5.).
5) *Triviální okruh* je $(R,+,\cdot)$, kde $R=\{0\}$.

Příklad 3.19. Množina přirozených čísel s operacemi obvyklého sčítání a násobní $(\mathbb{N},+,\cdot)$ okruhem není. Vzhledem k operaci „+“ neexistuje v $\mathbb{N}$ neutrální prvek, navíc k žádnému přirozenému číslu neexistuje v $\mathbb{N}$ opačné číslo. ✓

Nula a jednička kruhu

Obvykle požadujeme, aby operace okruhu splňovaly další vlastnosti. Nejprve zavedeme pojem nuly a jedničky okruhu.

Definice Řekneme, že okruh $(R,+,\cdot)$ je *komutativní* právě tehdy, když je komutativní operace „$\cdot$“, tj. pokud pro každé $a,b\in R$ platí $a\cdot b=b\cdot a$.

Neutrální prvek grupy $(R,+)$ se nazývá *nula okruhu* a značíme jej 0. Pokud existuje neutrální prvek pologrupy $(R,\cdot)$, nazývá se *jednička okruhu* a značí se 1. Dále řekneme, že nenulový prvek $a\in R$ je *jednotkou okruhu* právě tehdy, když k němu existuje inverzní prvek a^{-1} v monoidu $(R\setminus\{0\},\cdot)$.

Pro prvky okruhu umíme z definice ukázat následující vlastnosti.

Věta 3.5. Vlastnosti okruhu

Mějme okruh $(R,+,\cdot)$. Pro každé $a,b,c\in R$ platí následující rovnosti.

(i) $a0=0a=0$
(ii) $a(-b)=(-a)b=-(ab)$
(iii) $(-a)(-b)=ab$
(iv) $a(b-c)=ab-ac$ *a* $(b-c)a=ba-ca$

Jestliže navíc $(R,+,\cdot)$ má jedničku 1, tak platí následující rovnosti.

(v) $(-1)a=-a$
(vi) $(-1)(-1)=1$

Důkaz.
(i) S využitím levé distributivity operace „$\cdot$“ vzhledem k operaci „+“ dostaneme

$$a0=a(0+0)=a0+a0,$$

odkud krácením (nebo odečtením opačného prvku k $a0$) v grupě $(R,+)$ dostáváme $0=a0$. Podobně $0a=$ $=(0+0)a=0a+0a$, odkud krácením (odečtením opačného prvku) dostáváme $0=0a$.
(ii) Pro důkaz další rovnosti využijeme neutrální prvek 0, distributivitu operace „+“ a předchozí bod (i).

$$a(-b)+ab=a(-b+b)=a0=0$$

Odečtením ab od obou stran rovnosti dostaneme $a(-b)=0-(ab)=-(ab)$. Analogicky $(-a)b+ab=(-a+$ $+a)b=0b=0$ a opět odečtením ab dostaneme $(-a)b=0-(ab)=-(ab)$.
(iii) Použijeme-li dvakrát předchozí bod (ii), tak dostaneme

$$(-a)(-b)=-(a(-b))=-(-(ab))=ab,$$

neboť opačným prvek k opačnému prvku $-(ab)$ je podle prvek ab.
(iv) Opět s využitím bodu (ii) dostaneme

$$a(b-c)=ab+a(-c)=ab+-(ac)=ab-ac.$$

Analogicky $(b-c)a=ba+(-c)a=ba+-(ca)=ba-ca$.
(v) Jestliže zvolíme $a=-1$, tak dle bodu (ii) dostaneme $(-1)b=-(1b)=-b$.

(vi) Jestliže navíc zvolíme $b = -1$, tak dle bodu (ii) dostaneme $(-1)(-1) = -(1(-1)) = -(-1) = 1$. □

Pokud navíc požadujeme, aby nosná množina R okruhu $(R, +, \cdot)$ byla netriviální, aby operace násobení byla komutativní a aby v okruhu ke každému nenulovému prvku existovala inverze, tak dostaneme těleso.

Definice Těleso
Netriviální komutativní okruh $(R, +, \cdot)$ s jedničkou se nazývá *těleso* právě tehdy, když ke každému nenulovému prvku existuje inverze. Konečné těleso s n prvky budeme značit $GF(n)$.

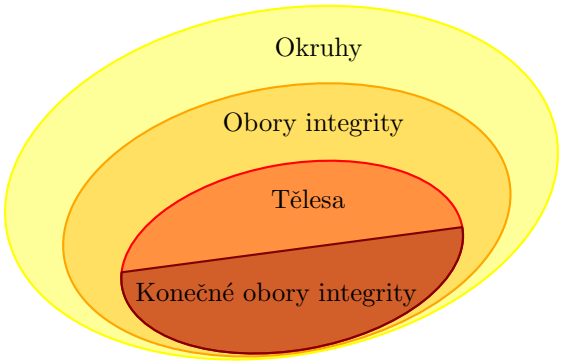
Pochopitelně existuje mnoho příkladů těles, z Matematické analýzy známe $(\mathbb{R}, +, \cdot)$ i $(\mathbb{C}, +, \cdot)$ avšak v tomto kurzu nás budou zajímat výhradně taková tělesa, která mají *konečně mnoho* prvků, tj. konečná tělesa.

Příklad 3.20. Uveďme několik příkladů těles.

1) Klasické číselné obory $(\mathbb{Q}, +, \cdot)$ a $(\mathbb{R}, +, \cdot)$ jsou tělesa. Jedničkou je číslo 1.
2) $(\mathbb{Z}_p, +, \cdot)$, kde p je prvočíslo, je tělesem. Jedničkou je třída $\overline{1}$. Důkaz tohoto tvrzení je ponechán jako Cvičení 3.2.3.

Příklad 3.21. Uveďme několik okruhů, které tělesem nejsou.

1) Obor integrity $(\mathbb{Z}, +, \cdot)$ není tělesem, například k číslu 2 neexistuje prvek inverzní.
2) Okruh $(\mathbb{S}, +, \cdot)$ není tělesem, neboť nemá jedničku.
3) Okruh $(\mathbb{Z}_6, +, \cdot)$ není tělesem, protože například k prvku 2 neexistuje inverzní prvek vzhledem k násobení.
4) Okruh $(\mathbb{Z}_n, +, \cdot)$, kde n je složené číslo, není tělesem, protože obsahuje dělitele nuly.
5) Triviální okruh není podle definice tělesem.



Obrázek 3.3.: *Hierarchie okruhů, oborů integrity a těles.*

Konečná tělesa $GF(n)$ řádu n existují právě tehdy, když n je mocnina prvočísla. Je možno ukázat, že okruh zbytkových tříd $(Z_p, +, \cdot)$ je těleso právě tehdy, když počet prvků p je prvočíslo. Pro řády rovny vyšší mocniny prvočísel p^k, kde $k > 1$ se konečná tělesa konstruují například jako vhodné faktorové okruhy okruhu polynomů.

Pro řády n, kde n není mocnina prvočísla, konečná tělesa $GF(n)$ neexistují. Zejména například neexistuje konečné těleso řádu 10, ale ani řádu 6 nebo 12. Důkaz tohoto tvrzení je nad rámec tohoto kurzu.

Věta 3.6. *Je-li p prvočíslo, tak okruh zbytkových tříd $(\mathbb{Z}_p, +, \cdot)$ je tělesem.*

Důkaz v tomto předmětu vynecháme.

Příklad 3.22. Ukážeme příklady malých konečných těles.

1) Těleso $(\mathbb{Z}_2, +, \cdot)$ má tabulky operací uvedeny v Tabulce 3.3.
2) Těleso $(\mathbb{Z}_3, +, \cdot)$ má tabulky operací uvedeny v Tabulce 3.4.
3) Těleso se čtyřmi prvky $\{0, 1, 2, 3\}$ má tabulky operací uvedeny v Tabulce 3.5.
4) Těleso $(\mathbb{Z}_5, +, \cdot)$ má tabulky operací uvedeny v Tabulce 3.6.

$+$	0	1
0	0	1
1	1	0

$\cdot$	0	1
0	0	0
1	0	1

Tabulka 3.3.: *Cayleyho tabulky operací „$+$“ a „$\cdot$“ v tělese $(\mathbb{Z}_2, +, \cdot)$.*

$+$	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

$\cdot$	0	1	2
1	0	0	0
1	0	1	2
2	0	2	1

Tabulka 3.4.: *Cayleyho tabulky operací „$+$“ a „$\cdot$“ v tělese $(\mathbb{Z}_3, +, \cdot)$.*

$+$	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

$\cdot$	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

Tabulka 3.5.: *Cayleyho tabulky operací „$+$“ a „$\cdot$“ v tělese se čtyřmi prvky.*

$+$	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

$\cdot$	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

Tabulka 3.6.: *Cayleyho tabulky operací „$+$“ a „$\cdot$“ v tělese $(\mathbb{Z}_3, +, \cdot)$.*

$+$	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	0
2	2	3	4	5	6	7	0	1
3	3	4	5	6	7	0	1	2
4	4	5	6	7	0	1	2	3
5	5	6	7	0	1	2	3	4
6	6	7	0	1	2	3	4	5
7	7	0	1	2	3	4	5	6

$\cdot$	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	3	1	7	5
3	0	3	6	5	7	4	1	2
4	0	4	3	7	6	2	5	1
5	0	5	1	4	2	7	4	6
6	0	6	7	1	5	4	2	4
7	0	7	5	2	1	6	4	3

Tabulka 3.7.: *Cayleyho tabulky operací „$+$“ a „$\cdot$“ v tělese $(\{0,1,2,3,4,5,6,7\}, +, \cdot)$ řádu 4.*

Pro malé řády existují tělesa řádu 2, 3, 4, 5, 7, 8, 9 a 11. Žádná tělesa řádu 6, 10 ani 12 nemohou existovat.

Příklad 3.23. Tabulky 3.7. popisují těleso řádu 8.

Příklady těles prvočíselných řádů jsou $(\mathbb{Z}_p, +, \cdot)$. Příklady těles řádů mocniny prvočísel konstruujeme jako faktorové okruhy okruhů polynomů $(\mathbb{Z}_2[x]/\langle p(x)\rangle, +, \cdots)$. Podrobný popis konstrukce spadá do jiného předmětu, do teoretické algebry.

Cvičení

3.2.1.$^{\heartsuit}$ *Mějme grupoid $(A, \circ)$, ve kterém pro každé $a, b \in A$ položíme $a \circ b = c$, kde c je libovolný pevně zvolený prvek A. Ukažte, že $(A, \circ)$ je pologrupa.*

3.2.2. Množina vektorů v $\mathbb{R}^3$ spolu s vektorovým součinem vektorů je grupoid $(\mathbb{R}^3, \times)$. Má tento grupoid neutrální prvek?

3.2.3. Ukažte, že pro každé prvočíslo p je okruh zbytkových tříd $(\mathbb{Z}_p, +, \cdot)$ tělesem.

3.2.4. Ukažte, že pro každé složené číslo p okruh zbytkových tříd $(\mathbb{Z}_n, +, \cdot)$ není tělesem.

3.2.5.$^{\heartsuit}$ *Ukažte, že trojice $(R, +, \cdot)$, kde operace jsou určeny Tabulkou 3.2., je okruh.*

3.3. Kongruence

Na každé množině můžeme zavést relaci. Relace R na množině A je nějaká pevná podmnožina R kartézského součinu $A \times A$. Speciálním typem relací je kongruence.

Fakt, že celé číslo a dává po dělení přirozeným číslem m zbytek r znamená, že podle Věty o jednoznačnosti podílu a zbytku existují taková celá čísla q, r, pro která platí $a = qm + r$ a r a $0 \leq r < m$. Číslu q říkáme *podíl* a číslu r *zbytek* po dělení čísla a číslem m.

Definice Mějme celá čísla a, b a přirozené číslo m. Řekneme, že čísla a, b jsou *kongruentní modulo* m, jestliže dávají stejný zbytek po dělení číslem m. Zapisujeme $a \equiv b \pmod{m}$.

V opačném případě říkáme, že čísla a, b nejsou kongruentní modulo m a píšeme $a \not\equiv b \pmod{m}$.

Příklad 3.24. Platí

1) $23 \equiv 3 \pmod{5}$
2) $21 \equiv 36 \pmod{5}$
3) $21 \not\equiv 36 \pmod{10}$
4) $31 \equiv -9 \pmod{10}$
5) $34 \equiv 13 \pmod{7}$

Počítání na hodinách

Hodiny „počítáme modulo 24“, protože nerozlišujeme více než 24 hodin během dne. V běžné řeči obvykle rozlišujeme jen 12 hodin. Hodiny uvádíme pouze v intervalu 1 až 12 (případně 0 až 11 nebo 0 až 23). Uvědomte si, že při počítání časových údajů na hodinách existuje pouze konečně mnoho výsledků početních operací! Například řekneme-li v 7 hodin „za čtyři hodiny“, tak mluvíme o čase v 11 hodin. Odkážeme-li se však v 11 hodin na čas „za čtyři hodiny“, tak mluvíme o čase ve 3 hodiny (přičemž z kontextu může, ale nemusí být jasné, zda se jedná o odpoledne nebo o hlubokou noc).

Výsledky počítání můžeme popsat Cayleyho tabulkou.

Příklad 3.25. Sestavte Cayleyho tabulku sčítání „modulo 12“ s čísly 0 až 11.

Počítání „modulo 12“ s čísly 0 až 11, shrnou následující Cayleyho tabulky. Sčítání modulo 12 reprezentuje Tabulka 3.8. a násobení modulo 12 Tabulka 3.9. ✓

+	0	1	2	3	4	5	6	7	8	9	10	11
0	0	1	2	3	4	5	6	7	8	9	10	11
1	1	2	3	4	5	6	7	8	9	10	11	0
2	2	3	4	5	6	7	8	9	10	11	0	1
3	3	4	5	6	7	8	9	10	11	0	1	2
4	4	5	6	7	8	9	10	11	0	1	2	3
5	5	6	7	8	9	10	11	0	1	2	3	4
6	6	7	8	9	10	11	0	1	2	3	4	5
7	7	8	9	10	11	0	1	2	3	4	5	6
8	8	9	10	11	0	1	2	3	4	5	6	7
9	9	10	11	0	1	2	3	4	5	6	7	8
10	10	11	0	1	2	3	4	5	6	7	8	9
11	11	0	1	2	3	4	5	6	7	8	9	10

Tabulka 3.8.: *Tabulka sčítání „modulo 12“ s čísly 0 až 11.*

·	0	1	2	3	4	5	6	7	8	9	10	11
0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	10	11
2	0	2	4	6	8	10	0	2	4	6	8	10
3	0	3	6	9	0	3	6	9	0	3	6	9
4	0	4	8	0	4	8	0	4	8	0	4	8
5	0	5	10	3	8	1	6	11	4	9	2	7
6	0	6	0	6	0	6	0	6	0	6	0	6
7	0	7	2	9	4	11	6	1	8	3	10	5
8	0	8	4	0	8	4	0	8	4	0	8	4
9	0	9	6	3	0	9	6	3	0	9	6	3
10	0	10	8	6	4	2	0	10	8	6	4	2
11	0	11	10	9	8	7	6	5	4	3	2	1

Tabulka 3.9.: *Tabulka násobení „modulo 12“ s čísly 0 až 11.*

Všimněte si, že operace násobení modulo 12 *není* uzavřená na množině nenulových čísel $\{1, 2, \ldots, 11\}$. Například $3 \cdot 8 = 0$. To mj. znamená, že trojice $(\mathbb{Z}_{12}, +, \cdot)$ *není* těleso.

Vlastnosti kongruencí modulo m

Pro dvě čísla kongruentní modulo m platí následující lemmata.

Lemma 3.7. *Mějme celá čísla a, b a přirozené číslo m. Potom $a \equiv b \pmod m$ právě tehdy, když $m \mid (b - a)$.*

Důkaz. Tvrzení má tvar ekvivalence, ukážeme obě implikace.
„⇒“ Jestliže platí $a \equiv b \pmod m$, tak podle definice kongruence existují taková celá čísla q_1, q_2, r, kde $0 \leq r < m$ že platí

$$a = q_1 m + r, \quad b = q_2 m + r.$$

Všimněte si, že zbytek po dělení číslem m je pro obě čísla a, b stejný. Rozdíl čísel $a - b$ pak je roven

$$a - b = (q_1 m + r) - (q_2 m + r) = (q_1 - q_2)m.$$

To ale znamená, že $a - b$ je násobkem modulu m, neboli, že m dělí $a - b$.
„⇐“ Jestliže m dělí $a - b$, tak existuje takové celé číslo k, že $a - b = mk$. To ale znamená $a = b + mk$.

Dále označme zbytek po dělení čísla a číslem m označme r. To znamená, že existuje takové celé číslo q, že

$$a = qm + r,$$

kde $0 \leq r < m$. Dosazením do levé strany rovnosti dostane

$$b + mk = qm + r.$$

Odtud vyjádříme $b = (q - k)m + r$, kde $0 \leq r < m$. Rozdíl celých čísela $q - k$ je jistě celé číslo. To znamená, že b dává po dělení číslem m stejný zbytek r jako číslo a. □

Lemma 3.8. *Mějme celá čísla a, b a přirozené číslo m. Potom $a \equiv b \pmod m$ právě tehdy, když existuje celé číslo k takové, že $b = a + km$.*

Důkaz je ponechán jako Cvičení 3.3.1.

Konečná tělesa a kongruence

Mějme množinu A. Každá relace ekvivalence na množině A jednoznačně určuje nějaký rozklad množiny A a naopak, z každého rozkladu množiny A je možno jednoznačně sestavit relaci ekvivalence na množně A.

Protože relace kongruence je relací ekvivalence na množině všech přirozených čísel, tak tato relace určuje rozklad množiny $\mathbb{Z}$ na *třídy ekvivalence*. Také se jim říká *zbytkové třídy modulo m*. Zbytkové třídy obvykle značíme $\overline{a}$, případně $\overline{a}_m$, kde $a \in \overline{a}$ je *reprezentant* třídy $\overline{a}$ a m je modul relace kongruence. Reprezentantem může být libovolný prvek třídy.

Jak jsme uvedli dříve, nejjednodušším příkladem konečných těles jsou právě tělesa zbytkových tříd modulo p, kde p je prvočíslo.

V algebře bylo ukázáno, že třídy rozkladu $\overline{0}, \overline{1}, \ldots, \overline{m-1}$ při nějakém pevně zvoleném modulu m tvoří faktorovou grupu s operací sčítání komplexů (zbytkových tříd) a třídy s operací násobení komplexů tvoří

grupoid. Pokud je modul m prvočíslem, tak třídy tvoří (konečné) těleso s operacemi sčítání a násobení zbytkových tříd, které značíme $(\mathbb{Z}_p, +, \cdot)$.

Pokud však m není prvočíslo, tak operace sčítání nenulových zbytkových tříd netvoří grupu a takový okruh $(\mathbb{Z}_m, +, \cdot)$ těleso netvoří.

Při počítání se zbytkovými třídami (násobení a sčítání) můžeme pěkně využít reprezentantů a vlastností zbytků při daném modulu. Například v $\mathbb{Z}_{10}$ je $\overline{4}_{10} \cdot \overline{9}_{10} = \overline{4 \cdot 9}_{10} = \overline{36}_{10} = \overline{6}_{10}$. Výpočet může být ještě jednodušší, pokud si uvědomíme, že $\overline{9}_{10} = \overline{-1}_{10}$, potom $\overline{4}_{10} \cdot \overline{9}_{10} = \overline{4}_{10} \cdot \overline{-1}_{10} = \overline{-4}_{10} = \overline{6}_{10}$.

ISBN-10 kód

Většina dnešních knih má přiřazený ISBN kód (International Standard Book Number). Každá kniha má svůj unikátní ISBN kód, přičemž čísla ISBN kódu jsou volena tak, aby umožnila detekovat nejběžnější chyby, které při psaní kódu mohou vzniknout.

Do roku 2007 se používal ISBN-10 kód, jehož kódová slova jsou vybraná slova délky 10 v jedenáctkové soustavě. Prvních devět cifer je vždy voleno mezi ciframi 0 až 10, desátá však může nabývat hodnoty 10, která se zapisuje jako X. Kódová slova $x_1, x_2, \ldots, x_{10}$ splňují kontrolní schéma

$$10x_1 + 9x_2 + \cdots + 2x_9 + 1x_{10} \equiv 0 \pmod{11}. \tag{4}$$

Všimněte si, že kontrolní schéma může být ekvivalentně zapsáno dvěma způsoby $\sum_{i=1}^{10}(11-i)x_i \equiv \sum_{i=1}^{10} ix_i \equiv 0 \pmod{11}$.

Příklad 3.26. Ukážeme, že ISBN-10 kód detekuje jednu chybu.

Předpokládejme, že v zápisu ISBN-10 kódu dojde k jedné chybě. Místo kódového slova $x_1, x_2, \ldots, x_{10}$ bude přijato kódové slovo $y_1, y_2, \ldots, y_{10}$, které se liší v jediné cifře $y_j = x_j + a$, kde $a \neq 0$. Dosazením do kontrolního schematu dostaneme

$$\sum_{i=1}^{10} iy_i = \sum_{i=1}^{10} ix_i + ja \equiv 0 + ja \pmod{11}.$$

Protože $a \neq 0$ a $j \neq 0$, tak součin $aj \not\equiv 0 \pmod{11}$, neboť těleso $(Z_{11}, +, \cdot)$ nemá dělitele nuly. ✓

Příklad 3.27. Ukážeme, že ISBN-10 kód detekuje transpozici libovolných dvou znaků.

Předpokládejme, že v zápisu ISBN-10 kódu dojde k prohození libovolných dvou znaků. Místo kódového slova $x_1, x_2, \ldots, x_{10}$ bude přijato kódové slovo $y_1, y_2, \ldots, y_{10}$, které se liší prohozením dvou cifer $y_j = x_k$ a $y_k = x_j$, kde $1 \leq j, k \leq 10$, $j \neq k$. Dosazením do kontrolního schematu dostaneme

$$\sum_{i=1}^{10} iy_i = \sum_{i=1}^{10} ix_i + jx_k - kx_k - (kx_j - jx_j) = (j-k)(x_k - x_j) \equiv 0 \pmod{11}.$$

Protože $j - k \neq 0$ a $x_k - x_j \neq 0$, tak součin $(j-k)(x_k - x_j) \not\equiv 0 \pmod{11}$, neboť těleso $(Z_{11}, +, \cdot)$ nemá dělitele nuly. To znamená, že ISBN-10 kód detekuje chybu. ✓

Příklad 3.28. Ukážeme, že minimální vzdálenost dvou kódových slov ISBN-10 kódu (neboli dvou platných ISBN-10 kódů) je 2.

Tvrzení ukážeme přímo. Abychom ukázali, že nejmenší vzdálenost není větší než 2, stačí najít dva platné ISBN kódy, které se liší ve dvou cifrách. Například 1500000000 je platný ISBN-10 kód a současně 3400000000 je platný ISBN-10 kód, a přitom Hammingova vzdálenost těchto dvou slov je 2. To znamená, že minimální vzdálenost ISBN-10 kódu je nejvýše 2.

Na druhou stranu podle Cvičení 3.26. víme, že ISBN-10 kód detekuje jednu chybu, proto minimální vzdálenost ISBN-10 kódu je alespoň 2. Celkem dostáváme, že minimální vzdálenost ISBN-10 kódu je 2. ✓

ISBN-13 kód

Od roku 2007 se používá třináctimístný ISBN-13 kód, jehož kódová slova jsou vybraná slova v desítkové (nikoliv jedenáctkové!) soustavě. Prvních dvanáct cifer je opět vybíráno z cifer 0 až 9 a třináctá cifra je doplněna tak, aby kódová slova $x_1, x_2, \ldots, x_{13}$ vyhovovala schématu

$$x_1 + 3x_2 + x_3 + 3x_4 + x_5 + 3x_6 + x_7 + 3x_8 + x_9 + 3x_{10} + x_{11} + 3x_{12} + x_{13} \equiv 0 \pmod{10}.$$

Důkaz, že ISBN-13 kód odhalí jednu chybu i velkou část transpozic sousedních cifer je ponechán jako Cvičení 3.3.6. a 3.3.8.

Cvičení

3.3.1. Dokažte Lemma 3.8.: Pro celá čísla a, b platí $a \equiv b \pmod{m}$ právě tehdy, když existuje celé číslo k takové, že $b = a + km$.

3.3.2. Vypočítejte zbytek po dělení a) $2^{10} \bmod 7$, b) $3^{50} \bmod 5$,

3.3.3. Ukažte, že druhá mocnina libovolného celého čísla je kongruentní s 0 nebo 1 modulo 4.

3.3.4. Najděte tvrzení analogická ke Cvičení 3.3.3. i pro zbytky a) modulo 3, b) modulo 5.

3.3.5. Kolik chyb dokáže opravit ISBN-10 kód?

3.3.6. Kolik chyb dokáže detekovat ISBN-13 kód?

3.3.7. Kolik chyb dokáže opravit ISBN-13 kód? Své tvrzení dokažte.

3.3.8. Ukažte, že ISBN-13 kód dokáže detekovat prohození dvou sousedních cifer, které se neliší o 5. Lze pomocí ISBN-13 detekovat prohození libovolných dvou cifer?

3.3.9. Jaká je minimální vzdálenost ISBN-13 kódu?

3.3.10. Pro jednoduchost předpokládejme, že všechna slova ISBN-10 kódu, která splňují kontrolní schéma (4), jsou planým slovem ISBN-10 kódu. Jaká je velikost ISBN-10 kódu?

3.4. Vektorový prostor

V podkapitole 2.2. jsme ukázali, že při sestavování kódových slov můžeme provádět operace s jejich jednotlivými souřadnicemi. Můžeme sčítat i násobit příslušné symboly jako prvky číselného tělesa $F(q)$, kde q je prvočíslo, případně mocnina prvočísla. Nyní tento postup zobecníme. Budeme počítat s kódovými slovy jako s prvky vektorového prostoru, tj. budeme je (po složkách) sčítat a budeme je násobit prvkem tělesa $F(q)$ (násobit skalárem).

Operace s vektory
Předpokládejme, že q je mocnina prvočísla a $GF(q)$ je konečné těleso s n prvky.

Definice Množinu všech uspořádaných n-tic prvků z $GF(q)$ označme $V(n, q)$. Prvky množiny $V(n, q)$ budeme značit $\vec{x} = (x_1, x_2, \ldots, x_n)$ a budeme jim říkat *vektory*. Prvkům množiny $GF(q)$ budeme říkat *skaláry*.

Všimněte si, že slova i kódová slova můžeme chápat jako prvky vektorového prostoru a naopak vektory vektorového prostoru odpovídají slovům zavedeným v podkapitole 1.2. Zavedeme následující obvyklé operace.

Definice Mějme dva vektory $\vec{x}, \vec{y} \in V(n, q)$, kde $\vec{x} = (x_1, x_2, \ldots, x_n)$, $\vec{y} = (y_1, y_2, \ldots, y_n)$. *Součet vektorů* je operace na množině $V(n, q)$ definovaná následujícím způsobem.

$$\vec{x} + \vec{y} = (x_1 + y_1, x_2 + y_2, \ldots, x_n + y_n)$$

Násobení vektory skalárem $a \in GF(q)$ je operace $V \times GF(q) \to V$ definovaná následujícím způsobem.

$$a\vec{x} = (ax_1, ax_2, \ldots, ax_n)$$

Nyní můžeme vyslovit následující definici

Definice Množinu $V(n, q)$ pro nějaké přirozené číslo n a vhodné přirozené číslo q společně s tělesem $GF(q)$ a společně s operacemi sčítání vektorů a násobení vektorů skalárem nazveme *vektorový prostor* nad tělesem $GF(q)$.

Příklad 3.29. Uvedeme několik jednoduchých příkladů vektorových prostorů.

1) Množina $V(n, 2)$ binárních vektorů délky n spolu s operacemi sčítání vektorů po složkách a (triviálně definovaným) násobením 0 a 1 je vektorový prostor. Množina $V(n, 2)$ obsahuje 2^n vektorů.
2) Množina $V(n, q)$ všech q-árních vektorů délky n, kde q je mocnina prvočísla, spolu s operacemi sčítání vektorů po složkách a násobkem vektoru je vektorový prostor. Množina $V(n, q)$ obsahuje q^n vektorů.

3) Množina $V(3,3)$ je množina všech ternárních vektorů délky 3 spolu s operacemi sčítání vektorů po složkách a násobkem vektoru je vektorový prostor. Množina $V(3,3)$ obsahuje 27 vektorů.

Není těžké dokázat, že pro součet vektorů a pro součin skaláru s vektorem platí následující tvrzení.

Věta 3.9. *Pro každé $\vec{u},\vec{v},\vec{w} \in V(n,q)$ a každé $a,b \in GF(q)$ platí*

(i) $\vec{u}+\vec{v} \in V(n,q)$, *(uzavřenost)*
(ii) $\vec{u}+\vec{v} = \vec{v}+\vec{u}$, *(komutativita)*
(iii) $(\vec{u}+\vec{v})+\vec{w} = \vec{v}+(\vec{u}+\vec{w})$, *(asociativita)*
(iv) pro nulový vektor $\vec{0}=(0,0,\ldots,0)$ *platí* $\vec{0}+\vec{u}=\vec{u}$ *a* $\vec{u}+\vec{0}=\vec{u}$, *(nulový vektor)*
(v) opačný vektor *k vektoru* $\vec{u}$ *je vektor* $-\vec{u}=(-u_1,-u_2,\ldots,-u_n)$ *a platí* $\vec{u}+(-\vec{u})=\vec{0}$, *(opačný vektor)*
(vi) $a\vec{u} \in V(n,q)$, *(uzavřenost vzhledem k násobení skalárem)*
(vii) $a(\vec{u}+\vec{v}) = a\vec{u}+a\vec{v}$, *(distributivita vzhledem ke sčítání vektorů)*
(viii) $(a+b)\vec{u} = a\vec{u}+b\vec{u}$, *(distributivita vzhledem ke sčítání skalárů)*
(ix) $(ab)\vec{u} = a(b\vec{u})$, *(asociativita vzhledem k násobení skalárem)*
(x) $1\vec{u}=\vec{u}$, *kde* 1 *je jednička tělesa* $GF(q)$. *(invariance vektoru při násobení jedničkou)*

Důkaz vynecháme.

Vektorový podprostor

Podmnožina C množiny $V(n,q)$, která je sama vektorovým prostorem, se nazývá *vektorový podprostor*. Pokud podmnožina C obsahuje pouze vektor $\vec{0}$, tak C je *triviální* podprostor, v opačném případě máme *netriviální* vektorový podprostor.

Příklad 3.30. Uvedeme několik jednoduchých příkladů podprostorů.

1) V prostoru $V(n,q)$ označíme $A_i = \{(0,\ldots,0,x_i,0,\ldots,0) : x_i \in GF(q)\}$ pro $i=1,2,\ldots,n$. Množiny A_i tvoří podprostory v prostoru $V(n,q)$ (Příklad 3.31.). Množina A_i obsahuje q prvků.
2) Označme E_n množinu všech binárních vektorů sudé váhy v prostoru $V(n,2)$. Množina E_n je podprostorem $V(n,2)$. Množina E_n obsahuje 2^{n-1} prvků.
3) Označme L_n množinu všech binárních vektorů liché váhy v prostoru $V(n,2)$. Množina L_n není podprostorem $V(n,2)$.
4) Množina $C=\{\vec{0}\}$ je triviální podprostor každého vektorového prostoru, obsahuje jediný prvek.

V každém vektorovém prostoru $V(n,q)$ najdeme *triviální podprostor* $\{\vec{0}\}$ a *nevlastní podprostor* $V(n,q)$. Podprostor se nazývá *netriviální podprostor* prostoru $V(n,q)$, jestliže obsahuje alespoň jeden vektor různý od vektoru $\vec{0}$.

Abychom ověřili, že podmnožina $m \subseteq V(n,q)$ je vektorovým podprostorem, nemusíme ověřovat všechny vlastnosti definice vektorového prostoru.

Věta 3.10. *Mějme neprázdnou podmnožinu $M \subseteq V(n,q)$. M je vektorovým prostorem právě tehdy, když jsou splněny obě následující vlastnosti:*

(i) pro každé $\vec{u},\vec{v} \in M$ je $\vec{u}+\vec{v} \in M$,
(ii) pro každé $\vec{u} \in M$ a každé $a \in GF(q)$ je $a\vec{u} \in M$.

Důkaz.
„⇒“ Obě vlastnosti plynou ihned z definice vektorového prostoru.
„⇐“ Stačí ukázat, že ostatní vlastnosti plynou z faktu, že $V(n,q)$ je vektorovým prostorem. Důkaz vynecháme. □

Příklad 3.31. Ukážeme, že A_i je podprostorem prostoru $V(n,q)$.
Ověříme předpoklad Věty 3.10. Pro libovolné dva prvky $\vec{u}=(0,\ldots,0,u_i,0,\ldots,0)$, $\vec{v}=(0,\ldots,0,u_i,0,\ldots,0)$ z množiny A_i platí $\vec{u}+\vec{v} = (0,\ldots,0,u_i,0,\ldots,0)+(0,\ldots,0,v_i,0,\ldots,0) = (0,\ldots,0,u_i+v_i,0,\ldots,0) \in A_i$. Podobně pro libovolný prvek $\vec{u}=(0,\ldots,0,u_i,0,\ldots,0)$ z množiny A_i a libovolný prvek $a \in GF(q)$ platí $a\vec{u} = a(0,\ldots,0,u_i,0,\ldots,0) = (0,\ldots,0,a\cdot u_i,0,\ldots,0) \in A_i$. Podle Věty 3.10. je A_i podprostorem $V(n,q)$. ✓

Ověření, zda daná množina tvoří podprostor, můžeme ještě zjednodušit. Dvě podmínky Věty 3.10. je možno shrnout do podmínky jediné.

Věta 3.11. *Mějme neprázdnou podmnožinu $M \subseteq V(n,q)$. M je vektorovým prostorem právě tehdy, když je splněno, že pro každé $\vec{u}, \vec{v} \in M$ a každé $a, b \in GF(q)$ je $a\vec{u} + b\vec{v} \in M$.*

Důkaz je ponechán jako Cvičení 3.4.1.

Příklad 3.32. Mějme množinu $B_{ij} = \{(0,\ldots,0,x_i,0\ldots,0,\ldots,0,x_j,0,\ldots,0) : x_i, x_j \in GF(q)\}$ je vektorovým podprostorem $V(n,q)$. Ukážeme, že B_{ij} je podprostorem prostoru $V(n,q)$.
Pro libovolné dva prvky $\vec{u} = (0,\ldots,0,u_i,0,\ldots,0,u_j,0,\ldots,0)$, $\vec{v} = (0,\ldots,0,v_i,0,\ldots,v_j,0,\ldots,0)$ z množiny B_{ij} a libovolné dva prvky $a, b \in GF(q)$ platí pro $a\vec{u} + b\vec{v} = a(0,\ldots,0,u_i,0,\ldots,0,u_j,0,\ldots,0) + +b(0,\ldots,0,v_i,0,\ldots,0,v_j,0,\ldots,0) = (0,\ldots,0,a(u_i+v_i),0,\ldots,0,b(u_i+v_i),0,\ldots,0) \in B_{ij}$. Podle Věty 3.11. je B_{ij} podprostorem $V(n,q)$. ✓

Lineární kombinace
Připomeňme, že v tomto textu se soustředíme převážně na *konečné* vektorové prostory. Řada pojmů je však definována v obecném vektorovém prostoru.

Definice Mějme vektory $\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_r \in V(n,q)$ a skaláry $a_1, a_2, \ldots, a_r \in GF(q)$. Vektor $a_1\vec{u}_1 + a_2\vec{u}_2 + + \cdots + a_r\vec{v}_r$ nazveme *lineární kombinací* $\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_r$.

Není těžké ukázat následující tvrzení.

Věta 3.12. Lineární kombinace vektorů tvoří podprostor
Mějme vektory $\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_r \in V(n,q)$. Množina

$$\{a_1\vec{u}_1 + a_2\vec{u}_2 + \cdots + a_r\vec{v}_r : a_1, a_2, \ldots, a_r \in GF(q)\}$$

tvoří vektorový podprostor prostoru $V(n,q)$.

Důkaz. Tvrzení dostaneme opakovaným použitím Věty 3.11. □

Příklad 3.33. Uvedeme několik příkladů.

1) V prostoru $V(n,q)$ označíme vektor $\vec{u} = (0,\ldots,0,1,0,\ldots,0)$. Množina lineárních kombinací $\{a\vec{u} : a \in GF(q)$ tvoří podprostor A_i v prostoru $V(n,q)$. Množina A_i obsahuje q prvků.
2) V prostoru $V(n,q)$ označíme dvojici vektorů $\vec{u} = (1,,0,\ldots,0)$, $\vec{v} = (0,1,0,\ldots,0)$. Množina lineárních kombinací $B = \{a\vec{u} + b\vec{v} : a, b \in GF(q)$ tvoří podprostor B v prostoru $V(n,q)$. Množina B obsahuje q^2 prvků.
3) Označme E_n množinu všech binárních vektorů sudé váhy v prostoru $V(n,2)$. Množina E_n je podprostorem $V(n,2)$. Množina E_n obsahuje 2^{n-1} prvků.
4) Označme L_n množinu všech binárních vektorů liché váhy v prostoru $V(n,2)$. Množina L_n není podprostorem $V(n,2)$.
5) Množina $C = \{\vec{0}\}$ je triviální podprostor každého vektorového prostoru, obsahuje jediný prvek.

Další důležitý pojem je lineární nezávislost vektorů.

Definice Množina vektorů $\{\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_r\} \subseteq V(n,q)$ se nazývá *lineárně závislá*, jestliže existuje množina skalárů $a_1, a_2, \ldots, a_r \in GF(q)$ taková, že ne všechny skaláry jsou nulové a platí

$$a_1\vec{u}_1 + a_2\vec{u}_2 + \cdots + a_r\vec{u}_r = \vec{0}.$$

Naopak, množina vektorů $\{\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_r\} \subseteq V(n,q)$ se nazývá *lineárně nezávislá*, jestliže z rovnosti $a_1\vec{u}_1 + + a_2\vec{u}_2 + \cdots + a_r\vec{u}_r = \vec{0}$ plynou rovnosti $a_1 = a_2 = a_r = 0$.
Stručně mluvíme také o lineárně závislých a lineárně nezávislých vektorech.

Příklad 3.34. Uvedeme několik příkladů lineárně nezávislých množin vektorů.

1) Množina vektorů $\{(1,0,0,\ldots,0),(0,1,0,\ldots,0),\ldots,(0,\ldots,0,1)\}$ je množina n lineárně nezávislých vektorů v libovolném vektorovém prostoru $V(n,q)$ nad $GF(q)$.
2) Množina vektorů $\{(1,1,0),(1,0,1)\}$ je množina dvou lineárně nezávislých vektorů ve vektorovém prostoru $V(3,2)$ nad $GF(2)$.
3) Trojice vektorů $\{(1,2,0),(2,1,1),(0,2,0)\}$ je množina lineárně nezávislých vektorů v prostoru $V(4,3)$ (Cvičení 3.4.4.).
4) Označme E_n množinu všech binárních vektorů sudé váhy v prostoru $V(n,2)$. Množina E_n *nikdy* není množinou lineárně nezávislých vektorů, protože obsahuje vektor $\vec{0}$.

5) Označme X_n množinu všech nenulových binárních vektorů sudé váhy v prostoru $V(n,2)$. Množina X_n je množinou lineárně nezávislých vektorů pouze pro $n = 2$, neboť $X_2 = \{(1,1)\}$. Pro větší n množina X_n obsahuje trojici lineárně závislých vektorů $(0,1,1,0\ldots)$, $(1,0,1,0\ldots)$, $(1,1,0,0\ldots)$,

Báze

Množina M vektorů z prostoru $V(n,q)$ může, ale nemusí být vektorovým prostorem. Má však smysl zkoumat nejmenší takový podprostor prostoru $V(n,q)$, který množinu M obsahuje. Dostáváme tak přirozeně následující pojem.

Definice **Generující množina**
Mějme množinu vektorů $M = \{\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_r\}$ prostoru $V(n,q)$. Mějme podprostor C prostoru $V(n,q)$ takový, že každý vektor $\vec{v} \in C$ může být vyjádření jako lineární kombinace vektorů množiny M. Množině M říkáme *generující množina* podprostoru C.

Příklad 3.35. Uvedeme několik příkladů generujících množin vektorů.

1) Množina vektorů $\{(1,0,0,\ldots,0),(0,1,0,\ldots,0),\ldots,(0,\ldots,0,1)\}$ je generující množina vektorového prostoru $V(n,q)$ nad $GF(q)$.
2) Množina vektorů $\{(1,1,0),(1,0,1)\}$ je generující množina vektorového prostoru E_2, není však generující množinou vektorového prostoru $V(3,2)$.
3) Množina E_n i množina X_n jsou generující množiny vektorového prostoru E_n, nejsou však generujícími množinami prostoru $V(n,2)$.
4) Množina L_n je generující množina prostoru $V(n,2)$.

Jestliže generující množina vektorového podprostoru je navíc množina lineárně nezávislých vektorů, tak máme bázi podprostoru.

Definice Mějme množinu vektorů $M = \{\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_r\}$ prostoru $V(n,q)$. Jestliže M je lineárně nezávislá generující množina nějakého podprostoru C, tak M nazýváme *báze* podprostoru C.

Příklad 3.36. Uveďme několik jednoduchých příkladů bází.

1) Kanonická báze celého prostoru $V(n,q)$ je tvořena množinou vektorů

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

2) Množina vektorů $\{(1,0,0),(1,1,0),(1,1,1)\}$ tvoří bázi prostoru $V(3,q)$ pro libovolné q.

Obecnější příklady jsou uvedeny ve cvičeních.

Otázka: Má každý prostor bázi?

Následující tvrzení ukazuje, že z každé generující množiny je možno vybrat bázi. Všimněte si, že tvrzení sice platí i pro nekonečně velké vektorové prostory, je však formulováno a dokázáno jen pro konečné prosotry.

Věta 3.13. *Mějme netriviální podprostor C prostoru $V(n,q)$. Každá konečná generující množina M podprostoru C obsahuje bázi C.*

Důkaz. Tvrzení dokážeme přímo. Mějme množinu vektorů $M = \{\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_r\}$ podprostoru C prostoru $V(n,q)$. Jestliže M je množina nezávislých vektorů, tak M je hledaná báze.

Jestliže M je množina závislých vektorů, tak podle definice lineární závislosti existuje množina skalárních prvků $a_1, a_2, \ldots, a_r \in GF(q)$ takových, že ne všechny jsou nulové a platí

$$a_1\vec{u}_1 + a_2\vec{u}_2 + \cdots + a_r\vec{u}_r = \vec{0}.$$

Jestliže $a_j \neq 0$, tak v $GF(q)$ existuje a_j^{-1} a můžeme vyjádřit

$$\vec{u}_j = a_j^{-1}\left(\sum_{i=1, i\neq j}^{r} a_i\vec{u}_i\right), \tag{5}$$

a proto vektor $\vec{u}_j$ je lineární kombinací zbývajících vektorů a můžeme jej z generující množiny M vynechat. Dostaneme množinu M', která je stále generující množinou podprostoru C, neboť v každém vyjádření libovolného vektoru $\vec{v} \in C$ můžeme nahradit $\vec{u}_j$ vztahem (5), čímž vektor $\vec{u}_j$ nebude v množině generujících vektorů potřeba.

Jestliže M' je množina lineárně nezávislých vektorů, tak máme hledanou bázi. Pokud je M' množina závislých vektorů, opět některý z vektorů vynecháme. Celý postup je konečný, neboť množina M je konečná. Po konečném počtu kroků dostaneme množinu lineárně nezávislých vektorů (jistě alespoň jeden nenulový vektor v množině M' zůstane) a máme hledanou bázi. □

Předchozí věta říká, že báze je v jistém smyslu minimální generující množina. Abychom vyjádřili všechny vektory podprostoru C, potřebujeme všechny vektory báze, žádný není nadbytečný. Následující tvrzení ukazuje, že v takovém případě je nejen možno každý vektor podprostoru C vyjádřit jako lineární kombinaci vektorů báze, ale navíc je toto vyjádření jednoznačné až na pořadí vektorů.

Věta 3.14. Věta o jednoznačném vyjádření v bázi
Mějme bázi $\{\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_k\}$ podprostoru C prostoru $V(n,q)$. Potom

(i) každý vektor $\vec{v} \in C$ může být vyjádřen jako lineární kombinace vektorů $\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_k$ jednoznačně až na pořadí vektorů,
(ii) podprostor C obsahuje právě q^k vektorů.

Důkaz. Ukážeme postupně obě části tvrzení.
(i) Postupujeme sporem. Podle definice víme, že každý vektor $\vec{v} \in C$ umíme vyjádřit pomocí vektorů báze $\{\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_k\}$. Předpokládejme, že máme dvě různá vyjádření vektoru $\vec{v} \in C$ v bázi $\{\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_k\}$, například $\vec{v} = a_1\vec{u}_1 + a_2\vec{u}_2 + \cdots + a_k\vec{u}_k$ a $\vec{v} = b_1\vec{u}_1 + b_2\vec{u}_2 + \cdots + b_k\vec{u}_k$, kde alespoň některé koeficienty se liší, tj. $a_i \neq b_i$ pro nějaké $i = 1, 2, \ldots, k$. Odečtením obou vyjádření dostaneme

$$(a_1 - b_1)\vec{u}_1 + (a_2 - b_2)\vec{u}_2 + \cdots, +(a_k - b_k)\vec{u}_k = \vec{0}.$$

Koeficient $a_i - b_i \neq 0$. Dostáváme spor, neboť vektory báze $\{\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_k\}$ jsou lineárně nezávislé. To znamená, že $a_i = b_i$ pro všechna $i = 1, 2, \ldots, k$, a proto je vyjádření každého vektoru $\vec{v}$ v bázi $\{\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_k\}$ jednoznačné.
(ii) Existuje q^k různých k-tic skalárních prvků $a_1, a_2, \ldots, a_k \in GF(q)$. Každá taková k-tice dává podle již dokázané části (i) jinou lineární kombinaci vektorů báze $\{\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_k\}$. Proto podprostor C obsahuje právě q^k různých vektorů. □

Každé dvě báze daného podprostoru mají stejný počet vektorů, neboť podprostor C má právě q^k prvků, kde k je počet vektorů báze. Můžeme mít různé báze téhož podprostoru, všechny však mají stejný počet k vektorů. Nyní můžeme vyslovit následující definici. Její korektnost vyplývá z Věty 3.14.

Definice Mějme podprostor C prostoru $V(n,q)$. Počet vektorů báze podprostoru C nazýváme *dimenze* prostoru C. Značíme ji $\dim(C)$.

Příklad 3.37. Uvedeme několik jednoduchých příkladů dimenze.

1) Prostor $V(n,q)$ má dimenzi n.
2) Podprostor E_n prostoru $V(n,q)$ má dimenzi $n-1$ (Cvičení 3.4.3.).
3) Podprostory A_i prostoru $V(n,q)$ z Příkladu 3.30. mají dimenzi 1 (Cvičení 3.4.5.).

Cvičení

3.4.1. Dokažte Větu 3.11.

3.4.2. Najděte příklad báze podprostoru E_n prostoru $V(n,2)$.

3.4.3. Ukažte, že dimenze podprostoru E_n prostoru $V(n,2)$ je $n-1$.

3.4.4. Ukažte, že a) trojice vektorů $\{(1,2,0),(2,1,1),(0,2,0)\}$ tvoří množinu lineárně nezávislých vektorů v prostoru $V(4,3)$, b) $\{(1,2,0),(2,1,1),(0,0,2)\}$ není lineárně nezávislá množina vektorů.

3.4.5. Určete dimenzi podprostoru A_i prostoru $V(n,q)$ z Příkladu 3.30.

Kapitola 4. Úvod do lineárních kódů

V této kapitole spojíme poznatky z předchozích kapitol. Budeme sestavovat kódy, jehož kódová slova budou současně vektory nějakého vektorového prostoru. Ukážeme, jaké výhody takové kódy přináší.

4.1. Pojem lineárního kódu

Mějme konečné těleso $GF(q)$. V celé kapitole budeme předpokládat, že $GF(q)$ je současně tělesem s q prvky (to znamená, že q je přirozená mocnina nějakého prvočísla) a současně tvoří abecedu F_q nějakého kódu. Nyní ospravedlníme značení z předchozích kapitol, že $\vec{x}$ značilo jednak slovo množiny $(F_q)^n$ (nebo dokonce kódové slovo nějakého kódu), ale současně označovalo také vektor nějakého vektorového prostoru. Vektory budeme v dalším textu považovat za kódová slova a naopak. Bude tedy mít smysl kódová slova sčítat a dokonce násobit skalárem, tj. prvkem z $GF(q)$.

Připomeňme, že $\vec{x} = (x_1, x_2, \ldots, x_n)$ budeme stručně zapisovat jako $x_1x_2\ldots x_n$. Můžeme mluvit o znacích slova $\vec{x}$, ale současně o souřadnicích vektoru $\vec{x}$.

Definice Mějme vektorový prostor $V(n, q)$ nad tělesem $GF(q)$ pro nějaké přirozené číslo n. Jeho podprostor C nazveme *lineárním kódem*.

Uvědomte si, že podle Věty 3.10. je podmnožina C prostoru $V(n, q)$ lineárním kódem právě tehdy, když

(i) pro každé $\vec{u}, \vec{v} \in C$ je $\vec{u} + \vec{v} \in C$,
(ii) pro každé $\vec{u} \in C$ a každé $a \in GF(q)$ je $a\vec{u} \in C$.

Jinými slovy: součet každých dvou kódových slov daného lineárního kódu je opět kódovým slovem a dokonce násobek kódového slova je opět kódovým slovem. A naopak pokud součet každých dvou kódových slov je také kódovým slovem a násobek každého kódového slova je kódovým slovem, tak daný kód je lineární. Vektorový prostor a lineární kód jsou stejným objektem, který můžeme zkoumat z různých hledisek.

Příklad 4.1. Uvedeme několik jednoduchých příkladů.

1) Kód C_1 z Příkladu 1.9. je lineární kód.

$$\text{kód } C_1 = \begin{cases} 00 \\ 01 \\ 10 \\ 11. \end{cases}$$

Všechna čtyři kódová slova tvoří vektorový prostor $V(2, 2)$. Všimněte si, že libovolná lineární kombinace kódových slov je opět kódovým slovem kódu C_1.

2) Kód C_2 z Příkladu 1.10. je lineární kód.

$$\text{kód } C_2 = \begin{cases} 000 \\ 011 \\ 101 \\ 110. \end{cases}$$

Součet (a triviálně i násobek) kódových slov je opět kódovým slovem.

3) Kód C_3 z Příkladu 1.11. je lineární kód. Součet i násobek kódových slov je opět kódovým slovem.

$$\text{kód } C_3 = \begin{cases} 00000 \\ 01110 \\ 10101 \\ 11011. \end{cases}$$

4) Kód C_F z Příkladu 3.5. je lineární kód. Součet i násobek kódových slov je opět kódovým slovem.

$$\begin{aligned} \text{kód } C_F = \{&0000000, 1000101, 1100010, 0110001, 1011000, 0101100, 0010110, 0001011, \\ &1111111, 0111010, 0011101, 1001110, 0100111, 1010011, 1101001, 1110100.\} \end{aligned}$$

5) Každý q-ární opakovací kód délky n je lineární kód.
6) Kód, ve kterém je každý vektor prostoru $V(n,q)$ kódovým slovem, je lineární kód.

Pochopitelně ne každý kód je lineárním kódem.

Příklad 4.2. Uvedeme několik příkladů kódů, které nejsou lineární.

1) Kód C_2', který dostaneme z kódu C_2 z Příkladu 1.10. vynecháním posledního kódového slova není lineární kód.
$$\text{kód } C_2' = \begin{cases} 000 \\ 011 \\ 101. \end{cases}$$
Už se nejedná o vektorový podprostor. Není pravda, že lineární kombinace dvou kódových slov je kódovým slovem. Například $011 + 101 = 110$, ale $110 \notin C_2'$.
2) Kód C_3', který vznikne z kódu C_3 z Příkladu 1.11. přidáním kódového slova 00111 není lineární kód.
$$\text{kód } C_3' = \begin{cases} 00000 \\ 01110 \\ 10101 \\ 11011 \\ 00111. \end{cases}$$
Opět se nejedná o vektorový podprostor. Není pravda, že lineární kombinace dvou kódových slov je kódovým slovem. Například $11011 + 00111 = 11100$, ale $11100 \notin C_3'$.
3) Kód C', který vznikne z lineárního opakovacího kódu C_4 délky 4 záměnou posledního bitu není lineární kód.
$$\text{kód } C_4' = \begin{cases} 0001 \\ 1110. \end{cases}$$
Už se nejedná o vektorový podprostor. Není pravda, že lineární kombinace dvou kódových slov je kódovým slovem. Například $0001 + 1110 = 1111$, ale $1111 \notin C_4'$. Navíc ne každý násobek kódového slova se nachází v kódu C_4'. Například $0 \cdot 0001 = 0000$, ale $0000 \notin C_4'$.
4) Žádný kód sestavený ze slov v desítkové soustavě není lineárním kódem, neboť neexistuje žádný vektorový prostor nad $GF(10)$.

Dimenze lineárního kódu

Dimenze podprostoru, který je lineárním kódem, je důležitým parametrem, který daný lineární kód charakterizuje.

Definice Mějme nějaký podprostor C dimenze k prostoru $V(n,q)$ nad $GF(q)$. Lineární kód C budeme značit $[n,k]$-kód. Číslo k nazýváme *dimenze* lineárního kódu C. Pokud d je minimální vzdálenost kódu C, budeme kód někdy značit $[n,k,d]$-kód.

Všimněte si rozdílných závorek: každý q-ární lineární $[n,k,d]$-kód je současně (n,q^k,d)-kód. S využitím Věty 3.14. víme, že $[n,k,d]$-kód má právě q^k kódových slov.

Je dobrá si uvědomit, že opačná implikace neplatí. Ne každý (n,q^k,d)-kód je současně lineárním $[n,k,d]$-kódem. Například kód C_4' z Příkladu 4.2. je $(4,2^1,4)$-kód, avšak nejedná se o lineární kód.

Příklad 4.3. Uvedeme několik jednoduchých příkladů.

1) Kód C_1 z Příkladu 1.9. je lineární $[2,2]$-kód, respektive $[2,2,1]$-kód, neboť $d = \operatorname{dist} C_1 = 1$. Všechna čtyři kódová slova tvoří celý vektorový prostor $V(2,2)$ dimenze 2.
2) Kód C_2 z Příkladu 1.10. je lineární $[3,2]$-kód, respektive $[3,2,2]$-kód, neboť $d = \operatorname{dist} C_2 = 2$. Všechna čtyři kódová slova tvoří vektorový podprostor dimenze 2 ve vektorovém prostoru $V(3,2)$ dimenze 3.
3) Kód C_3 z Příkladu 1.11. je lineární $[5,2]$-kód, respektive $[5,2,3]$-kód. Všechna čtyři kódová slova tvoří vektorový podprostor dimenze 2 ve vektorovém prostoru $V(5,2)$ dimenze 5.
4) Kód C_F z Příkladu 3.5. je lineární $[7,4]$-kód, respektive $[7,4,3]$-kód, neboť v Příkladu 3.5. jsme ukázali, že $d = \operatorname{dist} C_F = 3$. Všech 16 kódových slov tvoří vektorový podprostor dimenze 4 ve vektorovém prostoru $V(7,2)$ dimenze 7.
5) Každý q-ární opakovací kód délky n je lineární $[n,1]$-kód, respektive $[n,1,n]$-kód. Všech q kódových slov tvoří podprostor dimenze 1 vektorového prostoru $V(n,q)$ dimenze n.

Otázka: Mohou existovat dva lineární $[n, k]$-kódy C a C', které mají různou minimální vzdálenost d?

Příklad 4.4. Binární kód C_5 z Příkladu 2.3. není lineární.

$$\text{kód } C_5 = \begin{cases} 10000 \\ 11110 \\ 00011 \\ 01101 \end{cases}$$

Všimněte si, že kód C_5 neobsahuje kódové slovo $\vec{0} = 00000$, a proto 0 násobek žádného kódového slova *nepatří* do množiny kódových slov C_5. To znamená, že množina kódových slov kódu C_5 netvoří vektorový prostor a nejedná se o lineární kód. Nemá smysl hovořit o dimenzi kódu C_5, má však smysl určit jeho minimální vzdálenost. Ověřením všech šesti dvojic kódových slov vidíme, že $\operatorname{dist} C_5 = 3$.

Pozorování z předchozího příkladu můžeme použít pro snadné rozhodnutí u některých kódů, že se o lineární kód nejedná.

Poznámka 4.1. Vektor $\vec{0}$ vždy patří do podprostoru C, což znamená, že kódové slovo $\vec{0}$ patří do každého lineárního kódu. Velikost q-árního lineárního $[n, k]$-kódu je $M = q^k$.

Váha lineárního kódu

Ukážeme, že u lineárních kódů je snadné určit jejich minimální vzdálenost. Váhu vektoru $\vec{u}$ ve vektorovém prostoru definujeme stejně jako váhu slova $\vec{u}$, tj. počet nenulových souřadnic vektoru $\vec{u}$.

Definice Mějme lineární kód C. *Váha lineárního kódu* C je nejmenší váha nenulového kódového slova a značíme ji $\mathrm{w}(C)$.

Definici můžeme zapsat symbolicky

$$\mathrm{w}(C) = \min \left\{ \mathrm{w}(u) : \vec{u} \in C, \vec{u} \neq \vec{0} \right\}.$$

Nejprve ukážeme, že vzdálenost dvou kódových slov lineárního kódu můžeme snadno vyjádřit jako váhu jejich rozdílu.

Lemma 4.1. Lemma o vzdálenosti
Mějme libovolné dva vektory $\vec{x}, \vec{y} \in V(n, q)$. Vzdálenost odpovídajících kódových slov $\vec{x}$ a $\vec{y}$ je $\operatorname{dist}(\vec{x}, \vec{y}) = \mathrm{w}(\vec{x} - \vec{y})$.

Důkaz. Důkaz tvrzení je snadný. Vzdálenost dvou vektorů odpovídá počtu rozdílných souřadnic, tj. počet souřadnic, kde se $\vec{x}$ a $\vec{y}$ liší. □

Všimněte si, že pro $q = 2$ tvrzení Věty 4.1. odpovídá tvrzení Lemmatu 2.3. Pro $q > 2$ je tvrzení Věty 4.1. obecnější.

Minimální vzdálenost lineárního kódu

Nyní můžeme snadno vyjádřit minimální vzdálenost lineárního kódu jako jeho váhu.

Věta 4.2. Minimální vzdálenost a váha lineárního kódu
Mějme lineární kód C. Platí $\operatorname{dist}(C) = \mathrm{w}(C)$.

Důkaz. Rovnost ukážeme tak, že odvodíme dvě opačné nerovnosti.

V kódu C existuje dvojice kódových slov $\vec{x}$, $\vec{y}$, jejichž vzdálenost je rovna minimální vzdálenosti kódu C. V lineárním kódu je jejich rozdíl opět kódovým slovem a podle Lemmatu 4.1. a tento rozdíl nemůže mít menší váhu než má jakékoliv nenulové kódové slovo. Platí $\operatorname{dist}(C) = \operatorname{dist}(\vec{x}, \vec{y}) = \mathrm{w}(\vec{x} - \vec{y}) \geq \mathrm{w}(C)$.

Dále si uvědomíme, že minimální vzdálenost (každého, nejen lineárního) kódu C je nejvýše tak velká, jako je vzdálenost kódových slov od nulového kódového slova $\vec{0}$. Je-li $\vec{x}$ slovo, které realizuje minimální vzdálenost kódu C, tak platí $\operatorname{dist}(C) \leq \operatorname{dist}(\vec{x}, \vec{0}) = \mathrm{w}(\vec{x}) = \mathrm{w}(C)$.

Celkem dostáváme, že v každém lineárním kódu platí $\operatorname{dist}(C) = \mathrm{w}(C)$. □

Výhody lineárních kódů

Věta 4.2. říká, že zatímco u obecného kódu s M kódovými slovy, kde $M = |C|$, je pro určení minimální vzdálenosti prověřit všech $\binom{M}{2}$ vzdáleností dvojic kódových slov, tak u lineárního kódu stačí porovnat váhy všech $M - 1$ nenulových kódových slov. Pro velké kódy je ověření mnohem snadnější.

Další podstatnou výhodou je, že existují jednoduché algoritmy pro kódování i pro dekódování lineárních kódů. Budeme se jim věnovat v Kapitole 5.

A další výhodou je, že zatímco u obecného kódu pro popis kódu musíme zadat všech M kódových slov, u lineárního kódu stačí popsat vhodně vybraná kódová slova: bázi vektorového prostoru. Ostatní kódová slova můžeme dopočítat, vyjádříme je jako všechny možné lineární kombinace kódových slov báze. S tím souvisí i následující definice.

Generující matice lineárního kódu

Definice Mějme lineární $[n, k]$-kód C. Matice rozměru $k \times n$, jejíž řádky jsou tvořeny souřadnicemi vektorů báze lineárního kódu C, se nazývá *generující matice* kódu C.

Příklad 4.5. Uvedeme několik jednoduchých příkladů generujících matic.

1) Generující matice lineárního $[2, 1]$-kódu C_1 z Příkladu 1.9. je

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

2) Generující matice lineárního $[3, 2]$-kódu C_2 z Příkladu 1.10. je

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

3) Generující matice lineárního $[5, 2]$-kódu C_3 z Příkladu 1.11. je

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

4) Generující matice lineárního $[6, 2]$-kódu C_3' z Příkladu 2.9. je

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

5) Generující matice lineárního $[7, 4]$-kódu C_F z Příkladu 3.5. je

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

6) Generující matice q-árního opakovacího lineární $[n, 1]$-kódu délky n je

$$[\,1 \quad 1 \quad \dots \quad 1\,].$$

Příklad 4.6. Kód C_5 z Příkladu 2.3. není lineární, proto nemá smysl sestavovat jeho generující matici.
Tady chybí část textu, kterou je třeba doplnit.

Nevýhody lineárních kódů

Lineární kódy jsou podprostory vektorového prostoru $V(n, q)$ a takový prostor existuje pouze, pokud q je těleso, tedy pokud q je mocnina nějakého prvočísla.

Lineární binární kódy pochopitelně existují, neboť $(\mathbb{Z}_2, +, \cdot)$ je těleso. Avšak kódy v desítkové soustavě nemohou být lineární, neboť neexistuje konečné těleso řádu 10. Tuto nevýhodu lze zmírnit podobně jako jsme ukázali na straně 34 při popisu ISBN kódu. Sestavíme kód na množině s větší abecedou (například s jedenáctkovou abecedou), avšak použijeme pouze kódová slova, která mají souřadnice v desítkové soustavě. Slova, která využívají symbol 10 (nebo X) prostě z kódu vynecháme. Vynechání kódových slov však snižuje velikost kódu M a počet slov takového kódu nemůže dosáhnout hodnoty $A_q(n, d)$.

Stejný problém by mohl teoreticky nastat i u obecného lineárního kódu. Víme, že hodnota $A_q(n, d)$ udává největší velikost kódu délky n s minimální vzdáleností d. Omezíme-li se na lineární kódy, tak takové kódy

by nemusely hranice $A_q(n,d)$ dosáhnout. Zpravidla se však ukazuje, že největší hodnoty M lze dosáhnout i v rámci lineárních kódů. Dokonce všechny netriviální perfektní kódy jsou lineární kódy.

Cvičení

4.1.1. Sestavte generující matici lineárního $[n,1,1]$-kódu.

4.1.2. Je kód C_H ze Cvičení 3.1.7. lineární? Pokud ano, najděte nějakou jeho bázi.

4.2. Ekvivalence lineárních kódů

V podkapitole 2.1. jsme zavedli pojem ekvivalence kódů. V tomto smyslu umíme rozhodnout, zda dva lineární kódy jsou ekvivalentní. Nyní definice ekvivalence kódů pro lineární kódy zpřísníme. Aby dva lineární kódy byly ekvivalentní, dovolíme pouze takové permutace symbolů na pevně zvolené souřadnici, kterou dostaneme vynásobením nenulovým skalárem z tělesa $GF(q)$. Dostáváme tak následující definici.

Definice Ekvivalence lineárních kódů
Mějme dva q-ární lineární kódy C, C' nad tělesem $GF(q)$ stejné délky se stejným počtem slov. Řekneme, že kódy C a C' jsou *ekvivalentní lineární kódy*, jestliže kód C' může být získán z kódu C pomocí dvou typů operací:

(i) permutací souřadnic $a_1, a_2, \ldots, a_n$ všech slov kódu,
(ii) násobení symbolů na pevně zvolené souřadnici kódu nenulovým skalárem z tělesa $GF(q)$.

Všimněte si, že pro ekvivalentní lineární kódy nedovolíme libovolnou permutaci symbolů na pevně zvolené souřadnici, ale pouze takové permutace, které obdržíme vynásobením některým nenulovým prvkem z $GF(q)$.

Příklad 4.7. Jestliže máme dva binární lineární kódy C a C', tak rozhodnout o jejich ekvivalenci znamená prověřit pouze permutace sloupců popsané vlastností (i) v definici, neboť je přípustná pouze jediná permutace prvků na každé pevně zvolené souřadnici: násobení nenulovým prvkem je násobení prvkem 1.

Příklad 4.8. Mějme binární kód C_5 z Příkladu 2.3.

$$\text{kód } C_5 = \begin{cases} 10000 \\ 11110 \\ 00011 \\ 01101. \end{cases}$$

Kód C_5 je ekvivalentní kódu C_3 z Příkladu 1.11., avšak *není* ekvivalentní jako lineární kód, neboť kód C_5 není lineární (neobsahuje kódové slovo $\vec{0}$).

Příklady dvou ekvivalentních ternárních kódů a neekvivalentních lineárních kódů.

Tady chybí část textu, kterou je třeba doplnit.

Generující matice ekvivalentních lineárních kódů
Na straně 43 jsme zavedli generující matici lineárního kódu. Následující věta ukazuje, jak spolu souvisí generující matice dvou ekvivalentních lineárních kódů.

Věta 4.3. Věta o generujících maticích ekvivalentních lineárních kódů
Dvě generující matice rozměru $k \times n$ generují ekvivalentní lineární $[n,k]$-kódy nad tělesem $GF(q)$, jestliže jedna matice může být z druhé matice získána posloupností následujících kroků:
(R1) libovolná permutace řádků,
(R2) roznásobení řádku nenulovým skalárem z tělesa $GF(q)$,
(R3) přičtení libovolné lineární kombinace ostatních řádků k jinému řádku,
(S1) libovolná permutace sloupců,
(S2) roznásobení sloupce nenulovým skalárem z tělesa $GF(q)$.

Důkaz. Řádkové operace (R1), (R2) a (R3) zachovají lineární nezávislost řádků generující matice. Nahradí jednu bázi kódu stejnou nebo jinou bází stejného kódu se stejným počtem vektorů. Upravená matice generuje totožný podprostor vektorového prostoru, generuje proto stejný kód.

Sloupcové operace (S1) a (S2) převedou matici na generující matici ekvivalentního kódu. Jediný rozdíl spočívá v tom, že nedovolíme libovolnou permutaci symbolů ve sloupci, ale pouze takové permutace, které dostaneme roznásobením nenulovým skalárem tělesa. □

Ekvivalentní úpravy popsané ve Větě 4.3. umožňují generující matici upravit na vhodný tvar, ideálně na takový tvar, který je přehledný a umožní snadný proces kódování. S tím souvisí následující definice.

Definice Mějme generující matici B lineárního $[n,k]$-kódu o rozměru $k \times n$. Řekneme, že matice B je v *standardním tvaru*, jestliže má tvar $[\,I_k|A\,]$, kde I_k je jednotková matice řádu k a A je nějaká matice o rozměru $k \times (n-k)$.

Příklad 4.9. Uvedeme několik příkladů generujících matic a generujících matic v základním tvaru.

1) Generující matice lineárního $[2,1]$-kódu C_1 z Příkladu 1.9. je

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

Tato matice není v základním tvaru. Vezmeme-li oba řádky v opačném pořadí, dostaneme generující matici *stejného* lineárního $[2,1]$-kódu C_1, která je v základním tvaru.

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

2) Generující matice q-árního opakovacího lineární $[n,1]$-kódu délky n je triviálně v základním tvaru.

$$[\,1 \quad 1 \quad \dots \quad 1\,].$$

Následující věta říká, že generující matici každého lineárního kódu můžeme pomocí úprav popsaných ve Větě 4.3. převést na standardní tvar. Místo libovolného lineárního kódu tak vždy můžeme vzít ekvivalentní lineární kód, jehož generující matice je ve standardním tvaru.

Věta 4.4. Úprava generující matice na standardní tvar
Mějme generující matici B lineárního q-árního $[n,k]$-kódu. Provedením operací (R1), (R2), (R3) (S1) a (S2) může být generující matice B převedena do standardního tvaru $[\,I_k|A\,]$.

Důkaz. Důkaz je přímý, konstruktivní. Pro názornost označme $\vec{r}_1, \vec{r}_2, \dots, \vec{r}_k$ řádky matice B a dále označme $\vec{s}_1, \vec{s}_2, \dots, \vec{s}_n$ sloupce matice B. Prvky matice B označme $b_{i,j}$, hodnota b_{ij} se mění v průběhu operací.

Opakováním dále uvedených kroků pro každý sloupec $j = 1, 2, \dots, k$ upravíme prvních k sloupců matice B do požadovaného tvaru, kdy prvek $b_{jj} = 1$ a $b_{ij} = 0$ pro $i \neq j$.

V dalším předpokládejme, že matice B má již prvních $j-1$ sloupců upraveno do požadovaného tvaru (6).

$$B = \begin{bmatrix} 1 & 0 & \cdots & 0 & b_{1j} & \cdots & b_{1n} \\ 0 & 1 & \cdots & 0 & b_{2j} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & b_{j-1,j} & \cdots & b_{j-1,n} \\ 0 & 0 & \cdots & 0 & b_{jj} & \cdots & b_{jn} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & b_{kj} & \cdots & b_{kn} \end{bmatrix} \tag{6}$$

Krok 1
Jestliže prvek $b_{jj} \neq 0$, pokračuj krokem 2. Jestliže prvek $b_{jj} = 0$ a pro nějaké $i > j$ je prvek $b_{ij} \neq 0$, pomocí operace (R1) prohodíme řádky $\vec{r}_i$ a $\vec{r}_j$. Jestliže prvek $b_{jj} = 0$ a pro každé $i > j$ je prvek $b_{ij} = 0$, tak najdeme takový sloupec $k > j$, že $b_{hj} \neq 0$ a pomocí operace (C1) prohodíme sloupce $\vec{s}_i$ a $\vec{s}_j$.
Krok 2
Víme, že $b_{jj} \neq 0$ a pomocí operace (R2) vynásobíme (celý) řádek $\vec{r}_j$ inverzním prvkem b_{jj}^{-1} (víme, že takový prvek b_{jj}^{-1} v tělese $GF(q)$ existuje) a dostaneme generující matici, ve které $b_{jj} = 1$.
Krok 3

Už víme, že $b_{jj} = 1$ a pomocí operace (R3) odečteme od každého řádku $\vec{r}_i$, kde $i \neq j$, b_{ij} násobek řádku $\vec{r}_j$ (každý řádek $\vec{r}_i$ pro $i \neq j$ nahradíme lineární kombinací $\vec{r}_i - b_{ij}\vec{r}_j$).

Opakováním Kroku 3 pro každý řádek $\vec{r}_j$ bude mít celý sloupec $\vec{s}_j$ požadovaný tvar $\vec{s}_{jj} = 1$ a $\vec{s}_{ij} = 0$ pro $i \neq j$. Opakováním celého postupu pro prvních k sloupců dostaneme požadovaný standardní tvar matice B. □

Poznámka 4.2. Všimněte si, že v důkazu Věty 4.4. jsme použili úpravy (R1), (R2), (R3) a (S1). Jestliže během úpravy generující matice B nebude nikdy potřeba nepoužít permutaci sloupců – operace (S1) – tak výsledná generující matice ve standardním tvaru bude generovat *stejný* kód, jako matice A.

Jestliže však prvních k sloupců generující matice nebude lineárně nezávislých, tak úpravu (S1) budeme muset použít a výsledná generující matice ve standardním tvaru bude generovat *ekvivalentní* kód, ne nutně stejný.

Operaci (S2) nemusíme použít, ale pokud by během upravování nastala situace, že bude šikovné ji použít, tak výsledná generující matice bude opět generující maticí ekvivalentního kódu. Celý postup v důkazu Věty 4.4. je navržený tak, aby pokud možno vedl k sestavení generující matice ekvivalentního kódu.

Také je dobré si uvědomit, že generující matice kódu není určena jednoznačně, ani standardní tvar není určen jednoznačně, neboť permutací řádků před procesem úpravy můžeme dostat jinou generující matici stejného kódu.

Poznámka 4.3. Všimněte si, že uvedený postup známe z lineární algebry, kde se používá například při hledání inverzních matic k regulárním maticím. Při hledání matice inverzní k regulární matici A sestavíme pomocnou matici $[\,A|I\,]$ a postupem uvedeným výše dostaneme matici $[\,I|B\,]$, kde $B = A^{-1}$. Nemáme však dovoleno prohazovat sloupce!

Pokud matice A není regulární, tak řádky matice jsou lineárně závislé a nepodaří se levou část upravit na jednotkovou matici. V případě generující matice lineárního kódu bude všech k řádků lineárně nezávislých, protože generující matice je sestavena z vektorů báze vektorového prostoru, který odpovídá danému lineárnímu kódu a podle definice jsou řádky generující matice A lineárně nezávislé. Navíc, pokud by prvních k sloupců nebylo lineárně nezávislých, máme dovoleno prohazovat sloupce.

Tady chybí část textu, kterou je třeba doplnit.

Příklad 4.10. Ukážeme několik jednoduchých příkladů generujících matic a jak je můžeme upravit do standardního tvaru. Navážeme na Příklad 4.5.

1) Generující matice lineárního $[2, 1]$-kódu C_1 z Příkladu 1.9. je

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

Prohozením řádků ihned dostaneme generující matici ve standardním tvaru.

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

2) Generující matice lineárního $[3, 2]$-kódu C_2 z Příkladu 1.10. je

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

Prohozením řádků ihned dostaneme generující matici ve standardním tvaru.

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

3) Generující matice lineárního $[5, 3]$-kódu C_3' z Příkladu 1.11. je

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

První dva sloupce nejsou lineárně nezávislé. Při převodu na standardní tvar musíme použít i permutace sloupců a budeme generovat ekvivalentní kód, nikoliv stejný kód. Prohozením prvního a čtvrtého sloupce dostaneme generující matici

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

Odpovídající kód je právě kód C_3.

$$\text{kód } C_3 = \begin{cases} 00000 \\ 10101 \\ 01110 \\ 11011 \end{cases}$$

4) Generující matice lineárního $[7, 4]$-kódu C_F z Příkladu 3.5. je

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

Nejprve řádek přemístíme na konec a dále přičtením prvního řádku k druhému a čtvrtému řádku dostaneme matici

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \sim \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}.$$

Dále přičteme druhý řádek ke třetímu a čtvrtému řádku a potom třetí řádek ke čtvrtému řádku. Dostaneme matici

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix} \sim \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

Posloupností úprav jsme dostali generující matici kódu C_F ve standardním tvaru. Protože jsme použili pouze řádkové úpravy, dostali jsme generující matici stejného (nikoliv ekvivalentního) kódu C_F.

✓

Příklad 4.11. Mějme lineární ternární $[6, 3]$-kód nad tělesem $GF(3)$ daný generující maticí

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

Sestavíme odpovídající generující matici ve standardním tvaru

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

Prohozením pořadí řádků (prvního a třetího řádku) a následnou permutací sloupců dostaneme generující matici ekvivalentního kódu ve standardním tvaru. Protože jsme provedli permutaci sloupců, dostali jsme generující matici *ekvivalentního* kódu, nikoliv stejného kódu. ✓

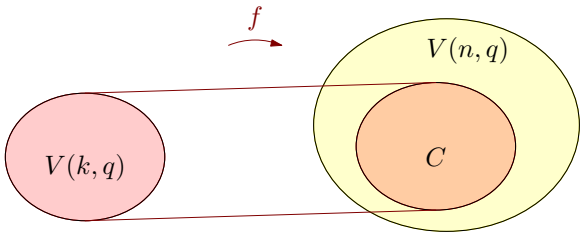
Cvičení

4.2.1. Dokažte nebo vyvraťte: každé dva lineární kódy, které jsou ekvivalentní jako kódy jsou také ekvivalentní jako lineární kódy.

Kapitola 5. Kódování a dekódování pomocí lineárních kódů

5.1. Kódování pomocí lineárního kódu

Výhody lineárních kódů se projeví v plné míře při kódování a dekódování. Mějme libovolný lineární $[n, k]$-kód C nad tělesem $GF(q)$. Označme G jeho generující matici. Kódová slova kódu C tvoří podprostor dimenze k v prostoru $V(n, q)$. Takových kódových slov je právě q^k. Kód C tak může sloužit k zaslání, resp. uložení, právě q^k různých zpráv. Tyto zprávy (nikoliv kódová slova) si můžeme představit jako q^k prvků (vektorů) vektorového prostoru $V(k, q)$. Všimněte si, že kódová slova patří do jiného prostoru, a sice do $V(n, q)$.



Obrázek 5.1.: *Kódování je bijekce* $f : V(k, q) \to C$.

Každé zprávě chceme jednoznačně přiřadit kódové slovo (Obrázek 5.1.). Slov v prostoru $V(k, q)$ i kódových slov v podprostoru C prostoru $V(n, q)$ je stejný počet, je jich právě q^k. Takové přiřazení má být jednoznačné, může být využita libovolná bijekce $f : V(k, q) \to C$.

Při šifrování se snažíme, aby vazby nebyly jednoduše zjistitelné nebo dohledatelné, avšak při kódování bychom naopak uvítali takové přiřazení, které provedeme snadno a rychle. Navíc preferujeme taková přiřazení, kdy vzory a obrazy umíme systematicky přiřadit oběma směry: jak při kódování přiřadit zprávám z prostoru $V(k, q)$ kódová slova v prostoru $V(n, q)$, tak následném dekódování s co nejmenší námahou a ideálně s možností detekce a opravy chyb přijatým slovům (ne nutně kódovým) z prostoru $V(n, q)$ přiřadit původní zprávy z prostoru $V(k, q)$.

Proces kódování

Existuje velmi jednoduchý a elegantní proces kódování lineárních kódů. Předpokládejme, že na vstupu máme zprávu $\vec{u}$ jako slovo $\vec{u} = u_1 u_2 \dots u_k$ z prostoru $V(k, q)$. Zprávu stačí jako řádkový vektor zprava vynásobit generující maticí G kódu C.

$$\vec{u}G = \vec{x} \tag{7}$$

Výsledkem je řádkový vektor z prostoru $V(n, q)$. Vskutku, pokud $\vec{r}_1, \vec{r}_2, \dots, \vec{r}_k$ jsou řádky generující matice G, tak

$$\vec{u}G = \sum_{i=1}^{k} u_i \vec{r}_i = \vec{x}$$

je lineární kombinace kódových slov báze, přičemž kódová slova báze jsou právě řádky generující matice. Tato lineární kombinace je kódovým slovem $\vec{x}$, které je přiřazeno vektoru $\vec{u}$ z prostoru C (podprostoru $V(n, q)$). Rovnice (7) popisuje bijekci, která každému vektoru k-rozměrného prostoru $V(k, q)$ jednoznačně přiřadí vektor k-rozměrného podprostoru C v n-rozměrném prostoru $V(n, q)$.

Příklad 5.1. Mějme kód C_3 z Příkladu 1.11. Podle Příkladu 4.5. víme, že jeho generující matice je například

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

Ukážeme příklad kódování pomocí generující matice kódu C_3.

Kód C_3 obsahuje jen čtyři binární kódová slova 00000, 01110, 10101 a 11011, která tvoří vektorový podprostor C_3 dimenze 2 ve vektorovém prostoru $V(5,2)$. Kód C_3 tak může sloužit k posílání čtyř různých zpráv. Tyto čtyři zprávy můžeme popsat jako (všechny) čtyři vektory binárního vektorového prostoru $V(2,2)$. Tyto vektory jsou 00, 01, 10 a 11.

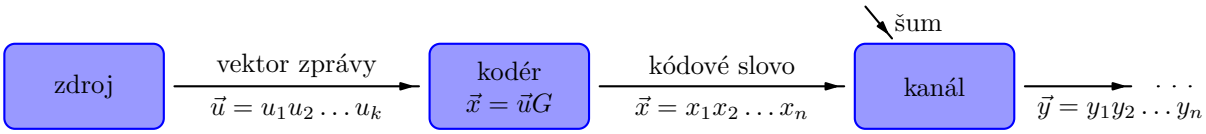
Abychom zprávy zakódovali, stačí příslušný vektor vynásobit jako řádkový vektor zprava generující maticí. Například zprávu 11 zakódujeme

$$[1,1]\cdot\begin{bmatrix}0&1&1&1&0\\1&0&1&0&1\end{bmatrix}=[1,1,0,1,1]=11011$$

Zprávě 11 je tak přiřazeno kódové slovo 11011. Ostatní tři zprávy zakódujeme následovně:

$$[0,0]\cdot\begin{bmatrix}0&1&1&1&0\\1&0&1&0&1\end{bmatrix}=[0,0,0,0,0]=00000,$$
$$[0,1]\cdot\begin{bmatrix}0&1&1&1&0\\1&0&1&0&1\end{bmatrix}=[1,0,1,0,1]=10101,$$
$$[1,0]\cdot\begin{bmatrix}0&1&1&1&0\\1&0&1&0&1\end{bmatrix}=[0,1,1,1,0]=01110.$$

Všimněte si, že každé zprávě je přiřazena některá lineární kombinace bázových vektorů vektorového podprostoru C_3. ✓



Obrázek 5.2.: *Kódování lineárním kódem.*

Kódování pomocí matice ve standardním tvaru

Pokud je generující matice G kódu C ve standardním tvaru, je proces kódování ještě jednodušší. Generující matice je tvaru $G=[\,I_k|A\,]$, kde matice A je tvaru $k\times(n-k)$. Máme-li na vstupu vektor $\vec{u}=u_1u_2\ldots u_k$, tak po vynásobení generující maticí ve standardním tvaru dostaneme vektor $\vec{x}=\vec{u}G=x_1x_2\ldots x_n$. Vzhledem k tvaru generující matice bude prvních k znaků identických se znaky vstupního vektoru, tj. $x_i=u_i$ pro $i=$ $=1,2,\ldots k$. Jedná se právě o *znaky zprávy*. Zbývajících $(n-k)$ znaků vyjádříme dle součinu $\vec{u}G$ jako

$$x_{i+k}=\sum_{j=1}^{k}u_ja_{ji}$$

pro $i=1,2\ldots n-k$. Tyto zbývající znaky tvoří se nazývají *kontrolní znaky*. Představují redundanci, kterou jsme do kódu přidali, aby bylo možno detekovat a případně opravovat chyby.

Příklad 5.2. Mějme opět kód C_3 z Příkladu 1.11. Vezmeme si jeho generující matici ve standardním tvaru. Stačí prohodit oba řádky a dostaneme

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

Ukážeme příklad kódování pomocí generující matice ve standardním tvaru.
Abychom zakódovali vektory vstupní zprávy, stačí opět příslušný vektor vynásobit zprava generující maticí $G=[\,I_2|A\,]$, kde

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

$$[0,0]\cdot\begin{bmatrix}1&0&1&0&1\\0&1&1&1&0\end{bmatrix}=[0,0,0,0,0]=00000,$$
$$[0,1]\cdot\begin{bmatrix}1&0&1&0&1\\0&1&1&1&0\end{bmatrix}=[0,1,1,1,0]=01110,$$
$$[1,0]\cdot\begin{bmatrix}1&0&1&0&1\\0&1&1&1&0\end{bmatrix}=[1,0,1,0,1]=10101,$$
$$[1,1]\cdot\begin{bmatrix}1&0&1&0&1\\0&1&1&1&0\end{bmatrix}=[1,1,0,1,1]=11011.$$

Všimněte si, že každé zprávě 00, 01, 10 a 11 je přiřazeno kódové slovo tak, že za bity zprávy *přidáme kontrolní bity*, které dostaneme součinem vektoru zprávy a submatice A. ✓

V další sekci rozebereme následující příklad kódu podrobně. Ukážeme nejen proces kódování, ale i proces dekódování včetně detekce a opravy chyb.

Příklad 5.3. Mějme binární lineární $[4,2]$-kód C_6 daný generující maticí

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

Určíme vlastnosti kódu C_6 a ukážeme proces kódování pomocí kódu C_6.
Kód C_6 obsahuje čtyři kódová slova 0000, 1010, 0111, 1101. Podle Věty 4.2. ihned vidíme, že tento kód má minimální vzdálenost 2, proto je kód C_6 schopen podle Důsledku 1.3. detekovat jednu chybu a obecně nemůže opravit každou chybu (některé chyby možná ano).

Abychom zakódovali vektory vstupní zprávy, stačí opět příslušný vektor vynásobit generující maticí $G=[\,I_2|A\,]$, kde

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

Dostaneme

$$[0,0]\cdot\begin{bmatrix}1&0&1&0\\0&1&1&1\end{bmatrix}=[0,0,0,0]=0000,$$
$$[0,1]\cdot\begin{bmatrix}1&0&1&0\\0&1&1&1\end{bmatrix}=[0,1,1,1]=0111,$$
$$[1,0]\cdot\begin{bmatrix}1&0&1&0\\0&1&1&1\end{bmatrix}=[1,0,1,0]=1010,$$
$$[1,1]\cdot\begin{bmatrix}1&0&1&0\\0&1&1&1\end{bmatrix}=[1,1,0,1]=1101.$$

Všimněte si, každé kódové slovo vniklo ze slova zprávy přidáním kontrolních bitů, které dostaneme součinem vektoru zprávy a submatice A. ✓

Implementace procesu kódování

Některé hardwarové prvky, které se používají pro přenos dat, zpravidla nemají příliš velkou výpočetní sílu. Proces kódování není implementován jako maticový součin, ale pomocí soustav hradel se z výchozí zprávy generují přímo kódová slova. Hlavní myšlenku si ukážeme na následujícím příkladu.

Příklad 5.4. Mějme binární lineární $[4,2]$-kód C_6 z Příkladu 5.3. Ukážeme, jak zakódovat slovo $\vec{u}=u_1u_2$ pomocí operace sčítaní.
Abychom zakódovali vektor $\vec{u}=u_1u_2$ vstupní zprávy, vynásobíme slovo $\vec{u}$ generující maticí $G=[\,I_2|A\,]$, kde

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

Dostaneme

$$[\,u_1,u_2\,]\cdot\begin{bmatrix}1&0&1&0\\0&1&1&1\end{bmatrix}=[\,u_1,u_2,u_1+u_2,u_2\,].$$

To znamená, že za dva bity zprávy u_1u_2 přidáme další dva bity u_1+u_2 a u_2. Pro nalezení kódového slova stačí kopírovat bity vstupní zprávy a jedna operace sčítání. ✓

Příklad 5.5. Podle Příkladu 4.10. víme, že generující matice ve standardním tvaru binárního lineárního $[4,2]$-kódu C_F z Příkladu 3.5. je

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

Kód C_F má 16 kódových slov. Ukážeme, jak zakódovat slova 0011, 1110, 1101 a 1111 pomocí kódu C_F. Každé slovo 0011, 1110, 1101, 1111 vynásobíme generující G a dostaneme

$$[0,0,1,1] \cdot \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} = [0,0,1,1,1,0,1] = 0011101,$$

$$[1,1,1,0] \cdot \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} = [1,1,1,0,1,0,0] = 1110100,$$

$$[1,1,0,1] \cdot \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} = [1,1,0,1,0,0,1] = 1101001,$$

$$[1,1,1,1] \cdot \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} = [1,1,1,1,1,1,1] = 1111111.$$

Všimněte si, každé kódové slovo vniklo ze slova zprávy přidáním tří kontrolních bitů, které dostaneme součinem vektoru zprávy a submatice A, kde

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

✓

Poznámka 5.1. Lineární transformace zachovává lineární kombinaci
Všimněte, že v předchozím Příkladu 5.5. pro slova 0011 a 1110 platí $0011 + 1110 = 1101$. I pro odpovídající kódová slova platí $0011101 + 1110100 = 1101001$, kde výsledné slovo 1101001 je kódovým slovem, které odpovídá slovu 1101. To není náhoda, neboť kód C_F je současně vektorovým (pod)prostorem prostoru $V(7,2)$ a násobení generující maticí odpovídá lineární transformaci vektorového prostoru $V(4,2)$. Z lineární algebry víme, že transformace vektorového prostoru zachovává lineární kombinaci vektorů.

Cvičení

5.1.1. Navážeme na Příklad 5.5. Určete kódová slova přiřazená každému kódovému slovu 0000, 0001, …, 1111.

5.1.2. Navážeme na Příklad 5.5. Ukažte, jak zakódovat vstupní slovo $\vec{u} = u_1u_2u_3u_4$.

5.2. Dekódování pomocí lineárního kódu

Mějme kód C daný generující maticí G. Předpokládejme, že zpráva $\vec{u}$ byla zakódována jako $\vec{x} = \vec{u}G$. Při přenosu zašuměným kanálem došlo ke zkreslení a místo kódového slova $\vec{x}$ bylo přijato slovo $\vec{y}$. Označíme *chybový vektor* $\vec{e}$, kde

$$\vec{e} = \vec{y} - \vec{x} = e_1e_2 \dots e_n.$$

Nyní stojíme před úkolem zjistit, které kódové slovo $\vec{x}$ bylo vysláno. Alternativně můžeme určit, jaký je chybový vektor $\vec{e}$.

V roce 1960 navrhl Slepian jednoduché dekódovací schéma, které umožňuje najít nejbližší kódové slovo $\vec{x}$ k přijatému slovu $\vec{y}$. Za předpokladu, že při přenosu nastalo jen málo chyb, bude přijaté slovo $\vec{y}$ správně dekódováno jako $\vec{x}$.

Než popíšeme Slepianovo dekódovací schéma, nejprve připomeneme některé pojmy a některá známá tvrzení z algebry.

Definice Koset kódu
Mějme lineární $[n,k]$-kód C nad tělesem $GF(q)$. Mějme libovolný vektor $\vec{a} \in V(n,q)$. Množina $\vec{a}+C$ definovaná předpisem

$$\vec{a}+C = \{\vec{a}+\vec{x} : \vec{x} \in C\}$$

se nazývá *koset* kódu C.

Koset kódu odpovídá třídě rozkladu grupy podle podgrupy, který byl zaveden v předmětu algebra. Podle definice vektorového prostoru $V(n,q)$ víme, že dvojice $(V(n,q),+)$ je grupou a $(V(k,q),+)$ je její normální podgrupou. Proto rozklad $V(n,q)/V(k,q)$ existuje a jeho prvky (třídy rozkladu) jsou právě výše uvedené kosety.

V dalším využijeme následující tvrzení. Jejich důkaz uvedeme, třebaže je přímou analogií tvrzení z teorie grup. Důkaz však budeme formulovat v jazyce teorie kódování.

Lemma 5.1. *Mějme koset $\vec{a}+C$ kódu C a mějme libovolný vektor $\vec{b} \in \vec{a}+C$. Potom platí $\vec{b}+C = \vec{a}+C$.*

Důkaz. Rovnost dvou kosetů ukážeme tak, že postupně ukážeme dvě množinové inkluze. Mějme libovolný vektor $\vec{b} \in \vec{a}+C$. Protože $\vec{b} \in \vec{a}+C$, tak existuje takový vektor $\vec{x} \in C$, že $\vec{b} = \vec{a}+\vec{x}$. Nyní podle definice kosetu $\vec{b}+C$ je $\vec{b}+\vec{y} \in \vec{b}+C$, pro nějaký vektor $\vec{y} \in C$. Můžeme psát $\vec{b}+\vec{y} = (\vec{a}+\vec{x})+\vec{y} = \vec{a}+(\vec{x}+\vec{y}) \in \vec{a}+C$, neboť součet dvou vektorů $\vec{x}+\vec{y}$ z C je opět vektorem v C. Tím jsme ukázali, že $\vec{b}+C \subseteq \vec{a}+C$.

Na druhou stranu mějme libovolný vektor $\vec{a}+\vec{y} \in \vec{a}+C$, kde $\vec{y} \in C$. Můžeme psát $\vec{a}+\vec{y} = (\vec{b}-\vec{x})+$ $+\vec{y} = \vec{b}+(\vec{y}-\vec{x}) \in \vec{b}+C$, neboť rozdíl dvou vektorů $\vec{y}-\vec{z}$ z C je opět vektorem v C. Tím jsme ukázali, že $\vec{a}+C \subseteq \vec{b}+C$, a celkem dostáváme $\vec{a}+C = \vec{b}+C$. □

Věta 5.2. *Mějme lineární $[n,k]$-kód C nad tělesem $GF(q)$. Potom platí*

(i) každý vektor prostoru $V(n,q)$ patří do nějakého kosetu kódu C,
(ii) každý koset kódu C obsahuje právě q^k vektorů,
(iii) dva kosety $\vec{a}+C$, $\vec{b}+C$ kódu C jsou buď disjunktní, nebo totožné.

Důkaz.
i) Vektor $\vec{0}$ patří do kódu C, a protože $\vec{a} = \vec{a}+\vec{0}$, tak $\vec{a}$ patří do kosetu $\vec{a}+C$.
ii) Definujeme zobrazení $f : C \to \vec{a}+C$ předpisem $f(\vec{x}) = \vec{a}+\vec{x}$ pro každé $\vec{x} \in C$. Zobrazení f je bijekce, neboť operace sčítaní vektorů tvoří grupu $(C,+)$. Pro $\vec{a}+\vec{x} = \vec{a}+\vec{y}$ dostaneme krácením $\vec{a} = \vec{b}$ a zobrazení f je injektivní. Vzorem prvku $\vec{y} \in \vec{a}+C$ je prvek $\vec{y}-\vec{a}$ a zobrazení f je surjektivní. To znamená, že $|\vec{a}+C| = |C| = q^k$.
iii) Předpokládejme, že dva kosety $\vec{a}+C$, $\vec{b}+C$ nejsou disjunktní, tedy, že $\vec{u} \in (\vec{a}+C) \cap (\vec{b}+C)$. Z definice kosetů pak existují slova $\vec{x}$, $\vec{y}$ taková, že $\vec{u} = \vec{a}+\vec{x} = \vec{b}+\vec{y}$. Vyjádříme $\vec{b} = \vec{a}+(\vec{x}-\vec{y})$, a tedy Podle Lemmatu 5.1. $\vec{b}+C = \vec{a}+C$. □

Věta 5.2. je speciální případ věty z teorie grup o třídách rozkladu grupy podle normální podgrupy.

Nyní navážeme na Příklad 5.3.

Příklad 5.6. Mějme $[4,2]$-kód C_6 z Příkladu 5.3. Sestavíme jednotlivé kosety kódu C_6.
Protože kód C_6 obsahuje kódová slova 0000, 1010, 0111 a 1101 a protože $\vec{0} \in C_6$, tak koset $0000+C =$ $= \{0000, 1010, 0111, 1101\}$. Ostatní kosety dostaneme přičtením dalších prvků prostoru $V(4,2)$. Například koset $0001+C = \{0001, 1011, 0110, 1100\}$. Všechny kosety pro přehlednost vypíšeme pod sebe.

$$\begin{aligned}
0000+C &= \{0000, 1010, 0111, 1101\}\\
1000+C &= \{1000, 0010, 1111, 0101\}\\
0100+C &= \{0100, 1110, 0011, 1001\}\\
0001+C &= \{0001, 1011, 0110, 1100\}
\end{aligned}$$

Všimněte si, že další kosety už budou shodné s některým z uvedených kosetů. Například koset $0010 + C$ je stejný jako koset $1000 + C$, neboť $0010 + C = \{0010, 1000, 0101, 1111\}$. Navíc podle Věty 5.2. víme, že $0010 + C = 1000 + C$, neboť $0010 \in 1000 + C$. ✓

Není náhoda, že jsme pro popis kosetů zvolili prvek s nejmenší váhou. Důvod vysvětlíme v dalším textu. Nejprve vyslovíme následující definici.

Reprezentant kosetu

Definice Mějme $[n, k]$-kód C a jeho kosety $\vec{x} + C$. Vektor $\vec{a}$ nazveme *reprezentant* kosetu $\vec{x} + C$, jestliže mezi všemi vektory kosetu $\vec{x}+C$ má vektor $\vec{a}$ nejmenší váhu. Pokud existuje více vrcholů se stejnou nejmenší váhou, zvolíme za reprezentanta libovolný z nich.

Příklad 5.7. Navážeme na Příklad 5.6. Určíme reprezentanty všech kosetů.
Reprezentantem kosetu $\vec{0} + C$ je právě vektor $\vec{0}$. Tento reprezentant je určen jednoznačně. Reprezentantem kosetu $1000+C$ je vektor 1000, mohli bychom však reprezentantem zvolit i vektor 0010. Reprezentantem kosetu $0100+C$ je vektor 0100 a reprezentantem kosetu $0001+C$ je vektor 0001. Další kosety v prostoru $V(4, 2)$ nenajdeme. Například koset $1101 + C = 0000 + C$ a jeho reprezentantem je 0000. Dále vektor 0010 nebyl zvolen reprezentantem žádného kosetu, neboť $0010 + C = 1000 + C$ a reprezentantem je 1000. ✓

Celý prostor $V(n, q)$ tak můžeme podle Věty 5.2. rozložit na kosety stejné velikosti q^k. Rozklad je

$$\vec{0} + C, \vec{a}_1 + C, \vec{a}_2 + C, \ldots, \vec{a}_s + C.$$

Protože podprostor C je dimenze k a má právě q^k prvků, tak celý prostor $V(n, q)$ má právě $q^n/q^k = q^{n-k}$ kosetů. Počet reprezentantů je q^{n-k}, a protože vektor 0 je vždy reprezentantem kosetu $\vec{0}+C$, tak $s = q^{n-k}-1$.

Standardní dekódování
Pro nepříliš velké kódy tak můžeme dekódovat podle následujícího postupu. Máme $[n, k]$-kód C jako podprostor dimenze k prostoru $V(n, k)$. Sestavíme všechny kosety a v každém zvolíme reprezentanta s nejmenší váhou $\vec{0}, \vec{a}_1, \vec{a}_2, \ldots, \vec{a}_s$. Každé přijaté slovo $\vec{y}$, které se nachází v kosetu $a_i + C$, dekódujeme jako kódové slovo $\vec{x} = \vec{y} - \vec{a}_i$.

Postup můžeme pěkně znázornit tzv. *Slepianovým standardním rozmístěním*. Sestavíme tabulku všech slov prostoru $V(n, q)$, která má M sloupců, $M = |C| = q^k$ a $s + 1 = q^{n-k}$ řádků. V prvním řádku napíšeme všechna kódová slova kódu C. Reprezentant této třídy je vektor $\vec{0}$, který bude napsán v prvním sloupci. Podobně v každém dalším řádku vypíšeme prvky nějakého kosetu. Přitom v prvním sloupci bude reprezentant $\vec{a}_i$ kosetu a ostatní vektory kosetu $\vec{a}_i + C$ napíšeme v řádku v takovém pořadí, aby slovo $\vec{a}_i + \vec{x}$ se nacházelo ve stejném sloupci jako kódové slovo $\vec{w}$. Reprezentant $\vec{a}_i$ je *chybovým slovem*.

Celý postup si ilustrujeme na příkladu.

Příklad 5.8. Navážeme na Příklad 5.7. Sestavíme Slepianovo standardní rozmístění slov kódu C_6 a dekódujeme přijatá slova 1101, 1011, 1000 a 0010.
Dostaneme Tabulku 5.1.

	reprezentanti ↓			
kódová slova →	0000	1010	0111	1101
	1000	0010	1111	0101
	0100	1110	0011	1001
	0001	1011	0110	1100

Tabulka 5.1.: *Slepianovo standardní rozmístění.*

Všimněte si, že první řádek obsahuje všechna kódová slova a že každý další řádek vznikl z prvního řádku přičtením reprezentanta.

Přijaté slovo $\vec{y}_1 = 1101$ najdeme ve čtvrtém sloupci a prvním řádku, jedná se proto přímo o kódové slovo $\vec{x}_1 = \vec{y}_1$.

Přijaté slovo $\vec{y}_2 = 1011$ najdeme v druhém sloupci a čtvrtém řádku, v jehož horním záhlaví je kódové slovo 1010. Přijaté slovo $\vec{y}_2$ dekódujeme jako $\vec{x}_2 = 1010$. (Také bychom mohli od $\vec{y}_2$ odečíst reprezentanta čtvrtého řádku $\vec{x}_2 = \vec{y}_2 - \vec{e}_4 = 1011 - 0001 = 1010$.)

Přijaté slovo $\vec{y}_3 = 1000$ najdeme v prvním sloupci a druhém řádku, v jehož horním záhlaví je kódové slovo 0000. Přijaté slovo $\vec{y}_3$ dekódujeme jako 0000. (Také bychom mohli od $\vec{y}_3$ odečíst reprezentanta druhého řádku $\vec{x}_3 = \vec{y}_3 - \vec{e}_2 = 1000 - 1000 = 0000$.)

Přijaté slovo $\vec{y}_4 = 0010$ najdeme v druhém sloupci a druhém řádku, v jehož horním záhlaví je kódové slovo 1010. Přijaté slovo $\vec{y}_4$ dekódujeme jako 1010. (Také bychom mohli od $\vec{y}_4$ odečíst reprezentanta druhého řádku $\vec{x}_4 = \vec{y}_4 - \vec{e}_2 = 0010 - 1000 = 1010$.)

✓

Poznámka 5.2. Všimněte si, jak se projeví, že binární lineární $[4, 2]$-kód C_6 z Příkladu 5.3. má minimální vzdálenost $\mathrm{dist}(C_6) = 2$. Zpráva 10 bude zakódována jako 1010. Pokud při přenosu nastane jedna chyba, tak tato chyba jistě bude detekována, ale může a nemusí být opravena. V Příkladu 5.8. jsme ukázali, že pokud nastane chyba na čtvrté pozici, bude přijato kódové slovo $\vec{y}_2 = 1011$. Chyba bude rozpoznána a správně opravena, dekódované slovo $\vec{x}_2 = 1010$ odpovídá správnému kódovému slovu. Avšak pokud nastane chyba na třetí pozici, bude přijato kódové slovo $\vec{y}_3 = 1000$. Chyba bude rozpoznána, ale dekódované slovo $\vec{x}_3 = 0000$ neodpovídá správnému kódovému slovu. Navíc obě přijatá slova $\vec{y}_2 = 1011$ i $\vec{y}_3 = 1000$ mají správně první dva bity, které odpovídají původní zprávě. Chyba nastala pouze u kontrolních bitů a jednou bylo chyba opravena správně a podruhé nikoliv. Jednou bylo dekódováno slovo 10 správně a podruhé špatně jako 00.

Teprve pokud bychom pracovali s kódem s nejmenší vzdáleností 3, tak by jedna chyba mohla být vždy opravena správně.

Poznámka 5.3. Protože reprezentant každého kosetu je chybový vektor s nejmenší váhou, tak algoritmus dekódování popsaný v Příkladu 5.8. dekóduje přijatá slova jako nejbližší kódová slova.

Pozorování z Příkladu 5.8. můžeme shrnout do následujícího obecného postupu při dekódování lineárního kódu. Přijmeme slovo $\vec{y}$. Najdeme slovo $\vec{y}$ v tabulce Slepianova standardního rozmístění. Za předpokladu malého počtu chyb a symetrického q-árního kanálu je hledané vyslané slovo $\vec{x}$ v prvním řádku tabulky ve stejném sloupci jako přijaté slovo $\vec{y}$. Pokud nenastala žádná chyba, je přijaté slovo $\vec{y} = \vec{x}$. Pokud nějaké chyby nastaly, tak nejpravděpodobnější chyba je právě reprezentant $\vec{a}_i$ kosetu $\vec{a}_i + C = \vec{y} + C$.

Při dekódování můžeme buď přečíst vyslané kódové slovo $\vec{x}$ v horním záhlaví tabulky, nebo určit chybový vektor $\vec{e} = \vec{a}_i$ v levém záhlaví tabulky, kde je uveden reprezentant kosetu a vypočítat $\vec{x} = \vec{y} - \vec{e}$. V binárních kódech samozřejmě platí také $\vec{x} = \vec{y} + \vec{e}$.

Postup můžeme přehledně zapsat do následujícího algoritmu.

Algoritmus 5.3. Dekódování lineárního kódu C
Vysláno bylo kódové slovo $\vec{x}$ a přijato bylo slovo $\vec{y}$.

1) *Najdeme slovo $\vec{y}$ v tabulce Slepianova standardního rozmístění.*
2) *Slovo $\vec{y}$ dekódujeme jako slovo $\vec{x}$ v prvním řádku tabulky ve stejném sloupci, ve kterém se nachází slovo $\vec{y}$.*

Sestavení standardního rozmístění

V Příkladu 5.8. jsme s výhodou využili tabulku Slepianova standardního rozmístění slov. Pro sestavení tabulky můžeme použít následující algoritmus.

Algoritmus 5.4. Sestavení Slepianova standardního rozmístění
Mějme lineární kód C.

1) *Do prvního řádku tabulky vypíšeme všechna kódová slova kódu C, první bude kódové slovo $\vec{0}$.*
2) *Najdeme slovo $\vec{a}_1$ minimální váhy, které se v prvním řádku nevyskytuje. Do tabulky vypíšeme všechna slova kosetu $\vec{a}_1 + C$ tak, že v prvním sloupci je reprezentant $\vec{a}_1$ a pod kódové slovo $\vec{x}$ napíšeme slovo $\vec{a}_1 + \vec{x}$ pro každé $\vec{x} \in C$.*

3) Postup opakujeme: Najdeme slovo $\vec{a}_i$ minimální váhy, které se v prvních i řádcích nevyskytuje. Do tabulky vypíšeme všechna slova kosetu $\vec{a}_i + C$ tak, že v prvním sloupci je reprezentant $\vec{a}_i$ a pod kódové slovo $\vec{x}$ napíšeme slovo $\vec{a}_i + \vec{x}$ pro každé $\vec{x} \in C$.
4) Končíme, jakmile je v tabulce každé slovo kódu C právě jedenkrát (podle Věty 5.2.).

Příklad 5.9. Mějme kód C_3 z Příkladu 1.11. s generující maticí

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

Sestavíme Slepianovo standardní rozmístění kódu C_3.
Kód C_3 obsahuje jen čtyři binární kódová slova 00000, 01110, 10101 a 11011, která tvoří vektorový podprostor C_3 dimenze 2 ve vektorovém prostoru $V(5,2)$. Protože lineární $[5,2]$-kód C_3 má čtyři kódová slova a celý prostor $V(5,2)$ má $2^5 = 32$ slov, tak Slepianovo standardní rozmístění bude tabulka se 4 sloupci a s $32/4 = 8$ řádky. ✓

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

Tabulka 5.2.: *Slepianovo standardní rozmístění kódu C_3.*

Poznámka 5.4. Pro rozsáhlé kódy je dekódování pomocí Slepianova standardního rozmístění náročné na paměť a zejména na vyhledávání v rozsáhlé tabulce. Efektivnější způsob dekódování ukážeme v další kapitole.

Odkazy:

- https://en.wikipedia.org/wiki/Standard_array
- https://en.wikipedia.org/wiki/David_Slepian

Cvičení

5.2.1. Ukažte, že neexistuje binární lineární $[4,2,3]$-kód.

5.2.2. Sestavte Slepianovo standardní rozmístění

a) lineárního $[2,2]$-kódu C_1 daného generující maticí $G_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$,
b) lineárního $[3,2]$-kódu C_2 daného generující maticí $G_2 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$,
c) lineárního $[5,2]$-kódu C daného generující maticí $G_3 = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{bmatrix}$.

5.2.3.$^\heartsuit$ Pomocí Slepianova standardního rozmístění kódu C ze Cvičení 5.2.2. dekódujete přijatá slova a) $y_1 = 11111$ b) $y_2 = 01011$.

5.2.4. Určete minimální vzdálenost kódů C_1, C_2 a C z Příkladu 5.2.2. Určete, kolik chyb je každý kód schopen detekovat a opravit.

5.2.5. Uveďte příklad, kdy dojde při přenosu ke dvěma chybám a přijaté slovo je dekódováno a) správně b) špatně.

5.3. Pravděpodobnost opravy chyb

V předchozí kapitole v Poznámce 5.2. jsme ukázali, že lineární kód popsaný v Příkladech 5.3. až 5.8. v jednom konkrétním případě nadělal více škody než užitku. Zatímco bity zprávy byly přeneseny správně, tak chyba při přenosu kontrolních bitů způsobila, že kódové slovo bylo dekódováno chybně. Abychom mohli uvedenou situaci rozebrat a uměli „měřit" kvalitu daného kódu, musíme umět určit pravděpodobnost, že přijaté slovo $\vec{y}$ bude dekódováno správně jako vyslané slovo $\vec{x}$. U obecného kódu je těžké takovou pravděpodobnost určit. Avšak pro lineární kódy dekódované pomocí Slepianova standardního rozmístění to není obtížné, spíše jen pracné. Navíc u malých kódů nebo u perfektních kódů se ukazuje, že vyčíslení pravděpodobností je snadné.

Dále v této kapitole se omezíme na binární kódy délky n zasílané symetrickým (binárním) kanálem, předpokládáme symetrický binární (q-ární) kanál s pravděpodobností chyby p, jak byl zaveden v Kapitole 1.3. Potom pravděpodobnost, že chybový vektor $\vec{e} = \vec{y} - \vec{x}$ má váhu i je $p^i(1-p)^{n-i}$.

Správně dekódovaná slova

Všechna vyslaná slova jsou kódová slova z horního záhlaví Slepianova standardního rozmístění. Pokud nastanou právě ty chyby, které odpovídají chybovému vektoru z levého záhlaví, tak přijaté slovo bude dekódováno správně. Můžeme vyslovit následující tvrzení.

Věta 5.5. *Mějme binární $[n,k]$-kód C. Pro $i = 1, 2, \ldots, n$ označíme α_i počet reprezentantů kosetu, jejichž váha je i. Pravděpodobnost, že zaslaný vektor $\vec{x}$ bude užitím Slepianova standardního rozmístění správně dekódován, označme $P_{spr}(C)$. Platí*

$$P_{spr}(C) = \sum_{i=0}^{n} \alpha_i p^i (1-p)^{n-i}. \tag{8}$$

Důkaz je ponechán jako Cvičení 5.3.1.

Příklad 5.10. Opět navážeme na Příklad 5.8., ve kterém jsme sestavili Slepianovo standardní rozmístění kódu C_6. Určíme pravděpodobnost $P_{spr}(C_6)$, tj. pravděpodobnost, že přijaté slovo bude dekódováno správně. Vyčíslíme pravděpodobnost $P_{spr}(C_6)$ pro pravděpodobnost chyby přenosu $p = 0.1$ a pro pravděpodobnost chyby $p = 0.01$.

Kód C_6 je binární lineární $[4,2]$-kód. Reprezentanti kosetů z Příkladu 5.8. jsou 0000, 1000, 0100 a 0001. Proto platí $\alpha_0 = 1$, $\alpha_1 = 3$ a $\alpha_2 = \alpha_3 = \alpha_4 = 0$. Nyní podle Věty 5.5. vyčíslíme

$$P_{spr}(C_6) = \sum_{i=0}^{n} \alpha_i p^i (1-p)^{n-i} = 1(1-p)^4 + 3p(1-p)^3 + 0 = (1-p)^3(1+2p).$$

Pravděpodobnost $P_{spr}(C_6)$ je funkce závislá na hodnotě pravděpodobnosti chyby p.

Pro chybu přenosu $p = 0.1$ je $P_{spr}(C_6) = 0.8748$. To znamená, že více než 87 % zaslaných slov bude správně dekódováno. Pro $p = 0.01$ vychází dokonce $P_{spr}(C_6) = 0.9897$. To znamená, že téměř 99 % zaslaných slov bude správně dekódováno. ✓

Chybně dekódovaná slova

Míra chybných slov je pravděpodobnost, že vyslané slovo $\vec{x}$ bude přijato jako slovo $\vec{y}$ a bude dekódováno chybně, tj. nebude dekódováno jako slovo $\vec{x}$. Míru chybných slov budeme značit $P_{chyb}(C)$, kde C je daný kód. Ihned je zřejmé, že $P_{chyb}(C) = 1 - P_{spr}(C)$.

Příklad 5.11. Navážeme na Příklad 5.10. Určíme míru chybných slov při přenosu binárním symetrickým kanálem s využitím kódu C_6.

Pro porovnání vyčíslíme míru chybných slov pro pravděpodobnost chyby $p = 0.1$ a pro pravděpodobnost chyby $p = 0.01$. Porovnáme míru chybných slov s chybou v případně, že pro přenos nepoužili samoopravný kód.

Míra chybných slov při přenosu binárním symetrickým kanálem s využitím kódu C_6 dle Slepianova standardního rozmístění z Příkladu 5.8. je $p = 0.1$ dána $P_{chyb}(C_6) = 1 - P_{spr}(C_6) = 1 - (1-p)^3(1+2p)$.

Po dosazení $p = 0.01$ dostáváme $P_{chyb}(C_6) = 1 - P_{spr}(C_6) = 1 - 0.8748 = 0.1252$. To znamená, že 12 až 13 slov ze 100 poslaných bude přijato s chybou. Pro $p = 0.01$ vychází $P_{chyb}(C_6) = 1 - P_{spr}(C_6) = = 1 - 0.9897 = 0.0102$. To znamená, že přibližně zhruba jen každé sté poslané slovo bude přijato s chybou.

Kdybychom nepoužili samoopravný kód C_6, tak pro pravděpodobnost chyby, že dvoubitové slovo bude obdrženo bez chyby, je vyjádřeno $P_{spr}(C) = (1-p)^2$. Míra chybných slov pak je $P_{chyb}(C) = 1 - P_{spr}(C) = = 1-(1-p)^2 = p(2-p)$. Po dosazení $p = 0.1$ dostáváme $P_{chyb} = 0.19$ (19 ze sta odeslaných slov bude chybně

přijato) a pro $p = 0.01$ vychází $P_{chyb} = 0.0190$, tj. přibližně každé padesáté slovo bude přijato s chybou. Můžeme shrnout, že využitím samoopravného kódu C_6 jsme dosáhli přibližně poloviční míry chybných slov za cenu dvojnásobné délky zasílaných zpráv. ✓

Poznámka 5.5. Pozorování z této podkapitoly můžeme shrnout následujícím způsobem. Jestliže pro přenos využíváme lineární kód C s minimální vzdáleností $\mathrm{dist}(C) = 2t + 1$, případně $2t + 2$, tak kód C opraví libovolných t chyb. To znamená, že každý vektor s váhou nejvýše t může být reprezentantem kosetu, a potom můžeme vyčíslit $\alpha_i = \binom{n}{i}$ pro $i = 0, 1, \ldots t$. Avšak pro $i > t$ je náročné (respektive pracné) vyčíslit hodnotu α_i. Hodnota může záviset na volbě konkrétní kódu, avšak nebude záviset na konkrétní volbě sestaveného Slepianovova standardního rozmístění (proč?).

Existují však kódy, pro které je toto vyčíslení snadné, a sice perfektní kódy. Při využití perfektního kódu jsou všechny chybové vektory opravené kódem *právě* vektory s váhou nejvýše t. Pro perfektní lineární $[n, k, 2t + 1]$-kódy můžeme psát, že $\alpha_i = \binom{n}{i}$ pro $i = 0, 1, \ldots, t$, a $\alpha_i = 0$ pro $i > t$.

Příklad 5.12. Víme, že kód C_F z Příkladu 3.5. je perfektní. Určíme pravděpodobnost $P_{spr}(C_F)$ i pravděpodobnost P_{chyb}.
Tady chybí část textu, kterou je třeba doplnit. ✓

Cvičení

5.3.1. Dokažte Větu 5.5.

5.3.2. Dokažte zobecnění Věty 5.5. pro q-ární kódy.

5.3.3. Určete s jakou pravděpodobností při užití a) kódu C_1, b) kódu C_2 c) kódu C ze Cvičení 5.2.2. bude zaslané slovo dekódováno správně. Pravděpodobnost vyčíslete pro chybu $p = 0.01$.

5.3.4. Navážeme na Cvičení 5.3.3. Určete pravděpodobnost, že chyba přenosu nebude detekována a že vyslané slovo bude přijato chybně (určete P_{chyb}). Pro kódy a) C_1, b) C_2, c) C. Vyčíslete chybu P_{chyb} pro $p = 0.1$, pro $p = 0.01$ a výsledky porovnejte.

5.3.5. Je každý binární lineární $[7, 4, 3]$-kód perfektní?

5.3.6. Sestavte nějaký binární lineární $[7, 4, 2]$-kód.

5.3.7. Mějme binární lineární $[7, 4, 3]$-kód C daný generující maticí ve standardním tvaru

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

Určete pravděpodobnost, že přijaté slovo bude dekódováno správně. Pravděpodobnost vyčíslete pro $p = 0.1$ a $p = 0.01$.

5.3.8. Mějme perfektní binární lineární $[n, k, d]$-kód. Vyčíslete hodnoty α_i ve vztahu (8) z Věty 5.5.

Kapitola 6. Duální kód a syndromové dekódování

6.1. Duální kódy

V kapitole 4. jsme ukázali, že lineární kódy můžeme dobře popsat pomocí generující matice. Nyní ukážeme další (ekvivalentní) způsob, jak můžeme lineární kód jednoznačně popsat pomocí tzv. kontrolní matice a ukážeme, jak kontrolní matici můžeme využít při dekódování.

Skalární součin

Nejprve připomeneme dobře známý pojem z lineární algebry: skalární součin dvou vektorů.

Definice Skalární součin

Mějme vektorový prostor $V(n,q)$ nad tělesem $GF(q)$ a mějme dva libovolné vektory $\vec{u},\vec{v} \in V(n,q)$. *Skalární součin* vektorů $\vec{u} = u_1u_2\ldots u_n$ a $\vec{v} = v_1v_2\ldots v_n$ je skalár z $GF(q)$ definovaný předpisem $\vec{u}\cdot\vec{v} = u_1v_1 + u_2v_2 + \cdots + u_nv_n$.

Lineární kód je současně vektorovým (pod)prostorem, proto můžeme definici formulovat nejen pro vektory, ale i pro kódová slova lineárního kódu.

Příklad 6.1. Ukážeme několik jednoduchých příkladů skalárních součinů vektorů.

1) Mějme dva vektory $\vec{u} = 1010$ a $\vec{v} = 1011$ vektorového prostoru $V(4,2)$. Jejich skalární součin je $\vec{u}\cdot\vec{v} = 1010\cdot 1011 = 0$.
2) Mějme dva vektory $\vec{u} = 1010$ a $\vec{v} = 1011$ vektorového prostoru $V(4,3)$. Jejich skalární součin je $\vec{u}\cdot\vec{v} = 1010\cdot 1011 = 2$.
3) Mějme dva vektory $\vec{u} = 0121$ a $\vec{v} = 2021$ vektorového prostoru $V(4,3)$. Jejich skalární součin je $\vec{u}\cdot\vec{v} = 0121\cdot 2021 = 2$.
4) Součin vektorů $\vec{u} = 0121$ a $\vec{v} = 2021$ v prostoru $V(4,10)$ nemá smysl počítat, neboť $GF(10)$ není těleso.

Všimněte si, že výpočet provádíme nad číselným tělesem příslušného řádu. Je-li například skalární součin $2121\cdot 1021$ počítán nad tělesem $GF(3)$, tak $2\cdot 1+1\cdot 0+2\cdot 2+1\cdot 1 = 2+0+1+1 = 1$, zatímco nad tělesem $GF(5)$ stejný součin dává $2\cdot 1+1\cdot 0+2\cdot 2+1\cdot 1 = 2+0+4+1 = 2$ a nad tělesem $GF(11)$ dostaneme $2\cdot 1+1\cdot 0+2\cdot 2+1\cdot 1 = 2+0+4+1 = 7$.

Připomeneme ještě jeden pojem dobře známý z lineární algebry.

Definice Ortogonalita

Mějme vektorový prostor $V(n,q)$ nad tělesem $GF(q)$ a mějme dva libovolné vektory $\vec{u},\vec{v} \in V(n,q)$. Řekneme, že vektory $\vec{u}$ a $\vec{v}$ jsou *ortogonální*, jestliže jejich skalární součin $\vec{u}\cdot\vec{v} = 0$.

Protože i lineární kódy jsou vektorové (pod)prostory, tak má také smysl se ptát, zda jsou kódová slova ortogonální.

Příklad 6.2. Vektory $\vec{u} = 1010$ a $\vec{v} = 1011$ z Příkladu 6.1. jsou ortogonální ve vektorovém prostoru $V(4,2)$, neboť jejich skalární součin je $\vec{u}\cdot\vec{v} = 1010\cdot 1011 = 0$. Avšak $\vec{u} = 1010$ a $\vec{v} = 1011$ se stejnými souřadnicemi nejsou ortogonální ve vektorovém prostoru $V(4,3)$, neboť jejich skalární součin je $\vec{u}\cdot\vec{v} = 1010\cdot 1011 = 2$.

Pro skalární součin platí následující dobře známá tvrzení.

Lemma 6.1. *Mějme vektorový prostor $V(n,q)$ nad tělesem $GF(q)$. Mějme vektory $\vec{u},\vec{v},\vec{w} \in V(n,q)$ a mějme dva skaláry $k,l \in GF(q)$. Potom platí*

(i) $\vec{u}\cdot\vec{v} = \vec{v}\cdot\vec{u}$,
(ii) $(k\vec{u}+l\vec{v})\cdot\vec{w} = k(\vec{u}\cdot\vec{w}) + l(\vec{v}\cdot\vec{w})$.

Důkaz je ponechán jako Cvičení 6.1.1.

Duální kód

Podle definice lineárního kódu (strana 40) víme, že každý lineární kód je nějakým podprostorem vektorového prostoru $V(n,q)$ nad tělesem $GF(q)$. Prvky $\vec{x}$ kódu C, který je podprostorem vektorového prostoru $V(n,q)$

jsou současně vektory (vzhledem k prostoru $V(n,q)$ nebo podprostoru C) a současně kódová slova kódu C. V textu budeme prvkům $\vec{x}$ říkat „vektory“, jestliže budeme využívat jejich vlastnosti ve vektorovém prostoru, a „slova“ nebo „kódová slova“, jestliže budeme zkoumat jejich vlastnosti v kódu C. Jedná se však vždy o stejnou uspořádanou n-tici symbolů tělesa $GF(q)$.

Nyní ukážeme, že i ortogonální doplněk lineárního kódu je lineárním kódem.

Definice Duální kód
Mějme lineární $[n,k]$-kód C. *Duální kód* ke kódu C je definován jako množina všech vektorů, které jsou ortogonální ke každému vektoru kódu C. Duální kód ke kódu C značíme $C^\perp$.

Podle definice duální kódu víme, že $C^\perp$ je množina vektorů. Ve Větě 6.3. ukážeme, že definice duálního kódu je korektní, tj. vskutku se jedná o kód. Nejprve však vyslovíme a ukážeme následující pomocné tvrzení.

Lemma 6.2. *Mějme q-ární lineární $[n,k]$-kód C daný generující maticí G. Vektor $\vec{u}$ patří do duálního kódu $C^\perp$ právě tehdy, když je ortogonální ke každému řádkovému vektoru matice G.*

Symbolicky můžeme tvrzení lemmatu zapsat $\vec{u} \in C^\perp \Leftrightarrow \vec{u}G^T = \vec{0}$, kde G^T je transponovaná generující matice G.
Důkaz.
„$\Rightarrow$“ Mějme vektor $\vec{u} \in C^\perp$. Důkaz první implikace je snadný, neboť řádkové vektory matice G jsou bázové vektory prostoru C a každý vektor duálního kódu $C^\perp$ je ortogonální ke všem vektorům C, zejména k vektorům báze.
„$\Leftarrow$“ Označme řádky generující matice G jako $\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_k$. Podle předpokladu je každý vektor $\vec{u} \in C^\perp$ ortogonální ke všem vektorům kódu C, přičemž každé kódové slovo $\vec{x}$ kódu C můžeme podle Věty 3.14. jednoznačně vyjádřit jako lineární kombinaci vektorů báze, tj. jako lineární kombinaci řádkových vektorů generující matice.

$$\vec{x} = a_1\vec{r}_1 + a_2\vec{r}_2 + \cdots + a_k\vec{r}_k, \text{ kde } a_1, a_2, \ldots, a_k \in GF(q)$$

Potom pro vektor $\vec{u}$ platí

$$\vec{u} \cdot \vec{x} = \vec{u} \cdot (a_1\vec{r}_1 + a_2\vec{r}_2 + \cdots + a_k\vec{r}_k),$$

což s využitím Lemmatu 6.1. dává

$$\begin{aligned} &= a_1(\vec{u} \cdot \vec{r}_1) + a_2(\vec{u} \cdot \vec{r}_2) + \cdots + a_k(\vec{u} \cdot \vec{r}_k) \\ &= a_1 0 + a_2 0 + \cdots + a_k 0 = 0. \end{aligned}$$

To znamená, že libovolný vektor $\vec{u} \in C^\perp$ je ortogonální ke každému vektoru (kódovému slovu) kódu C. □

Nyní můžeme dokázat korektnost definice duálního kódu. Tvrzení následující věty dokonce specifikuje dimenzi duálního kódu.

Věta 6.3. Věta o duálním kódu
Mějme lineární $[n,k]$-kód C nad tělesem $GF(q)$. Potom duální kód $C^\perp$ kódu C je lineární $[n, n-k]$-kód.

Důkaz. Nejprve ukážeme, že $C^\perp$ je lineární kód. Mějme libovolné dva vektory (slova) $\vec{u}_1, \vec{u}_2 \in C^\perp$ a libovolné dva skaláry $a_1, a_2 \in GF(q)$. Potom s využitím Lemmatu 6.1. pro libovolné kódové slovo $\vec{x}$ kódu C platí

$$\begin{aligned} (a_1\vec{u}_1 + a_2\vec{u}_2) \cdot \vec{x} &= a_1(\vec{u}_1 \cdot \vec{x}) + a_2(\vec{u}_2 \cdot \vec{x}) \\ &= a_1 0 + a_2 0 = 0. \end{aligned}$$

To znamená, že vektor (slovo) $\vec{u} = a_1\vec{u}_1 + a_2\vec{u}_2$ patří také do duálního kódu $C^\perp$, proto je duální kód $C^\perp$ podle definice lineárního kódu a podle Věty 3.11. lineárním kódem.

Nyní ukážeme, že dimenze duálního kódu $C^\perp$ je $n-k$. Jestliže G, $G = [\, g_{ij} \,]$, je generující matice kódu C, tak podle Lemmatu 6.2. do duálního kódu $C^\perp$ patří právě vektory (slova) $\vec{v}$, $\vec{v} = v_1 v_2 \ldots v_n$, pro která platí

$$\sum_{j=1}^{n} v_j g_{ij} = 0, \text{ pro } i = 1, 2, \ldots, k. \tag{9}$$

Soustava rovnic (9) je soustava k lineárně nezávislých rovnic (rovnice jsou lineárně nezávislé, neboť řádky generující matice jsou lineárně nezávislé) o n neznámých, což znamená, že prostor řešení $C^\perp$ této soustavy

má dimenzi $n-k$. A protože právě každý vektor $\vec{v}$ duálního kódu soustavu rovnic splňuje, neboť $\vec{v}G^T = \vec{0}$, tak duální kód a prostor řešení soustavy rovnic jsou totožné a dimenze kódu $C^\perp$ je $n-k$.

Důkaz tohoto tvrzení zde uvedeme v jazyce kódů. Jistě platí, že pokud jsou dva kódy C_1 a C_2 ekvivalentní, tak jsou ekvivalentní i jejich duální kódy $C_1^\perp$ a $C_2^\perp$. Tvrzení stačí ukázat pro kód C s generující maticí ve standardním tvaru, tj. pro kód jehož generující matice je

$$G = \begin{bmatrix} 1 & 0 & \cdots & 0 & a_{11} & \cdots & a_{1,n-k} \\ 0 & 1 & \cdots & 0 & a_{21} & \cdots & a_{2,n-k} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & a_{k,1} & \cdots & a_{k,n-k} \end{bmatrix}.$$

Duální kód $C^\perp$ pak je množina vektorů $\vec{v} = v_1 v_2 \dots v_n$, které splňují rovnici

$$C^\perp = \left\{ (v_1, v_2, \dots, v_n) : v_i + \sum_{j=1}^{n-k} a_{ij} v_{k+j} = 0 \text{ pro } i = 1, 2, \dots, k \right\}. \tag{10}$$

Pro každou z q^{n-k} možností prvků $v_{k+1}, v_{k+2}, \dots, v_n$ existuje vhodnou volbou prvku v_i právě jeden vektor $\vec{v} = v_1 v_2 \dots v_n$, který splňuje rovnici $v_i + \sum_{j=1}^{n-k} a_{ij} v_{k+j} = 0$ v popisu množiny (10). Proto platí $\left|C^\perp\right| = q^{n-k}$ a dimenze prostoru $C^\perp$ je $n-k$. □

Poznámka 6.1. Věta o duálním kódu a Frobeniova věta
Studentům technických oborů neunikne, že tvrzení Věty 6.3. je analogií Frobeniovy věty. Vskutku, nejen samotné tvrzení, ale i metody důkazu jsou stejné, ať už formulujeme tvrzení o dimenzi soustavy a dimenzi prostoru řešení v jazyce lineární algebry nebo teorie kódování, neboť se jedná o stejné tvrzení. Na druhou stranu podstatným rozdílem obou tvrzení je, že zatímco v lineární algebře pracujeme s vektorovými prostory nad tělesem $\mathbb{R}$, tak v teorii kódování pracujeme nad konečnými tělesy řádu q.

Otázka: Platí pro kód C a jeho duální kód $C^\perp$, že $C \cap C^\perp = \emptyset$?

Otázka: Platí pro kód C a jeho duální kód $C^\perp$, že $C \cup C^\perp = V(n, q)$?

Příklad 6.3. Mějme kód C_2 z Příkladu 1.10.

$$\text{kód } C_2 = \begin{cases} 000 \\ 011 \\ 101 \\ 110 \end{cases}$$

Jeho duální kód $C_2^\perp$ je opakovací kód

$$\text{kód } C_2^\perp = \begin{cases} 000 \\ 111. \end{cases}$$

Příklad 6.4. Mějme kód C_7.

$$\text{kód } C_7 = \begin{cases} 0000 \\ 0011 \\ 1100 \\ 1111. \end{cases}$$

Určíme duální kód $C_7^\perp$.
Stačí si uvědomit, že dimenze kódu C_7 je 2, proto dimenze kódu $C_7^\perp$ je také $4-2=2$. To znamená, že kód C_7 obsahuje 4 vektory. Není těžké ověřit, všechny čtyři vektoru kódu C_7 jsou ortogonální navzájem, proto $C_7^\perp = C_7$. ✓

Pro každý lineární kód platí následující jednoduché pozorování.

Věta 6.4. *Pro každý lineární $[n,k]$-kód C platí $(C^\perp)^\perp = C$.*

Důkaz. Tvrzení má tvar množinové rovnosti. Ukážeme jednu inkluzi a porovnáme velikosti kódů.

Jistě platí $C \subseteq (C^\perp)^\perp$, neboť každý vektor $\vec{v} \in C$ je ortogonální ke všem vektorům v $C^\perp$. Současně platí, že dimenze kódu $(C^\perp)^\perp$ je $n-(n-k)=k$, což je právě dimenze kódu C, proto $(C^\perp)^\perp = C$. □

Kontrolní matice
Nyní ukážeme, že lineární kód lze jednoznačně popsat i pomocí jiné matice, než generující matice.

Definice Kontrolní matice
Mějme lineární $[n,k]$-kód C nad tělesem $GF(q)$. *Kontrolní matice* H kódu C je generující matice G duálního kódu $C^\perp$.

Podle definice generující matice G lineárního $[n,k]$-kódu C nad tělesem $GF(q)$ víme, že se jedná o matici tvaru $k\times n$. Podle definice kontrolní matice H víme, že kontrolní matice je tvaru $(n-k)\times n$. Protože skalární součin každého (zejména bázového) vektoru duálního kódu se všemi (zejména bázovými) vektory kódu C je nula, tak platí $GH^T=0$, kde 0 je čtvercová nulová matice řádu k. Nyní z Lemmatu 6.2. a Věty 6.4. ihned vidíme, že pokud H je kontrolní matice lineárního kódu C, tak platí

$$C=\left\{\vec{x}\in V(n,q) : \vec{x}H^T=\vec{0}\right\}.$$

To znamená, že kód C je určen nejen generující maticí G, ale současně je určen také kontrolní maticí H.

Příklad 6.5. Podle Příkladu 6.3. víme, že kontrolní matice kódu C_2 z Příkladu 1.10. je $H=[\,1\quad 1\quad 1\,]$.

Příklad 6.6. Podle Příkladu 6.4. víme, že generující matice a současně kontrolní matice kódu C_7 z Příkladu 6.4. je $G=H=\begin{bmatrix}1&1&0&0\\0&0&1&1\end{bmatrix}$.

Kontrolní součty
Řádky kontrolní matice odpovídají *kontrolním součtům*. Kontrolní součet říká, že jistá lineární kombinace souřadnic každého kódového slova se rovná 0. Kontrolní součet sestavíme snadno. Příslušný řádek $\vec{r}$ kontrolní matice skalárně vynásobíme s vektorem souřadnic $\vec{x}=x_1x_2\ldots x_n$ a porovnáme s nulou.

Příklad 6.7. Sestavíme kontrolní součty kódu C_7 z Příkladu 6.4.
Vynásobíme skalárně každý řádek kontrolní matice (Příklad 6.6.) s vektorem souřadnic $\vec{x}=x_1x_2x_3x_4$ a porovnáme s 0. Dostaneme

$$x_1+x_2=0,$$
$$x_3+x_4=0.$$

✓

Příklad 6.8. Sestavíme kontrolní součty kódu C_2 z Příkladu 6.3.
Dle Příkladu 6.5. víme, že kontrolní matice H kódu C_2 má jediný řádek. Vynásobením s vektorem souřadnic $\vec{x}=x_1x_2x_3$ a porovnáním s 0 dostaneme $x_1+x_2+x_3=0$. ✓

Poznámka 6.2. Pozorování z předchozího příkladu můžeme zobecnit. Mějme binární lineární kód E_n, jehož všechna kódová slova mají sudou váhu Jeho kontrolní matice je $H=[\underbrace{1\,1\,\ldots\,1}_{n}]$. To znamená, že kontrolní součet takového kódu je jediný, a sice $x_1+x_2+\cdots+x_n=0$.

Kontrolní součet slouží ke snadnému ověření, zda přijaté slovo $\vec{y}$ je kódové slovo nebo ne. Každé kódové slovo musí splňovat všechny kontrolní součty a naopak, každé slovo, které splňuje všechny kontrolní součty je kódovým slovem.

Kontrolní součty je navíc zpravidla jednoduché implementovat. Zatímco násobení kontrolní maticí vyžaduje operace násobení i sčítání, tak (binární) kontrolní součet je zpravidla možné implementovat pomocí hradel pro operace sčítání.

Sestavení kontrolní matice
Následující věta dává jednoduchý nástroj jak sestavit kontrolní matici H lineárního kódu, pokud známe jeho generující matici G (ve standardním tvaru), nebo naopak, jak sestavit generující matici G lineárního kódu, pokud známe jeho kontrolní matici H.

Věta 6.5. Standardní tvar kontrolní matice
Mějme lineární $[n,k]$-kód C s generující maticí ve standardním tvaru $G=[\,I_k|A\,]$. Kontrolní matice H kódu C je matice $H=[\,-A^T|i_{n-k}\,]$.

Důkaz. Mějme generující matici G kódu C ve standardním tvaru.

$$G = \begin{bmatrix} 1 & 0 & \cdots & 0 & a_{11} & \cdots & a_{1,n-k} \\ 0 & 1 & \cdots & 0 & a_{21} & \cdots & a_{2,n-k} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & a_{k,1} & \cdots & a_{k,n-k} \end{bmatrix}$$

Označme

$$H = \begin{bmatrix} -a_{11} & -a_{21} & \cdots & -a_{k1} & 1 & 0 & \cdots & 0 \\ -a_{12} & -a_{22} & \cdots & -a_{k2} & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ -a_{1,n-k} & -a_{2,n-k} & \cdots & -a_{k,n-k} & 0 & 0 & \cdots & 1 \end{bmatrix}$$

Matice H je požadovaného tvaru $n-k \times n$ a řádky matice H jsou jistě nezávislé. Zbývá ověřit, že každý řádek matice H je ortogonální ke každému řádku matice G. Uděláme příslušné skalární součiny i-tého řádku matice G a j-tého řádku matice H. Pro každé $i = 1, 2, \ldots, k$ a každé $j = 1, 2, \ldots, n-k$ dostaneme

$$0 + 0 + \cdots + 0 - a_{ij} + 0 + \cdots + 0 + a_{ij} + 0 + \cdots + 0 = 0.$$

To znamená že řádky matice H a řádky matice G jsou ortogonální a H je kontrolní matice kódu C. □

Příklad 6.9. Generující matice kódu C_2 z Příkladu 1.10. má podle Příkladu 4.10. standardní tvar

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

Sestavíme kontrolní matici kódu C_2.
Podle Věty 6.5. je kontrolní matice

$$\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}.$$

To odpovídá řešení z Příkladu 6.5. ✓

Všimněte si, že záporná znaménka v kontrolní matici binárního kódu nemusím uvádět, neboť v binárním kódy platí $-1 = 1$.

Příklad 6.10. Generující matice kódu C_F z Příkladu 3.5. má podle Příkladu 4.10. standardní tvar

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

Sestavíme kontrolní matici kódu C_F.
Podle Věty 6.5. je kontrolní matice

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

✓

Podle Věty 6.5. vidíme, že kód může být jednoznačné určen jak generující maticí, tak kontrolní maticí. Pro řadu kódů, například pro Hammingovy kódy, kterým se budeme věnovat v Kapitole 7., je šikovnější kód popsat pomocí kontrolní matice, případně pomocí kontrolních součtů.

Otázka: Můžeme ke každému lineárnímu kódu C najít kontrolní matici užitím Věty 6.5.?

Ternární kód
Znaménka jinak
Příklad
Tady chybí část textu, kterou je třeba doplnit.

Standardní tvar kontrolní matice

Vzhledem k tvrzení Věty 6.5. je praktické zadávat kontrolní matici v *jiném* tvaru než je standardní tvar generující matice. Odtud plyne následující definice i zdůvodnění názvu Věty 6.5.

Definice Standardní tvar kontrolní matice
Řekneme, že kontrolní matice se je ve *standardním tvaru*, jestliže $H = [\,B|I_{n-k}\,]$.

Příklad 6.11. Navážeme na Příklad 5.3. Generující matice kódu C_6 je

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

Sestavíme jeho kontrolní matici ve standardním tvaru.
Protože generující matice je ve standardním tvaru $G = [\,I_2|A\,]$, tak podle Věty 6.5. snadno sestavíme standardní tvar kontrolní matice $H = [\,-A^T|I_{n-k}\,]$. Označíme

$$A = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix},$$

Protože v binární soustavě $-1 = 1$ a nemusíme uvažovat záporná znaménka, tak platí

$$-A^T = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}.$$

Dostaneme kontrolní matici ve standardním tvaru

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

✓

Cvičení

6.1.1. Dokažte Lemma 6.1.

6.1.2. Mějme binární lineární kód E_n, který obsahuje právě všechna slova sudé váhy z prostoru $V(n,q)$. Dokažte, že duální kód $E_n^\perp$ je binární opakovací kód délky n.

6.1.3. Určete kontrolní součet kódu E_n ze Cvičení 6.1.2.

6.1.4. Jak vypadá kontrolní matice kódu C_6 z Příkladu 6.4.?

6.1.5. Mějme dán lineární $[n,k,d]$-kód C daný generující maticí G, jehož kontrolní matice je H. Ukažte, že lineární $[n,k,d]$-kód C', jehož generující matice vznikne z G provedením řádkových úprav dle Věty 4.3., bude mít kontrolní matici, která vznikne z matice H provedením stejných řádkových úprav.

6.1.6. Mějme dán lineární $[n,k,d]$-kód C daný generující maticí G, jehož kontrolní matice je H. Ukažte, že lineární $[n,k,d]$-kód C', jehož generující matice vznikne z G provedením sloupcové úpravy (S1) dle Věty 4.3., bude mít kontrolní matici, která vznikne z matice H provedením stejné sloupcové úpravy (S1).

6.1.7. Mějme dán lineární $[n,k,d]$-kód C daný generující maticí G, jehož kontrolní matice je H. Ukažte, že lineární $[n,k,d]$-kód C', jehož generující matice vznikne z G provedením sloupcové úpravy (S2) vynásobením j-tého sloupce libovolným nenulovým prvkem s dle Věty 4.3., bude mít kontrolní matici, která vznikne z matice H provedením stejné sloupcové úpravy (S2) vynásobením j-tého sloupce prvkem s^{-1}.

6.2. Syndromové dekódování

Při dekódování pomocí Slepianova standardního rozmístění jsme ukázali, že jakmile přijaté slovo $\vec{y}$ najdeme v tabulce, tak máme dvě možnosti, jak dále postupovat. Buď přečteme kódové slovo $\vec{x}$ z horního záhlaví tabulky ve stejném sloupci, jako přijaté slovo $\vec{y}$, nebo v levém záhlaví tabulky přečteme chybový vektor $\vec{e}$ a kódové slovo $\vec{x}$ vypočítáme jako $\vec{x} = \vec{y} - \vec{e}$. Druhý postup se jevil jako zbytečně složitý. Nyní ukážeme, že lineární kódy umožňují elegantní postup dekódování, který je založený právě na určení chybového vektoru.

Syndrom slova
Nejprve zavedeme pojem syndromu přijatého slova.

Definice Syndrom slova
Mějme q-ární lineární $[n,k]$-kód C a mějme jeho kontrolní matici H Pro libovolné slovo $\vec{y}$, $\vec{y} \in V(n,q)$ *syndromem* rozumíme řádkový vektor délky $n-k$, který který označíme $S(\vec{y})$ a který dostaneme jako součin $S(\vec{y}) = \vec{y}H^T$.

Některá literatura uvádí syndrom jako sloupcový vektor Hy^T, což je transponovaný vektor syndromu zavedeného v definici.

Poznámka 6.3. Syndrom můžeme určit pro libovolné slovo, nejen pro kódové slovo $\vec{y}$.

1) Syndrom kódového slova je nulový vektor $\vec{0}$, neboť kódové slovo je ortogonální ke všem vektorům duálního kódu $C^\perp$ daného generující maticí H. A současně naopak nulový syndrom mají pouze kódová slova.
2) Označíme-li $\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_{n-k}$ řádky kontrolní matice, tak syndrom můžeme vypočítat pomocí $n-k$ skalárních součinů $S(\vec{y}) = \vec{y}\cdot\vec{h}_1, \vec{y}\cdot\vec{h}_2, \ldots, \vec{y}\cdot\vec{h}_{n-k}$
3) Máme-li podle kontrolní matice implementovány kontrolní součty $s_i = h_{i1}y_1 + h_{i2}y_2 + \cdots + h_{in}y_n$, tak syndrom můžeme vypočítat pomocí $n-k$ kontrolních součtů $S(\vec{y}) = s_1 s_2 \ldots s_n$.

Příklad 6.12. Ukážeme několik jednoduchých příkladů syndromů.

1) V Příkladu 6.5. jsme ukázali, že kontrolní matice kódu C_2 je jednořádková matice $H = [1 \quad 1 \quad 1]$. Syndromy kódových slov 110 a 101 jsou

$$S(110) = [1 \quad 1 \quad 0]\cdot[1 \quad 1 \quad 1]^T = 1+1+0 = 0,$$
$$S(101) = [1 \quad 0 \quad 1]\cdot[1 \quad 1 \quad 1]^T = 1+0+1 = 0.$$

Syndromy slov 100 a 010 jsou

$$S(100) = [1 \quad 0 \quad 0]\cdot[1 \quad 1 \quad 1]^T = 1+0+0 = 1,$$
$$S(010) = [0 \quad 1 \quad 0]\cdot[1 \quad 1 \quad 1]^T = 0+1+0 = 1.$$

2) V Příkladu 6.6. jsme ukázali, že kontrolní matice kódu C_7 je jednořádková matice $H = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$.
Syndromy kódových slov 1100 a 1111 jsou

$$S(1100) = [1 \quad 1 \quad 0 \quad 0]\cdot\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}^T = [1+1+0+0 \quad 0+0+0+0] = 00,$$
$$S(1111) = [1 \quad 1 \quad 1 \quad 1]\cdot\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}^T = [1+1+0+0 \quad 0+0+1+1] = 00.$$

Syndromy slov 1101, 1000 a 0001 jsou

$$S(1100) = [1 \quad 1 \quad 0 \quad 1]\cdot\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}^T = [1+1+0+0 \quad 0+0+0+1] = 01,$$
$$S(1000) = [1 \quad 0 \quad 0 \quad 0]\cdot\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}^T = [1+0+0+0 \quad 0+0+0+0] = 10,$$
$$S(0001) = [0 \quad 0 \quad 0 \quad 1]\cdot\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}^T = [0+0+0+0 \quad 0+0+0+1] = 01.$$

Syndromy a kosety
Následující lemma ukazuje důležité pozorování, které umožní elegantní dekódování aniž bychom sestavovali Slepianovo standardní rozmístění: všechna slova ze stejného kosetu mají stejný syndrom.

Lemma 6.6. *Dva vektory $\vec{u}, \vec{v} \in V(n,q)$ jsou ve stejném kosetu kódu C právě tehdy, pokud mají stejný syndrom.*

Důkaz. Tvrzení má tvar ekvivalence. Ukážeme obě implikace.

„⇒“ Jestliže vektory (slova) $\vec{u}, \vec{v} \in V(n,q)$ patří do stejného kosetu kódu C, tak $\vec{u}+C = \vec{v}+C$. Podle známého tvrzení o třídách rozkladu faktorových grup (odkaz?) platí $\vec{u}-\vec{v} \in C$, což podle definice kontrolní matice dává $(\vec{u}-\vec{v})H^T = \vec{0}$. Roznásobením a převedením na druhou stranu rovnosti dostaneme $\vec{u}H^T = \vec{v}H^T$. Podle definice syndromu dostáváme $S(\vec{y}) = S(\vec{v})$.
„⇐“ Obrácením všech kroků v předchozí části dostaneme opačnou implikaci. □

Příklad 6.13. Navážeme na Příklad 6.11. Generující matice kódu C_6 (podle Příkladu 5.3.) je

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

Určíme syndromy všech reprezentantů kosetů.
Kontrolní matice kódu C_6 podle Příkladu 6.11. je

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

Protože reprezentanti kosetů jsou 0000, 1000, 0100 a 0001, jejich syndromy jsou

$$S(0000) = [\,0 \quad 0 \quad 0 \quad 0\,] \cdot \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}^T = [\,0+0+0+0 \quad 0+0+0+0\,] = 00 \text{ (kódové slovo)},$$

$$S(1000) = [\,1 \quad 0 \quad 0 \quad 0\,] \cdot \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}^T = [\,1+0+0+0 \quad 0+0+0+0\,] = 10,$$

$$S(0100) = [\,0 \quad 1 \quad 0 \quad 0\,] \cdot \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}^T = [\,0+1+0+0 \quad 0+1+0+0\,] = 11,$$

$$S(0001) = [\,0 \quad 0 \quad 0 \quad 1\,] \cdot \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}^T = [\,0+0+0+0 \quad 0+0+0+1\,] = 01.$$

Vidíme, že každý reprezentant má jiný syndrom. Je snadné ověřit, že všechna slova daného kosetu mají stejný syndrom jako má reprezentant daného kosetu. ✓

Lemma 6.6. říká, že mezi kosety (jejich reprezentanty) a mezi syndromy existuje jednoznačný vztah. Každému kosetu umíme jednoznačně přiřadit syndrom a hlavně naopak: každému syndromu umíme přiřadit celý koset.

Důsledek 6.7. *Existuje bijekce mezi syndromy a kosety (resp. reprezentanty kosetů).*

Bijekci z Důsledku 6.7. budeme navíc umět sestavit.

Příklad 6.14. Navážeme na Příklad 6.13. Doplníme tabulku standardního Slepianova rozmístění o sloupec syndromů.
Slepianovo standardní rozmístění kódu C_6 jsme sestavili v Příkladu 5.8. (Tabulka 5.1.). Přidáním sloupce kosetů jsme dostali Tabulku 6.1. ✓

	reprezentanti ↓				syndromy ↓
kódová slova →	0000	1010	0111	1101	00
	1000	0010	1111	0101	10
	0100	1110	0011	1001	11
	0001	1011	0110	1100	01

Tabulka 6.1.: *Rozšířené Slepianovo standardní rozmístění.*

V další části ukážeme, že pro dekódování není důležitá celá tabulka Slepianova standardního rozmístění, ale pouze reprezentanti kosetů a jejich syndromy. Ostatní sloupce tabulky nemusíme sestavovat a tedy ani ukládat.

Syndromová vyhledávací tabulka
Předpokládejme, že máme kód C daný generující maticí G. Snadno sestavíme také kontrolní matici H. Jestliže bylo vysláno kódové slovo $\vec{x}$ a přijato slovo $\vec{y}$, tak určíme syndrom slova $\vec{y}$. Nevíme sice, jaká chyba $\vec{e}$ při přenosu nastala, ale přijaté slovo $\vec{y}$ má stejný syndrom jako reprezentant $\vec{r}$ příslušného kosetu. Proto stačí každému syndromu přiřadit příslušného reprezentanta $\vec{r}$ kosetu, tento reprezentant je současně roven chybovému vektoru $\vec{e}$. Ihned tak poznáme, jaká chyba při přenosu nastala (za předpokladu malého počtu chyb).

Příklad 6.15. Navážeme na Příklad 6.14. Sestavíme tabulku pro vyhledání syndromů, které budeme říkat „syndromová vyhledávací tabulka“). První sloupec budou syndromy, druhý reprezentanti.
V Příkladu 6.14. jsme sestavili Tabulku 6.1. Tabulka 6.2. obsahuje dva vybrané sloupce: sloupec syndromů a sloupec reprezentantů. ✓

syndromy $\vec{z}$	reprezentanti $f(\vec{z})$
00	0000
10	1000
11	0100
01	0001

Tabulka 6.2.: *Tabulka syndromů a reprezentantů (syndromová vyhledávací tabulka).*

Syndromové dekódování
Celý proces syndromového dekódování popisuje následující algoritmus.

Algoritmus 6.8. Syndromové dekódování
Vysláno bylo kódové slovo $\vec{x}$ a přijato bylo slovo $\vec{y}$.

1) *Vypočítáme syndrom $S(\vec{y}) = \vec{y}H^T$.*
2) *Označíme $\vec{z} = S(\vec{y})$ a najdeme syndrom v prvním sloupci tabulky syndromů a reprezentantů. Odpovídající reprezentant kosetu je vektor $f(\vec{z})$.*
3) *Slovo $\vec{y}$ dekódujeme jako $\vec{y} - f(\vec{z})$.*

Uvedený algoritmus dekóduje slovo správně, pokud počet chyb, které nastaly, je malý. Ukážeme několik příkladů syndromového dekódování.

Příklad 6.16. Navážeme na Příklad 6.15. Kontrolní matice kódu C_6 je

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

Pomocí tabulky syndromů a reprezentantů kódu C_6 (Příklad 5.3.) dekódujeme slova a) 0111, b) 1001 a c) 1111.
a) Pro přijaté slovo $\vec{y}_1 = 0111$ spočítáme syndrom

$$S(\vec{y}_1) = [\,0 \quad 1 \quad 1 \quad 1\,] \cdot \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}^T = [\,0+1+1+0 \quad 0+1+0+1\,] = 00.$$

Protože syndrom je nulový vektor (a podle Tabulky 6.15. ihned vidíme, že příslušným reprezentantem je slovo $f(00) = 0000$, které je současně chybovým vektorem $\vec{e}_1$), tak přijaté slovo je přímo kódovým slovem $\vec{x}_1 = \vec{y}_1$.
b) Pro přijaté slovo $\vec{y}_2 = 1001$ spočítáme syndrom

$$S(\vec{y}_2) = [\,1 \quad 0 \quad 0 \quad 1\,] \cdot \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}^T = [\,1+0+0+0 \quad 0+0+0+1\,] = 11.$$

Podle Tabulky 6.15. ihned vidíme, že příslušným reprezentantem je $f(11) = 0100$, který je současně chybovým vektorem $\vec{e}_2$. Vypočítáme kódové slovo $\vec{x}_2 = \vec{y}_2 - \vec{e}_2 = 1001 - 0100 = 1101$. Všimněte si, že vypočítané kódové slovo vskutku odpovídá kódovému slovu v záhlaví Tabulky 6.1.

c) Pro přijaté slovo $\vec{y}_3 = 1111$ spočítáme syndrom

$$S(\vec{y}_3) = [1 \quad 1 \quad 1 \quad 1] \cdot \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}^T = [1+1+1+0 \quad 0+1+0+1] = 10.$$

Podle Tabulky 6.15. ihned vidíme, že příslušným reprezentantem je $f(10) = 1000$, který je současně chybovým vektorem $\vec{e}_3$. Vypočítáme kódové slovo $\vec{x}_3 = \vec{y}_3 - \vec{e}_3 = 1111 - 1000 = 0111$. Vypočítané kódové slovo opět odpovídá kódovému slovu v záhlaví Tabulky 6.1. ✓

Celý postup můžeme shrnout:

1) Vysláno bylo slovo $\vec{x}$ a přijato slovo bylo slovo $\vec{y}$.
2) Určíme syndrom přijatého slova $\vec{y}$, $S(\vec{y}) = \vec{y}H^T$.
3) Je-li syndrom nulový vektor $S(\vec{y}) = \vec{0}$, tak za předpokladu malého počtu chyb nedošlo chybě a přijaté slovo $\vec{y}$ je současně kódové slovo $\vec{x}$.
4) Je-li syndrom nulový nenulový vektor $S(\vec{y}) = \vec{z}$, tak za předpokladu malého počtu chyb dekódujeme přijaté slovo $\vec{y}$ jako kódové slovo $\vec{x} = \vec{y} - f(\vec{z})$.

Cvičení

6.2.1. Mějme binární lineární $[n, k]$-kód s kontrolní maticí H. Ukažte, že transponovaný syndrom přijatého slova $\vec{y}$ je roven součtu sloupců kontrolní matice H, které odpovídají pozicím s chybami.

6.2.2. Sestavte syndromovou vyhledávací tabulku pro binární lineární $[7, 4, 3]$-kód C_8 s generující maticí

$$\left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array}\right].$$

Použijte tabulku k dekódování slov $\vec{y}_1 =$

6.2.3. Mějme lineární $[10, 8]$-kód nad $GF(11)$ zadaný kontrolní maticí

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{bmatrix}.$$

Najděte generující matici ve standardním tvaru.

6.2.4. Navážeme na Cvičení ?? Mějme kód C (již ne lineární) který vznikne z kódu C vynecháním všech slov, která obsahují symbol 10. Dostaneme kód, jehož počet slov je $|C| = 82644629$.

Z generující matice vidíme, že slova kódu jsou tvaru $\vec{x} = x_1, x_2, \ldots, x_{10}$, kde $x_9 = 2x_1 + 3x_2 + 4x_3 + + 5x_4 + 6x_5 + 7x_6 + 8x_7 + 9x_8$ a $x_{10} =$ Tady chybí část textu, kterou je třeba doplnit..

6.3. Neúplné dekódování

Neúplné dekódování

Neúplné dekódování je kombinovaný proces detekce a opravy chyb přenosu. Detekce se místo opravy využívá v případě, že „oprava“ s vysokou pravděpodobností nedá správné kódové slovo. Následující postup zaručí pro kód C s minimální vzdáleností $\operatorname{dist}(C) = 2t + 1$ nebo $\operatorname{dist} C = 2t + 2$ opravu nejvýše t chyb. Navíc bude možno v několika případech detekovat více než t chyb.

Sestavíme Slepianovo standardní rozmístění s rostoucími váhami reprezentantů. Rozmístění rozdělíme na dvě části:

1) horní část bude obsahovat řádky, jejichž reprezentanti mají váhu nejvýše t,
2) dolní část bude obsahovat řádky, jejichž reprezentanti mají váhu vyšší než t.

Pokud přijaté slovo $\vec{y}$ bude z horní části (syndrom přijatého slova $\vec{y}$ bude mít váhu nejvýše t), tak přijaté slovo dekódujeme jako $\vec{y} - f(\vec{z})$, kde $\vec{z} = S(\vec{y})$ a $f(\vec{z})$ určíme podle syndromové vyhledávací tabulky. Pokud ale přijaté slovo $\vec{y}$ bude z dolní části (jeho syndrom bude mít váhu větší než t), tak víme, že nastalo více než t chyb a vyžádáme si nové zaslání, respektive přečtení kódového slova.

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

Tabulka 6.3.: *Slepianovo standardní rozmístění kódu C_3.*

Příklad 6.17. Mějme binární lineární $[5, 2]$-kód C_3 z Příkladu 1.11. s generující maticí

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

Pomocí neúplného dekódování zpracujte slova $\vec{y}_1 = 11011$, $\vec{y}_2 = 00110$, $\vec{y}_3 = 00011$.
Nejprve sestavíme horní a dolní část Slepianova standardního rozmístění. Dostaneme Tabulku 6.3.

Nejprve dekódujeme slovo $\vec{y}_1 = 11011$. Slovo se nachází v prvním řádku Tabulky 6.3., příslušný chybový vektor je $\vec{0}$. Dekódujeme kódové slovo $\vec{x}_1 = \vec{y}_1 = 11011$.

Dále dekódujeme slovo $\vec{y}_2 = 00110$. Slovo se nachází ve třetím sloupci a třetím řádku Tabulky 6.3., příslušný chybový vektor je $\vec{e}_2 = 01000$. Dekódujeme slovo $\vec{y}_2$ jako kódové slovo $\vec{x}_2 = 01110$ ze záhlaví třetího sloupce. (Navíc platí $\vec{x}_2 = \vec{y}_2 - \vec{e}_2 = 00110 - 01000 = 01110$.)

A konečně se pokusíme dekódovat slovo $\vec{y}_3 = 00011$. Slovo se nachází ve čtvrtém sloupci a sedmém řádku Tabulky 6.3. Protože slovo se nachází v dolní části Slepianova rozmístění, slovo nebudeme dekódovat, ale vyžádáme si nové zaslání nebo přečtení slova. V Příkladu 6.19. ukážeme, že je jen malá pravděpodobnost, že bychom slovo dekódovali správně, protože minimální vzdálenost kódu $C_3 = 3$, tj. $t = 1$, a proto kód C_3 opraví nejvýše jednu chybu. Detekovat však může dvě chyby, neboť $d - 1 = 2$. ✓

Neúplné syndromové dekódování

Ještě úspornější a je neúplné dekódování, které využívá syndromů přijatých slov. Místo Slepianova standardního rozmístění s horní a dolní částí sestavíme syndromovou vyhledávací tabulku. Při neúplném dekódování navíc stačí sestavit zkrácenou syndromovou vyhledávací tabulku jen pro horní část Slepainova standardního rozmístění.

Pro přijaté slovo $\vec{y}$ vypočítáme jeho syndrom $S(\vec{y})$. Pokud se syndrom $S(\vec{y})$ nachází v horní části syndromové vyhledávací tabulky, tak dekódujeme slovo $\vec{x} = \vec{y} - f(\vec{z})$, kde $\vec{z} = S(\vec{y})$ a $f(\vec{z})$ určíme dle sydromové vyhledávací tabulky. Pokud se však syndrom nachází v dolní části (respektive se ve zkrácené syndromové vyhledávací tabulce vůbec nenachází, protože jeho váha je vyšší než hodnota parametru t), tak víme, že nastalo více chyb, než daný kód umí opravit, a slovo nebudeme dekódovat. Vyžádáme nové zaslání nebo čtení slova.

Příklad 6.18. Navážeme na Příklad 6.17. Pomocí neúplného syndromového dekódování kódu C_3 zpracujte slova $\vec{y}_1 = 11011$, $\vec{y}_2 = 00110$, $\vec{y}_3 = 00011$.
Máme generující matici G binárního lineárního $[5, 2]$-kódu C_3 z Příkladu 1.11.

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

Sestavíme kontrolní matici kódu C_3. Protože generující matice má tvar $G = [\, I_2 | A \,]$, kde

$$A = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix},$$

tak kontrolní matice má tvar

$$H = [\, -A^T | I_3 \,] = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

	syndromy	reprezentanti
horní	000	00000
část	101	10000
	110	01000
	100	00100
	010	00010
	001	00001
dolní	011	11000
část	111	10010

Tabulka 6.4.: *Syndromová vyhledávací tabulka kódu C_3.*

Dále sestavíme syndromovou vyhledávací Tabulku 6.4.

Nyní pokud bude přijato slovo $\vec{y}_1 = 11011$, tak jeho syndrom $S(\vec{y}_1) = \vec{y}_1 H^T = [0 \quad 0 \quad 0]$ a kódové slovo $\vec{x}_1 = \vec{y}_1$.

Pokud bude přijato slovo $\vec{y}_2 = 00110$, tak jeho syndrom $\vec{z} = S(\vec{y}_2) = \vec{y}_2 H^T = [1 \quad 1 \quad 0]$. Stejný syndrom má reprezentant $f(\vec{z}) = \vec{e}_2 = 01000$ a kódové slovo $\vec{x}_2 = \vec{y}_2 - \vec{e}_2 = 00110 - 01000 = 01110$.

A konečně pokud bude přijato slovo $\vec{y}_3 = 00011$, tak jeho syndrom $S(\vec{y}_3) = \vec{y}_2 H^T = [0 \quad 1 \quad 1]$. Stejný syndrom má reprezentant $\vec{e}_6 = 11000$. Protože nastaly dvě chyby a kód C_3 opraví nejvýše jednu chybu, tak vyslané kódové slovo mohlo být $\vec{x}_3 = \vec{y}_3 - \vec{e}_6 = 00011 - 11000 = 01110$, ale mohlo se jednat i o jiné slovo a $\vec{y}_3$ nebudeme dekódovat. ✓

Mohlo by se zdát, že rozhodnutím nedekódovat slovo, pokud víme, že nastaly alespoň dvě chyby, přicházíme o možnost některé chyby opravit. Problém je, že pokud nastaly dvě nebo více chyb a podle syndromu tuto situaci rozpoznáme, tak bychom slovo dekódovali nejspíš špatně. Situaci vysvětlíme na následujícím příkladu.

Příklad 6.19. Navážeme na Příklad 6.18. Předpokládejme, že při přenosu kódového slova $\vec{x}$ bylo přijato slovo $\vec{y}_3 = 00011$ a nastaly právě dvě chyby. Určete pravděpodobnost, že při dekódování pomocí Slepianova standardního rozmístění nebo při syndromovém dekódování dekódujeme přijaté slovo správně. Jak se situace změní, jestliže budeme předpokládat, že při přenosu nastaly alespoň dvě chyby?

Jestliže při přenosu nastaly právě dvě chyby, tak by se mohlo zdát, že dvě chyby mohly nastat jedním z $\binom{5}{2} = 10$ způsobů. Ukážeme, že umíme počet různých případů výrazně snížit. Víme totiž, že zaslané slovo $\vec{x}$ musí patřit do kódu C_3 a přijaté slovo je právě $\vec{y}_3 = 00011$.

1) Pokud bylo zasláno slovo 00000 musely nastat dvě chyby na posledních dvou pozicích. Při dekódování však bude dekódováno slovo $\vec{x} = 11011$ a slovo nebude dekódováno správně.
2) Slovo 10101 nemohlo být zasláno, neboť $\mathrm{dist}(10101, \vec{y}_3) = 3$.
3) Ani slovo 01110 nemohlo být zasláno, neboť $\mathrm{dist}(01110, \vec{y}_3) = 3$.
4) Pokud bylo zasláno slovo 11011 musely nastat dvě chyby na prvních dvou pozicích. Při dekódování bude správně dekódováno slovo $\vec{x} = 11011$.

Předpokládáme, že v symetrickém binárním kanálu je každá chyba stejně pravděpodobná. V prvním případě slovo nebude dekódováno správně, v druhém případě ano, proto bude přijaté slovo dekódováno správně s pravděpodobností $1/2$. To vše za předpokladu, který nemáme zaručen, že nastaly právě dvě chyby.

Pokud budeme předpokládat, že nastaly dvě nebo tři chyby, tak dvě chyby nastaly s pravděpodobností $10p^2(1-p)^2$ a tři chyby s pravděpodobností $10p^3(1-p)^2$. Pouze v jediném případě bude slovo dekódováno správně, ve zbývajících (s trochu větší pravděpodobností než $1/2$) bude dekódováno chybně.

Při přenosu však obecně nevíme kolik chyb nastalo, a proto přijaté slovo se dvěma a více chybami bude spíše dekódováno chybě. Pravděpodobnost obecně není snadné určit. Nyní rozumíme, proč při neúplném dekódování v případě, kdy víme, že nastalo více než $t = \lfloor \frac{d-1}{2} \rfloor$ chyb, tak slovo nedekódujeme. ✓

Neúplné syndromové dekódování a perfektní kódy

Slepianovo standardní rozmístění sestavené pro perfektní kódy bude obsahovat pouze horní část. Jestliže minimální vzdálenost takového kódu je $d = 2t + 1$, resp. $d = 2t$, tak všichni reprezentanti kosetů budou mít váhu nejvýše t, neboť z každého kosetu bude vybrán právě jediný reprezentant, který se liší od kódového slova $\vec{0}$ na nejvýše t souřadnicích. Dolní část Slepianova standardního rozmístění bude prázdná a proces dekódování nebude neúplný.

Celý postup shrne následující algoritmus.

Algoritmus 6.9. Neúplné syndromové dekódování

Vysláno bylo kódové slovo $\vec{x}$ a přijato bylo slovo $\vec{y}$.

1) Vypočítáme syndrom $S(\vec{y}) = \vec{y}H^T$.
2) Označíme $\vec{z} = S(\vec{y})$ a najdeme syndrom v prvním sloupci tabulky syndromů a reprezentantů. Odpovídající reprezentant kosetu je vektor $f(\vec{z})$.
3) Slovo $\vec{y}$ dekódujeme jako $\vec{y} - f(\vec{z})$.
4) Pokud se syndrom v tabulce nenachází, jeho reprezentant má váhu větší než v a vyžádáme nový přenos slova $\vec{x}$.

Cvičení

6.3.1. Sestavte Tabulku syndromů a reprezentantů pro perfektní kód C_F. Čím se liší od tabulky kódu C_8 z příkladu 6.2.2.?

Kapitola 7. Hammingovy kódy

V této kapitole popíšeme speciální případ lineárních kódů – Hammingovy kódy. Budou mít několik velmi pěkných vlastností, snadný proces konstrukce pomocí kontrolní matice, snadný proces dekódování, který bude velmi úsporný na operační paměť. Pro dekódování Hammingových kódů není nutno ukládat ani Slepianovo standardní rozmístění a dokonce není nutno ukládat ani syndromovou vyhledávací tabulku.

Nejprve se pro jednoduchost a názornost budeme věnovat binárním Hammingovým kódům a až v pozdějších podkapitolách q-árním Hammingovým kódům nad tělesem $GF(q)$. Ukážeme, že Hammingovy kódy mají minimální vzdálenost 3 a obecně umí opravit jednu chybu a detekovat dvě chyby. Jejich předností je jednoduché schéma pro kódování a zejména pro dekódování.

7.1. Hammingův kód

Hammingův kód může být definován různými ekvivalentními způsoby. Vzhledem k procesu dekódování je šikovné definovat kód popisem kontrolní matice, ze kterého půjde generující matici odvodit podobným postupem, jako v důkazu Věty 6.5.

Definice Hammingův kód
Mějme přirozené číslo r, $r \geq 2$. Sestavíme kontrolní matici H tvaru $r \times (2^r - 1)$ tak, že sloupce matice H budou právě všechny nenulové vektory prostoru $V(r, 2)$. *Binární Hammingův kód* je každý takový kód, jehož kontrolní matice je H. Binární Hammingův kód značíme $\mathrm{Ham}(r, 2)$.

V podkapitole 7.2. ukážeme, jak definici binárního Hammingova kódu $\mathrm{Ham}(r, 2)$ rozšířit i pro libovolnou velikost abecedy q, kde q je mocninou prvočísla a definici zobecníme i pro $\mathrm{Ham}(r, q)$.

Poznámka 7.1. Hammingův kód $\mathrm{Ham}(r, 2)$ má kódová slova délky $n = 2^r - 1$ a dimenzi $k = n - r = 2^r - r - 1$. To znamená, že počet kontrolních bitů každého kódového slova je r, kde $r = n - k$ a že redundance Hammingova $\mathrm{Ham}(r, 2)$ kódu je r.

Otázka: Jak by vypadal kód $\mathrm{Ham}(1, 2)$, kdybychom v definici dovolili $r = 1$?

Poznámka 7.2. Všimněte si, že existuje více možností, jak Hammingův kód $\mathrm{Ham}(r, 2)$ vypadá (Cvičení 7.1.1.), neboť pořadí sloupců kontrolní matice Hammingova kódu $\mathrm{Ham}(r, 2)$ není v definici jednoznačně určeno. Ukážeme, že jisté pořadí sloupců je pro opravu chyb při dekódování výhodnější než ostatní. Obecně symbol $\mathrm{Ham}(r, 2)$ zahrnuje kterýkoliv z možných kódů dle definice, vždy však má délku slov $2^r - 1$, dimenzi $2^r - r - 1$ a redundanci r.

Otázky:

- Kolik existuje binárních Hammingových kódů $\mathrm{Ham}(2, 2)$?
- Kolik existuje binárních Hammingových kódů $\mathrm{Ham}(3, 2)$?

Příklad 7.1. Uvedeme několik příkladů Hammingových kódů.

1) Kódy dané kontrolní maticí H_1 i kontrolní maticí H_2 jsou Hammingovy kódy $\mathrm{Ham}(2, 2)$.

$$H_1 = \begin{bmatrix} 110 \\ 101 \end{bmatrix}, \qquad H_2 = \begin{bmatrix} 011 \\ 101 \end{bmatrix}$$

   Každý Hammingův kód pro $r = 2$ má generující matici $G = [1 \quad 1 \quad 1]$ a jedná se o opakovací kód délky 3.
2) Binární Hammingův kód pro $r = 3$ je například binární lineární kód $\mathrm{Ham}(3, 2)$ daný kontrolní maticí

$$H_3 = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

Všimněte si, že sloupce kontrolní matice H_3 jsou uspořádány vzestupně podle binárního zápisu indexů každého sloupce zapsaných do sloupcových vektorů matice H.

3) Binární Hammingův kód pro $r = 4$ bude mít kontrolní matici tvaru 4×15. Kontrolní matice $\mathrm{Ham}(4, 2)$ bude například

$$H_4 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}.$$

Otázky:

- Kolik kódových slov obsahuje Hammingův kód $\mathrm{Ham}(3, 2)$?
- Kolik kódových slov obsahuje Hammingův kód $\mathrm{Ham}(4, 2)$?

Otázka:

Příklad 7.2. Sestavíme binární Hammingův kód pro $r = 3$, jehož kontrolní matice má standardní tvar. Sestavte jeho generující matici a porovnejte ji s maticí lineárního $[7, 4, 3]$-kódu z Příkladu 6.10.
Stačí vzít sloupce kontrolní matice H_3 z Příkladu 7.1. v jiném pořadí.

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

Matice H určuje ekvivalentní Hammingův kód $\mathrm{Ham}(3, 2)$. Protože kontrolní matice má tvar $[\, -A^T | I_3 \,]$, tak snadno sestavíme generující matici $G = [\, I_4 | A \,]$ Hammingova kódu $\mathrm{Ham}(3, 2)$. Dostaneme

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

Generující matice H není identická s generující maticí z Příkladu 6.10. Obě matice však generují ekvivalentní kódy, oba příslušné kódy jsou navíc perfektní. ✓

Kódování pomocí Hammingova kódu
Pokud máme kontrolní matici binárního Hammingova kódu $\mathrm{Ham}(r, 2)$, tak stačí sestavit příslušnou generující matici kódu $\mathrm{Ham}(r, 2)$ jako v Příkladu 7.2. Kódování probíhá stejně jako u lineárního kódu tak, že slovo zprávy (délky $k = 2^r - r - 1$) vynásobíme generující maticí. Dostaneme kódové slovo Hammingova kódu.

Příklad 7.3. Pomocí generující matice G z Příkladu 7.2. zakódujeme slova $\vec{u}_1 = 0011$ a $\vec{u}_2 = 1101$.
Kódová slova $\vec{x}$ dostaneme vynásobením slova zprávy $\vec{u}_i$ generující maticí G. Generující matici G jsme sestavili v Příkladu 7.2. Kódováním slova $\vec{u}_1$ dostaneme kódové slovo $\vec{x}_1$.

$$\vec{x}_1 = \vec{y}_1 G = [\, 0011 \,] \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} = [\, 0011001 \,]$$

Kódováním slova $\vec{u}_2$ dostaneme kódové slovo $\vec{x}_2$.

$$\vec{x}_2 = \vec{u}_2 G = [\, 1101 \,] \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} = [\, 1101001 \,]$$

Všimněte si, že kódové slovo $\vec{x}_1$ je součtem třetího a čtvrtého řádku matice G, zatímco slovo $\vec{x}_2$ je součtem prvního, druhého a čtvrtého řádku matice G. ✓

Otázka: Budeme pracovat (kódovat) s binárním Hammingovým kódem $\mathrm{Ham}(r,2)$. Jakou délku mají slova zprávy, kterou můžeme kódem $\mathrm{Ham}(r,2)$ kódovat?

Vlastnosti Hammingova kódu

V Příkladu 7.2. jsme zmínili, že uvedený $\mathrm{Ham}(3,2)$ je perfektní. To není náhoda. V této části mimo jiné ukážeme, že každý Hammingův kód je perfektní.

Věta 7.1. *Mějme přirozené číslo r, $r \geq 2$. Binární Hammingův kód* $\mathrm{Ham}(r,2)$

(i) je lineární $[2^r-1, 2^r-r-1]$-kód,
(ii) má minimální vzdálenost 3 (opraví jednu chybu),
(iii) je perfektní.

Důkaz. Tvrzení věty má tři části, postupně dokážeme všechny tři.

i) Podle definice je Hammingův kód lineární a jeho duální kód $\mathrm{Ham}(r,2)^\perp$ je ihned podle tvaru matice H lineární $[2^r-1, r]$-kód. Podle Věty o duálním kódu (Věta 6.3.) je proto Hammingův kód $\mathrm{Ham}(r,2)$ lineární $[2^r-1, 2^r-1-r]$-kód.

ii) Protože Hammingův kód $\mathrm{Ham}(r,2)$ je podle definice lineární kód, tak podle Věty 4.2. stačí ukázat, že každé nenulové slovo Hammingova kódu má váhu alespoň 3.

Nejprve ukážeme, že žádné nenulové slovo nemá váhu 1. Postupujeme sporem. Předpokládejme, že slovo $\vec{x}$, $\vec{x} \in \mathrm{Ham}(r,2)$, má váhu 1. Potom můžeme psát

$$\vec{x} = 00\ldots010\ldots0,$$

kde pouze $x_i = 1$. Podle definice kontrolní matice je každé slovo kódu ortogonální ke kontrolní matici H, zejména $\vec{x}H^T = \vec{0}$. To ale znamená, že i-tý sloupce kontrolní matice H obsahuje samé nuly, což je spor s naší definicí Hammingova kódu $\mathrm{Ham}(r,2)$.

Dále ukážeme, že žádné nenulové slovo nemá váhu 2. Pro spor opět předpokládejme, že slovo $\vec{x}$, $\vec{x} \in \mathrm{Ham}(r,2)$, má váhu 2. Potom můžeme psát

$$\vec{x} = 00\ldots010\ldots010\ldots0,$$

kde pouze $x_i = 1$ a $x_j = 1$.

Označme $n = 2^r - 1$ a prvky l-tého řádku kontrolní matice H Hammingova kódů $\mathrm{Ham}(r,2)$ označme $\vec{h}_l = h_{l1}h_{l2}\ldots,h_{ln}$. Vektory $\vec{x}$ a $\vec{h}_l$ jsou ortogonální, proto pro každé $l = 1,2,\ldots,r$ platí

$$h_{li} + h_{lj} = 0 \pmod 2,$$

což pro binární kód můžeme upravit

$$h_{li} = h_{lj} \pmod 2,$$

kde $l = 1,2,\ldots,r$. To ale znamená, že i-tý a j-tý sloupec kontrolní matice H jsou stejné, což je opět spor s definicí Hammingova kódu $\mathrm{Ham}(r,2)$. Proto má každé nenulové slovo Hammingova kódu $\mathrm{Ham}(r,2)$ váhu alespoň 3.

iii) Abychom ukázali, že Hammingův kód $\mathrm{Ham}(r,2)$ je perfektní, stačí abychom ukázali, že v rovnosti (3) (Hammingova hranice z Věty 2.9.) nastává rovnost. Protože pro Hammingův kód platí $n = 2^r - 1$, $q = 2$, $t = 1$ a $M = 2^{n-r}$, tak levá strana rovnosti (3) je

$$\begin{aligned} M\left(\binom{n}{0} + \binom{n}{1}(q-1) + \binom{n}{2}(q-1)^2 + \cdots + \binom{n}{t}(q-1)^t\right) &= 2^{n-r}\left(1 + \binom{n}{1}\right) = \\ &= 2^{n-r}\,(1+n) = \\ &= 2^{n-r}\,(1+2^r-1) = \\ &= 2^n, \end{aligned}$$

což je právě pravá strana rovnosti (3) a Hammingův kód $\mathrm{Ham}(r,2)$ je proto perfektní. □

Není těžké si rozmyslet, že tvrzení Věty 7.1. (iii) nejde zesílit. Pro každé r, $r \geq 2$, obsahuje Hammingův kód slova s váhou 3. Jestliže seřadíme sloupce kontrolní matice Hammingova kódu $\mathrm{Ham}(r,2)$ tak, aby první tři sloupce matice H byly

$$\begin{array}{ccc} 0 & 0 & 0 \cdots \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \cdots \\ 0 & 1 & 1 \cdots \\ 1 & 0 & 1 \cdots, \end{array}$$

tak kódové slovo $\vec{x} = 1110\ldots0$ patří do Hammingova kódu $\mathrm{Ham}(3,2)$ a váha $\mathrm{w}(\vec{x}) = 3$.

Příklad 7.4.

1) Binární Hammingův kód $\mathrm{Ham}(2,2)$ (kód C_2 z Příkladu 1.10.) obsahuje čtyři kódová slova celého prostoru $V(2,2)$ a je perfektní.
2) Binární Hammingův kód $\mathrm{Ham}(3,2)$ (kód C_3 z Příkladu 1.11.) je podle Cvičení 2.4.1. perfektní. Obsahuje celkem $2^{7-3} = 16$ kódových slov.

Syndromy reprezentantů váhy 1

Je snadné si uvědomit, že syndrom kosetu s reprezentantem váhy 1 odpovídá právě jednomu sloupci kontrolní matice. Reprezentanti váhy 1 jsou pro kódy s minimální vzdáleností 3 klíčoví, neboť umíme opravit pouze taková kódová slova, u kterých dojde k jediné chybě. Protože ale binární Hammingovy kódy jsou perfektní (a mají minimální vzdálenost 3), tak každý koset má reprezentanta váhy nejvýše 1. Tato pozorování výrazně usnadní dekódování Hammingových kódů.

Lemma 7.2. *Mějme kontrolní matici H lineárního kódu a mějme nějakého reprezentanta $\vec{e}_j$ váhy 1 ve Slepianově standardním rozmístění, $\vec{e}_j = 00\ldots010\ldots0$, kde pouze j-tá složka je rovna 1. Syndrom $S(\vec{e}_j) = \vec{s}_j$, kde vektor $\vec{s}_j$ odpovídá transponovanému j-tému sloupci matice H.*

Důkaz. Podle definice je syndrom $S(\vec{e}_j) = \vec{e}_j H^T$. Všechny součiny jsou nulové s výjimkou součinů j-tého prvku vektoru $\vec{e}_j$ a j-tého sloupce matice H. Proto syndrom $S(\vec{e}_j) = \vec{s}_j$, kde vektor $\vec{s}_j$ odpovídá transponovanému j-tému sloupci matice H. □

V Tabulce 6.14. jsme viděli, že v kosetu bylo více slov váhy 1, například 1000 a 0010. Každý z nich mohl být zvolen reprezentantem. Všimněte si, že oba sloupce odpovídající kontrolní matice jsou stejné (Příklad 6.11.). Pokud bychom sestavili Slepianovo standardní rozmístění Hammingova kódu, tak každý koset bude mít právě jednoho reprezentanta váhy 0 (první řádek) nebo 1 (ostatní řádky). Každé slovo váhy 1 tak bude zvoleno reprezentantem.

Pro názornost sestavíme Slepianovo standardní rozmístění pro kód s kontrolní maticí H_3 z Příkladu 7.1. Ukážeme, že pro samotné dekódování nebude Slepianovo standardní rozmístění potřeba a dokonce nebude potřeba ani sestavovat syndromovou vyhledávací tabulku!

Příklad 7.5. Mějme Hammingův kód $\mathrm{Ham}(3,2)$ z Příkladu 7.1. Určíme, kolik sloupců a kolik řádků by mělo Slepianovo standardní rozmístění. Určíme všechny reprezentanty kosetů a sestavíme syndromovou vyhledávací tabulku. Ukážeme, proč ji u Hammingových kódu není potřeba sestavovat.

Hammingův kód $\mathrm{Ham}(3,2)$ má kódová slova délky $n = 2^3 - 1 = 7$ a celkem $2^{n-r} = 2^{7-3} = 16$ kódových slov. Množinu kódových slov označme C. Slepianovo standardní rozmístění by mělo 16 sloupců.

Celý vektorový prostor $V(7,2)$ má $2^7 = 128$ slov, proto Slepianovo standardní rozmístění by mělo $128/16 = 8$ řádků.

Protože každý Hammingův kód má podle Věty 7.1. minimální vzdálenost 3, tak každé kódové slovo obsahuje alespoň tři jedničky. Řádek kódových slov bude mít reprezentanta $\vec{0}$ a ukážeme, že každý zbývající řádek bude mít reprezentanta váhy 1. Zvolíme-li za reprezentanta dalšího řádku jakékoliv slovo $\vec{e}$ váhy 1, tak koset $\vec{e} + C$ bude podle Lemmatu 4.1. obsahovat slova váhy 2 nebo 4. Zejména nebude obsahovat jiné kódové slovo váhy 1, proto reprezentant dalšího řádku může opět být jiné slovo váhy 1. Celkem takto můžeme volit 7, ($n = 7$), různých reprezentantů, proto všech 8 řádků Slepianova standardního rozmístění bude mít reprezentanty váhy 0 nebo 1 (jediného reprezentanta váhy 0 a sedm reprezentantů váhy 1).

Tabulka 7.1. je syndromová vyhledávací tabulka Hammingova kódu $\mathrm{Ham}(3,2)$. Všimněte si, že podle Lemmatu 7.2. syndrom n-bitového chybového vektoru s hodnotou 1 v j-tém sloupci odpovídá j-tému sloupci

syndromy	reprezentanti
000	0000000
001	1000000
010	0100000
011	0010000
100	0001000
101	0000100
110	0000010
111	0000001

Tabulka 7.1.: *Syndromová vyhledávací tabulka kódu* $\mathrm{Ham}(3,2)$.

kontrolní matice. Pro Hammingův kód $\mathrm{Ham}(2,2)$ s kontrolní maticí H_3 z Příkladu 7.1. syndrom současně odpovídá bitovému zápisu indexu sloupce.

Syndromovou vyhledávací tabulku nemusíme sestavovat, pro nenulový syndrom stačí změnit bit, jehož index je dán bitovým zápisem syndromu. ✓

Abychom určili syndrom $S(\vec{y})$ přijatého slova $\vec{y}$, stačí vynásobit (zprava) kontrolní maticí. Pro ruční výpočet je však praktičtější využít kontrolní součty, které byly zavedeny na straně 61. Protože *každé* přijaté slovo $\vec{y}$ budeme násobit stejnou kontrolní maticí, můžeme součin provést obecně se slovem $\vec{y} = y_1 y_2 \dots y_n$ s proměnnými $y_1, y_2, \dots, y_n$. Dostaneme vektor kontrolních součtů, do kterého při dekódování dosadíme hodnoty příslušných souřadnic slova $\vec{y}$.

Příklad 7.6. Sestavte syndrom kontrolních součtů binárního a) Hammingova kódu $\mathrm{Ham}(3,2)$ z Příkladu 7.1. b) Hammingova kódu $\mathrm{Ham}(3,2)$ z Příkladu 7.2.
Přijaté slovo $\vec{y} = y_1 y_2 \dots y_n$ vynásobíme kontrolní maticí H.
a) Pro kontrolní matici H_3 z Příkladu 7.1. dostaneme syndrom kontrolních součtů

$$\vec{y}H_3^T = \begin{bmatrix} y_1 & y_2 & y_3 & y_4 & y_5 & y_6 & y_7 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}^T =$$
$$= \begin{bmatrix} y_4 + y_5 + y_6 + y_7 & y_2 + y_3 + y_6 + y_7 & y_1 + y_3 + y_5 + y_7 \end{bmatrix}.$$

b) Pro kontrolní matici H z Příkladu 7.2. dostaneme syndrom kontrolních součtů

$$\vec{y}H^T = \begin{bmatrix} y_1 & y_2 & y_3 & y_4 & y_5 & y_6 & y_7 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}^T =$$
$$= \begin{bmatrix} y_2 + y_3 + y_4 + y_5 & y_1 + y_3 + y_4 + y_6 & y_1 + y_2 + y_4 + y_7 \end{bmatrix}.$$

✓

Dekódování pomocí Hammingova kódu

Zatím jsme ukázali, že Hammingovy kódy jsou lineární a jsou perfektní. Nyní ukážeme hlavní výhodu Hammingova kódu. Postup uvedený v Příkladu 7.5. zobecníme a ukážeme elegantní implementaci dekódovacího algoritmu, který opraví jednu chybu. Nebude nutno sestavovat ani Slepianovo standardní rozmístění, ani syndromovou vyhledávací tabulku.

Mějme binární Hammingův kód $\mathrm{Ham}(r,2)$. Podle Věty 7.1. víme, že se jedná o perfektní kód s 2^{2^r-1-r} kódovými slovy délky n, $n = 2^r - 1$. Každý koset má také 2^{2^r-1-r} prvků a reprezentantů kosetů je právě 2^r: jeden reprezentant váhy 0 a $2^r - 1$ reprezentantů váhy 1 (Příklad 7.5.).

Reprezentant kosetu je současně chybovým vektorem. To znamená, že syndrom každého přijatého slova bude buď nulový vektor, nebo vektor, který je podle Lemmatu 7.2. transponováním takového sloupce kontrolní matice H, ve kterém nastala chyba. Vhodným sestavením sloupců Hammingova kódu dostaneme následující elegantní dekódovací algoritmus.

Algoritmus 7.3. Dekódování pomocí Hammingova kódu
Předpokládejme, že máme binární Hammingův daný kontrolní maticí H, ve které jsou sloupce po řadě *seřazená bitová vyjádření indexů sloupců. Vysláno bylo kódové slovo $\vec{x}$ a přijato bylo slovo $\vec{y}$.*

1) Vypočítáme syndrom $S(\vec{y}) = \vec{y}H^T$.
2) Pokud $S(\vec{y}) = \vec{0}$, tak předpokládáme, že nenastala žádná chyba a dekódujeme kódové slovo $\vec{x} = \vec{y}$.

3) Pokud $S(\vec{y}) \neq \vec{0}$, tak předpokládáme, že nastala právě jedna chyba. Syndrom $S(\vec{y})$ odpovídá bitovému vyjádření indexu sloupce kontrolní matice, tedy sloupce, ve kterém nastala chyba. Změníme paritu bitu s indexem $S(\vec{y})$ a dostaneme kódové slovo $\vec{x}$.

Příklad 7.7. Mějme Hammingův kód $\mathrm{Ham}(3,2)$ s kontrolní maticí

$$H_3 = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

Dekódujeme přijatá slova $\vec{y}_1 = 1100110$ a $\vec{y}_2 = 1010010$.
Postupujeme podle Algoritmu 7.3. Nejprve vypočítáme syndrom přijatého slova $\vec{y}_1 = 1100110$. Dostaneme

$$\begin{aligned} S(\vec{y}_1) &= [\,1100110\,]\,H^T = \\ &= [\,0+0+0+0+1+1+0 \quad 0+1+0+0+0+1+0 \quad 1+0+0+0+1+0+0\,] = 000. \end{aligned}$$

Protože syndrom je $\vec{0}$, tak přijaté slovo $\vec{y}_1$ dekódujeme jako kódové $\vec{x}_1 = \vec{y}_1$ slovo přenesené bez chyby.

Alternativně bychom mohli využít kontrolní součty z Příkladu 7.6. Dostaneme $S(\vec{y}_1) = [y_4 + y_5 + y_6 + {} + y_7 y_2 + y_3 + y_6 + y_7 y_1 + y_3 + y_5 + y_7] = 000$.

Dále vypočítáme syndrom přijatého slova $\vec{y}_2 = 0000100$. Dostaneme

$$\begin{aligned} S(\vec{y}_2) &= [\,1010010\,]\,H^T = \\ &= [\,0+0+0+0+0+1+0 \quad 0+0+1+0+0+1+0 \quad 1+0+1+0+0+0+0\,] = 100. \end{aligned}$$

Protože syndrom $S(\vec{y}_2) \neq \vec{0}$, tak slovo $\vec{y}_2$ dekódujeme jako kódové slovo přijaté s jednou chybou ve čtvrtém sloupci (syndrom $100_2 = 4_{10}$). Změnou čtvrtého bitu slova $\vec{y}_2$ dostaneme kódové slovo $\vec{x}_2 = \vec{y}_2 + 0001000 = {} = 1010010 + 0001000 = 1011010$. ✓

Cvičení

7.1.1. Kolik existuje různých generujících matic binárního Hammingova kódu $\mathrm{Ham}(r,2)$?

7.2. Rozšířený Hammingův kód

Nejprve popíšeme rozšíření binárního Hammingova kódu. Přidání paritních bitů bylo popsáno v podkapitole 2.2. na straně 2.5.

Definice *Rozšířený binární Hammingův kód* dostaneme z binárního Hammingova kódu $\mathrm{Ham}(r,2)$ přidáním paritního bitu. Rozšířený Hammingův kód značíme $\widehat{\mathrm{Ham}}(r,2)$.

Ve Větě 2.5. jsme ukázali, že pokud je minimální vzdálenost kódu lichá, tak přidání paritního bitu minimální vzdálenost o 1 zvýší. Protože binární Hammingův kód $\mathrm{Ham}(r,2)$ má minimální vzdálenost 3, tak rozšířený Hammingův kód $\widehat{\mathrm{Ham}}(r,2)$ má minimální vzdálenost 4. Je zřejmé, že rozšířený Hammingův kód $\widehat{\mathrm{Ham}}(r,2)$ má stejný počet slov, proto můžeme shrnout: rozšířený Hammingův kód $\widehat{\mathrm{Ham}}(r,2)$ je binární lineární $[2^r, 2^r - 1 - r, 4]$-kód.

Vzhledem k úplnému dekódování není rozšířený binární Hammingův kód silnější než binární Hammingův kód, neboť stále opraví jedinou chybu. Naopak, přenos nebo uložení paritního bitu spotřebuje určitou kapacitu kanálu navíc oproti Hammingovu kódu. Na druhou stranu, pro neúplné dekódování je minimální vzdálenost 4 vhodná. Ukážeme algoritmus, který při použití rozšířeného binárního Hammingova kódu zajistí současně opravu jedné chyby a detekci libovolných dvou chyb.

Otázka: Je rozšířený binární Hammingův kód $\widehat{\mathrm{Ham}}(r,2)$ perfektní?

Kontrolní matice rozšířeného Hammingova kódu

Mějme Hammingův kód $\mathrm{Ham}(r,2)$. Jeho kontrolní matici označme H. Kontrolní matici rozšířeného Hammingova kódu $\widehat{\mathrm{Ham}}(r,2)$ označíme $\widehat{H}$. Podle Cvičení 7.2.1. platí

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

Poslední řádek kontrolní matice $\widehat{H}$ odpovídá paritnímu součtu $y_1 + y_2 + \cdots + y_n = 0$ sudého počtu symbolů libovolného slova $\vec{y} = y_1 y_2 \dots y_n$, přičemž využijeme, že $n = 2^r$ je sudé číslo.

Dekódování pomocí rozšířeného binárního Hammingova kódu

Ukážeme postup dekódování pomocí rozšířeného binárního Hammingova kódu, který opraví jednu chybu, ale současně detekuje každé dvě chyby, které při přenosu/čtení nastaly.

Algoritmus 7.4. Dekódování pomocí rozšířeného Hammingova kódu

Předpokládejme, že máme rozšířený binární Hammingův kód daný kontrolní maticí $\widehat{H}$, která vznikla z kontrolní matice H jejíž sloupce jsou po řadě *seřazená bitová vyjádření indexů sloupců, přidáním posledního sloupce samých nul a posledního řádku samých jedniček.*

Vysláno bylo kódové slovo $\vec{x}$ a přijato bylo slovo $\vec{y}$.

1) *Vypočítáme syndrom $\vec{z} = S(\vec{y}) = \vec{y}\widehat{H}^T$.*
2) *Pokud $\vec{z} = \vec{0}$, tak předpokládáme, že nenastala žádná chyba a dekódujeme kódové slovo $\vec{x} = \vec{y}$.*
3) *Označme $\vec{z} = z_1 z_2 \dots z_{n-1} z_n$. Pokud nastane $[z_1 z_2 \dots z_{n-1}] = \vec{0}$ a $z_n = 1$, tak předpokládáme, že nastala jediná chyba v paritním bitu z_n, který opravíme.*
4) *Pokud nastane $[z_1 z_2 \dots z_{n-1}] \neq \vec{0}$ a $z_n = 1$, tak předpokládáme, že nastala jediná chyba na i-té pozici kódového slova, kde i je dáno binární reprezentací $[z_1 z_2 \dots z_{n-1}]$ (sloupce kontrolní matice). Chybu opravíme.*
5) *A konečně pokud $[z_1 z_2 \dots z_{n-1}] \neq \vec{0}$ a $z_n = 0$, tak předpokládáme, že nastaly alespoň dvě chyby, a vyžádáme nové zaslání/přečtení slova.*

Postup Algoritmu 7.4. ukážeme na následujícím příkladu.

Příklad 7.8. Navážeme na Příklad 7.7. Mějme Hammingův kód $\widehat{\mathrm{Ham}}(3,2)$, sestavíme kontrolní matici rozšířeného Hammingova kódu $\widehat{\mathrm{Ham}}(3,2)$. Dekódujeme přijatá slova $\vec{y}_1 = 11001100$, $\vec{y}_2 = 01010100$, $\vec{y}_3 = 01110000$ a $\vec{y}_4 = 01000010$, .

Kontrolní matice $\widehat{H}$ rozšířeného Hammingova kódu $\widehat{\mathrm{Ham}}(3,2)$ vznikne z kontrolní matice H Hammingova kódu $\mathrm{Ham}(3,2)$ přidáním sloupce nul a řádku jedniček. Dostaneme matici

$$\widehat{H}_3 = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Budeme dekódovat slovo $\vec{y}_1 = 11001100$. Nejprve vypočítáme syndrom $\vec{z} = \vec{y}_1\widehat{H}^T = 11001100\widehat{H}^T = 0000$. Protože $\vec{z} = \vec{0}$, tak předpokládáme, že slovo bylo přijaté bez chyby a platí $\vec{x}_1 = \vec{y}_1$.

Budeme dekódovat slovo $\vec{y}_2 = 01010100$. Nejprve vypočítáme syndrom $\vec{z} = \vec{y}_2\widehat{H}^T = 01010100\widehat{H}^T = 0001$. Protože $z_1 z_2 \dots z_{n-1} = \vec{0}$ a současně $z_n \neq 0$, tak předpokládáme, že při přenosu nastala jediná chyba u paritního bitu, kterou opravíme. Dekódované slovo je $\vec{x}_2 = 01010101$.

Dále budeme dekódovat slovo $\vec{y}_2 = 01110000$. Nejprve vypočítáme syndrom $\vec{z} = \vec{y}_2\widehat{H}^T = 01110000\widehat{H}^T = 1011$. Protože $z_1 z_2 \dots z_{n-1} \neq \vec{0}$ a současně $z_n \neq 0$, tak předpokládáme, že při přenosu nastala jediná chyba na pozici $z_1 z_2 z_3 = 101$. Protože $101_2 = 5_{10}$, tak chybu v pátém sloupci opravíme. Dekódované slovo je $\vec{x}_3 = 01111000$.

A konečně zkusíme dekódovat slovo $\vec{y}_4 = 01000010$. Nejprve vypočítáme syndrom $\vec{z} = \vec{y}_4\widehat{H}^T = 01000010\widehat{H}^T = 1010$. Protože $z_1 z_2 \dots z_{n-1} \neq \vec{0}$ a současně $z_n = 0$, tak předpokládáme, že při přenosu nastaly alespoň dvě chyby a vyžádáme nové zaslání nebo přečtení. ✓

Otázka: Proč při dekódování pomocí rozšířeného Hammingova kódů můžeme předpokládat, že pokud $z_n = 1$, tak nastala jediná chyba a pokud $z_n = 0$, tak nenastala řádná chyba, nebo nastaly alespoň dvě chyby?

Výpočet syndromu
Budeme-li nějaký lineární kód C pro přenos zpráv používat opakovaně, můžeme výpočet syndromu přijatého slova implementovat efektivně. Každé přijaté slovo $\vec{z}$ je násobeno kontrolní maticí, přičemž hodnotu syndromu mohou ovlivnit pouze nenulové prvky kontrolní matice H. Proto když označíme $\vec{z} = z_1 z_2 \dots z_n$ a vypočítáme $\vec{z}H^T$, tak dostaneme soustavu kontrolních součtů, které určují hodnotu souřadnic syndromového vektoru.

Příklad 7.9. Sestavíme syndromový vektor pro kontrolní matici $\widehat{H}_3$ z Příkladu 7.8. Vypočítáme syndrom slova $\vec{y}_2 = 01110000$.
Sestavíme kontrolní součty jako součin $\vec{z}\widehat{H}_3$. Dostaneme

$$\vec{z}\widehat{H}_3 = [\, z_1 z_2 z_3 z_4 z_5 z_6 z_7 z_8 \,] \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} =$$

$$= [\, z_4 + z_5 + z_6 + z_7 \quad z_2 + z_3 + z_6 + z_7 \quad z_1 + z_3 + z_5 + z_7 \quad z_1 + z_2 + z_3 + z_4 + z_5 + z_6 + z_7 + z_8 \,]$$

Syndrom $S(\vec{y}_2)$ slova $\vec{y}_2 = 01110000$ je

$$S(\vec{y}_2) = [\, 1+0+0+0 \quad 1+1+0+0 \quad 0+1+0+0 \quad 0+1+1+1+0+0+0+0 \,] = 1011.$$

✓

Cvičení

7.2.1. Ukažte, že kontrolní matice rozšířeného binárního Hammingova kódu $\widehat{\mathrm{Ham}}(r, 2)$ je

$$\widehat{H} = \begin{bmatrix} & & & & 0 \\ & & & & 0 \\ & & H & & \vdots \\ & & & & 0 \\ 1 & 1 & \cdots & 1 & 1 \end{bmatrix},$$

kde H je kontrolní matice binárního Hammingova kódu $\mathrm{Ham}(r, 2)$.

7.3. Hlavní věta o lineárních kódech

Nyní vyslovíme tvrzení na rozhraní teorie kování a lineární algebry. Následující věta ukazuje, že mezi minimální vzdáleností lineárního kódu a mezi pojmem hodnosti generující matice je přímá souvislost.

Věta 7.5. Hlavní věta o lineárních kódech
Mějme lineární $[n, k]$-kód C nad tělesem $GF(q)$. Označme H jeho kontrolní matici. Minimální vzdálenost kódu C je d právě tehdy, když každých $d-1$ sloupců matice H je lineárně nezávislých, ale některých d sloupců matice H je lineárně závislých.

Důkaz. Podle definice kontrolní matice (strana 61) víme, že pro každé kódové slovo $\vec{x} \in C$, kde $\vec{x} = x_1 x_2 \dots x_n$ a $C \subseteq V(n, q)$ platí $\vec{x}H^T = \vec{0}$. Označme $\vec{h}_1, \vec{h}_2, \dots, \vec{h}_n$ sloupce matice H. Potom platí

$$\vec{x}H^T = \vec{0}$$
$$x_1\vec{h_1} + x_2\vec{h}_2 + \dots + x_n\vec{h}_n = \vec{0}. \tag{11}$$

Poslední rovnost můžeme interpretovat jako zápis (lineární) kombinace lineárně závislých sloupců matice H.

Označme libovolné nenulové kódové slovo $\vec{x} \in C$. Slovo $\vec{x}$ váhy w, $w = \mathrm{w}(\vec{x})$, má právě w nenulových souřadnic $x_1, x_2, \dots, x_n$. Z rovnosti (11) vidíme, že příslušná lineární kombinace w sloupců matice H dává nulový vektor, tj. příslušných w vektorů je lineárně závislých.

Nyní podle Věty 4.2. víme, že minimální vzdálenost d lineárního kódu C je rovna nejmenší váze nenulového slova kódu C. Proto existuje d lineárně závislých sloupců matice H.

Na druhou stranu, pokud by existovalo $d-1$ lineárně závislých sloupců matice H, můžeme je označit $\vec{h}_{i_1}, \vec{h}_{i_2}, \dots \vec{h}_{i_{d-1}}$, tak příslušná lineární kombinace $x_{i_1}\vec{h_{i_1}} + x_{i_2}\vec{h}_{i_2} + \cdots + x_{i_{d-1}}\vec{h}_{i_d d-1} = \vec{0}$ říká, že vektor

$$\vec{y} = 0 \dots 0 x_{i_1} 0 \dots 0 x_{i_2} 0 \dots 0 x_{i_{d-1}} 0$$

patří do kódu C a jednalo by se o kódové slovo $\vec{y}$ s váhou menší než d. Takové slovo se však v lineárním kódu nenachází. □

Hlavní věta o lineárních kódech (Věta 7.5.) dává mocný nástroj, jak určit minimální vzdálenost kódu na základě znalosti kontrolní matice bez nutnosti sestavování všech kódových slov nebo dokonce určování vzdálenosti všech dvojic slov.

Příklad 7.10. Ukážeme, že binární Hammingův kód $\mathrm{Ham}(3,2)$ daný kontrolní maticí

$$H_3 = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

má minimální vzdálenost 3.
Ověříme předpoklady Hlavní věty o lineárních kódech (Věty 7.5.). Protože každé dva sloupce kontrolní matice H jsou lineárně nezávislé (u binárního kódu stačí ověřit, že jsou různé), tak minimální vzdálenost d kódu $\mathrm{Ham}(3,2)$ je alespoň 3. Naproti tomu první tři sloupce $\vec{h}_1 = 001$, $\vec{h}_2 = 010$, $\vec{h}_3 = 011$ matice H_3 jsou lineárně závislé, neboť $\vec{h}_3 - \vec{h}_1 - \vec{h}_1 = \vec{0}$. Proto minimální vzdálenost d kódu $\mathrm{Ham}(3,2)$ je nejvýše 3. Celkem dostáváme $\operatorname{dist}\mathrm{Ham}(3,2) = 3$. ✓

Všimněte si, že ověření minimální vzdálenosti v Příkladu 7.10. bylo díky Větě 7.5. jednodušší, než konstrukce všech šestnácti kódových slov a určení jejich váhy. Protože daný kód má minimální vzdálenost 3, stačilo porovnat dvojice sloupců. A pochopitelně nebylo nutno ani určovat vzdálenosti všech dvojic kódových slov, kterých je $\binom{16}{2} = 56$.

Kód s předepsanou minimální vzdáleností

Hlavní věta o lineárních kódech teoreticky dává i možnost konstrukce lineárních kódů s předepsanou minimální vzdáleností. Pro malé hodnoty d není při konstrukci kontrolní matice těžké dodržet, aby každých $d-1$ sloupců kontrolní matice bylo lineárně nezávislých. V následující podkapitole toto pozorování využijeme.

Avšak čím je minimální vzdálenost d kódu vyšší, tím obtížněji budeme hledat takové sloupce kontrolní matice, aby každých $d-1$ sloupců bylo lineárně nezávislých.

Cvičení

7.3.1. Jak pomocí Hlavní věty o lineárních kódech (Věta 7.5.) ukážete, že všechny binární Hammingovy kódy $\mathrm{Ham}(r,2)$*, pro* $r \geq 2$*, mají minimální vzdálenost 3?*

7.4. q-ární Hammingovy kódy

Z Hlavní věty o lineárních kódech (Věty 7.5.) plyne, že aby lineární kód měl minimální vzdálenost 3, tak každé dva sloupce jeho kontrolní matice musí být lineárně nezávislé. To znamená, že žádný sloupce nemůže být nulový a žádný sloupec nemůže být skalárním násobkem jiného sloupce.

Přirozeným úkolem je pro pevně určenou redundanci r najít (lineární) $[n, n-r, 3]$-kód nad tělesem $GF(q)$ pro co největší hodnotu n. Nyní zobecníme Hammingovy kódy pro tělesa řádu q a ukážeme, jak sestavit kontrolní matici takového q-árního Hammingova kódu pro $r = 3$. Bude stačit zvolit co největší množinu sloupcových vektorů takových, že ani jeden sloupec není násobkem jiného sloupce.

Ve vektorovém prostoru $V(r,q)$ má každý nenulový vektor $\vec{v}$ právě $q-1$ nenulových násobků. Taková množina násobků tvoří množinu $q-1$ vektorů $\{k\vec{v} : k \in GF(q),\ k \neq 0\}$, proto všech $q^r - 1$ nenulových vektorů prostoru $V(r,q)$ můžeme rozdělit do $(q^r-1)/(q-1)$ tříd rozkladu (každá o mohutnosti $q-1$) takových, že každé dva vektory z jedné třídy rozkladu jsou násobek jeden druhého a každé dva vektory z různých tříd jsou lineárně nezávislé.

Příklad 7.11. Sestavíme třídy rozkladu násobků vektorů ve vektorovém prostoru $V(2,5)$.
Vektorový prostor $V(2,5)$ obsahuje $5^2 - 1 = 24$ nenulových vektorů, každá třída obsahuje $5-1 = 4$ vektory. Šest tříd rozkladu je

$$\{[0,1],[0,2],[0,3],[0,4]\}, \quad \{[1,0],[2,0],[3,0],[4,0]\}, \quad \{[1,1],[2,2],[3,3],[4,4]\},$$
$$\{[1,2],[2,4],[3,1],[4,3]\}, \quad \{[1,3],[2,1],[3,4],[4,2]\}, \quad \{[1,4],[2,3],[3,2],[4,1]\}.$$

✓

Otázka: Při konstrukci tříd využíváme, že každý nenulový vektor má v $V(n,q)$ právě $q-1$ nenulových násobků. Kterou vlastnost tělesa $GF(q)$ zde využíváme?

Sestavení q-árního Hammingova kódu

Jestliže z každé třídy rozkladu popsaného v předchozích odstavcích vybereme jeden vektor, tak množina vybraných vektorů bude tvořit množinu $(q^r-1)/(q-1)$ lineárně nezávislých vektorů. Takových tříd rozkladu existuje $(q^r-1)/(q-1) = q^{r-1}+q^{r-2}+\cdots+q+1$. Tato množina je jistě největší možná, neboť každý další vektor by patřil do stejné třídy jako nějaký jiný vybraný vektor a byl by jeho násobkem. Jestliže z těchto vektorů sestavíme sloupce matice H, tak dostaneme kontrolní matici lineárního kódu, který nazveme q-árním Hammingovým kódem.

Bez újmy na obecnosti můžeme v každé třídě zvolit vektor, který má první nenulovou souřadnici rovnu 1, což je vždy možné, neboť některý z $q-1$ nenulových násobků takovou hodnotu má.

Definice q-ární Hammingův kód

Lineární kód, jehož kontrolní matice H se skládá z $(q^r-1)/(q-1)$ lineárně nezávislých vektorů prostoru $V(r,q)$ zapsaných do sloupců matice H, nazveme *q-ární Hammingův kód*. Budeme jej značit $\mathrm{Ham}(r,q)$.

Všimněte si, že binární Hammingův kód je speciální případ q-árního Hammingova kódu. Každá třída rozkladu obsahuje jediný prvek, neboť $(q-1) = 2-1 = 1$, a počet tříd rozkladu je $(2^r-1)/(2-1) = 2^r-1$.

Dále si všimněte, že kontrolní matice q-árního Hammingova kódu není určena jednoznačně. Jednak můžeme z každé třídy rozkladu vybrat jiný z $q-1$ násobku vektorů a jednak je můžeme do matice kontrolní matice umístit v libovolném pořadí. Všechny kontrolní matice však můžeme dostat permutací sloupců jedné pevně zvolené kontrolní matice, případně násobením sloupců nenulovými násobky z tělesa F_q.

Příklad 7.12. Sestavte kontrolní matici Hammingova kódu $\mathrm{Ham}(2,5,)$.

Navážeme na Příklad 7.11. Z každé třídy rozkladu vybereme jeden vektor (bez újmy na obecnosti vybereme vždy první vektor zapsaný v dané třídě) a dostaneme tak 6 lineárně nezávislých sloupců kontrolní matice H Hammingova kódu $\mathrm{Ham}(2,5)$.

$$H = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 2 & 3 & 4 \end{bmatrix}$$

✓

Příklad 7.13. Uvedeme další příklady kontrolních matic q-árních Hammingových kódů.

1) Kontrolní matice Hammingova kódu $\mathrm{Ham}(2,3)$ je

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

Jedná s o ternární lineární $[4,2]$-kód, neboť počet sloupců kontrolní matice je $n = (q^r-1)/(q-1) = (3^2-1)/(3-1) = 8/2 = 4$.

2) Kontrolní matice Hammingova kódu $\mathrm{Ham}(2,11)$ je

$$H = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{bmatrix}.$$

Jedná s o jedenáctkový lineární $[12,2]$-kód, neboť $(11^2-1)/(11-1) = 120/10 = 12$.

3) Kontrolní matice Hammingova kódu $\mathrm{Ham}(3,3)$ je

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

Jedná s o ternární lineární $[13,3]$-kód, neboť $(3^3-1)/(3-1) = 13$.

4) Kontrolní matice Hammingova kódu $\mathrm{Ham}(2,6)$ neexistuje, protože neexistuje těleso $GF(6)$.

Nyní můžeme vyslovit následují větu.

Věta 7.6. *Hammingův kód* $\mathrm{Ham}(r,q)$ *je perfektní kód, který umí opravit jednu chybu.*

Důkaz. Kontrolní matice sestavená podle definice je kontrolní matice lineárního $[n, M, 3]$-kódu, pro který platí $n = (q^r - 1)/(q - 1)$ a $M = q^{n-r}$. Protože minimální vzdálenost kódu je $2t + 1 = 3$, tak $t = 1$. Uvedené hodnoty dosadíme do Hammingovy hranice (3) a dostaneme

$$\begin{aligned} M\left(\binom{n}{0} + \binom{n}{1}(q-1) + \binom{n}{2}(q-1)^2 + \cdots + \binom{n}{t}(q-1)^t\right) &= q^{n-r}\left(1 + \binom{n}{1}(q-1)\right) = \\ &= q^{n-r}\left(1 + n(q-1)\right) = \\ &= q^{n-r}\left(1 + \frac{q^r-1}{q-1}(q-1)\right) = \\ &= q^{n-r}\left(1 + q^r - 1\right) = \\ &= q^n, \end{aligned}$$

což je právě pravá strana rovnosti (3) a Hammingův kód $\mathrm{Ham}(r, q)$ je perfektní s minimální vzdáleností 3, který umí opravit jednu chybu. □

Ihned dostaneme následující důsledek.

Důsledek 7.7. *Mějme přirozená čísla r a q, kde $r \geq 2$ a číslo q je mocnina prvočísla. Jestliže $n = (q^r - 1)/(q - 1)$, tak platí $A_q(n, 3) = q^{n-r}$.*

Důkaz. Protože q-ární Hammingův kód $\mathrm{Ham}(r, q)$ je podle Věty 7.1. perfektní kód s minimální vzdáleností 3, tak počet slov kódu je pro dané parametry $n = (q^r - 1)/(q - 1)$ a $d = 3$ největší možný. Počet jeho slov je $M = q^{n-r}$, proto $A_q(n, 3) = q^{n-r}$. □

Dekódování pomocí q-árního Hammingova kódu

Protože každý q-ární Hammingův kód $\mathrm{Ham}(r, q)$ je podle Věty 7.6. perfektní a má minimální vzdálenost 3, tak nenuloví reprezentanti kosetů jsou právě všechny vektory váhy 1 v prostoru $V(n, q)$. Syndrom každého přijatého slova je

- buď nulový vektor, pokud přijaté slovo odpovídá kódovému slovu,
- nebo syndrom některého vektoru váhy 1 z množiny $(q^r - 1)/(q - 1)$ takových vektorů.

Bylo vysláno slovo $\vec{x}$ a přijato slovo $\vec{y}$. Nenulový syndrom $S(\vec{y})$ odpovídá syndromu $\vec{s}_j$ reprezentanta s jedinou nenulovou souřadnicí a můžeme psát

$$\vec{s}_j = S(\vec{y}) = S(0 \ldots 0b0 \ldots 0) = [\,0 \ldots 0b0 \ldots 0\,]\, H^T = b \cdot \vec{h}_j^T,$$

kde $b \in GF(q)$ je jediná nenulová j-tá souřadnice vektoru $\vec{h}_j$ je j-tý sloupec kontrolní matice H.

Uvědomte si, že syndrom *nemusí* odpovídat sloupci kontrolní matice H, ale může být nějakým b násobkem sloupce $\vec{h}_j$. Syndrom však každopádně odpovídá syndromu chybovému vektoru. Proces dekódování tak může probíhat následujícím způsobem. Bude vysláno kódové slovo $\vec{x}$ a přijato bude slovo $\vec{y}$. Určíme syndrom $S(\vec{y}) = \vec{y}H^T = b\vec{h}_j^T$.

Pokud $S(\vec{y}) = \vec{0}$, předpokládáme, že nenastala chyba a kódové slovo $\vec{x}$ odpovídá přijatému slovu $\vec{y}$. Pokud $S(\vec{y}) \neq \vec{0}$, předpokládáme, že nastala jediná chyba na j-té souřadnici, kterou opravíme. Od přijatého slova $\vec{y}$ odečteme chybový vektor $[\,0 \ldots 0b0 \ldots 0\,]$. Dostaneme kódové slovo $\vec{x} = \vec{y} - [\,0 \ldots 0b0 \ldots 0\,]$. Ještě jednodušší je od j-té souřadnice slova $\vec{y}$ odečíst hodnotu b (počítáno nad tělesem $GF(q)$).

Celý postup shrneme v následujícím algoritmu.

Algoritmus 7.8. Dekódování pomocí q-árního Hammingova kódu

Předpokládejme, že máme q-ární Hammingův kód daný kontrolní maticí H, jejíž sloupce tvoří $(q^r - 1)/(q - 1)$ lineárně nezávislých vektorů z vektorového prostoru $V(r, q)$ zvolených tak, aby jejich první nenulová souřadnice byla $1 \in F_q$.

Vysláno bylo kódové slovo $\vec{x}$ a přijato bylo slovo $\vec{y}$.

1) *Vypočítáme syndrom $\vec{z} = S(\vec{y}) = \vec{y}H^T$.*
2) *Pokud $\vec{z} = \vec{0}$, tak předpokládáme, že nenastala žádná chyba a dekódujeme kódové slovo $\vec{x} = \vec{y}$.*
3) *Pokud $S(\vec{y}) \neq \vec{0}$, tak předpokládáme, že nastala právě jedna chyba. Syndrom $S(\vec{y})$ je nějakým nenulovým b násobkem některého sloupce kontrolní matice. Hodnotu b určíme jako hodnotu první nenulové souřadnice syndromu $\vec{z}$ a vypočítáme $h_j = b^{-1}\vec{z}$.*
4) *Určíme index j, jedná se o pořadí sloupce h_j v kontrolní matici H.*

5) Od j-té souřadnice přijatého slova $\vec{y}$ odečteme hodnotu b tj. $x_j = y_j - b$ a $x_i = y_i$ pro $i = 1, 2, \ldots (q^r - 1)/(q-1)$. Dekódujeme slovo $\vec{x}$.

Postup Algoritmu 7.8. ukážeme na následujícím příkladu.

Příklad 7.14. Mějme ternární Hammingův kód Ham$(3, 2)$ z Příkladu 7.13. daný kontrolní maticí

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

Dekódujeme přijatá slova $\vec{y}_1 = 1201$, $\vec{y}_2 = 2220$ a $\vec{y}_3 = 1221$.
Vypočítáme syndromy jednotlivých slov.

1) Syndrom $S(\vec{y}_1) = [\,1201\,]\,H^T = [\,0+2+0+1 \quad 1+0+0+2\,] = 00$. Předpokládáme, že slovo $\vec{y}_1$ bylo přijato bez chyby a dekódujeme jako kódové slovo $\vec{x}_1 = \vec{y}_1 = 1201$.
2) Syndrom $S(\vec{y}_2) = [\,2220\,]\,H^T = [\,0+2+2+0 \quad 2+0+2+0\,] = 11$. Vidíme, že syndrom je stejný jako třetí sloupec kontrolní matice H, proto $S(\vec{y}_2) = 1\vec{h}_3$. Předpokládáme, že slovo $\vec{y}_2$ bylo přijato s jednou chybou 1 v třetím sloupci a dekódujeme $\vec{x}_2 = \vec{y}_2 - [\,0010\,] = 2210$.
3) Syndrom $S(\vec{y}_3) = [\,1221\,]\,H^T = [\,1+2+2+1 \quad 1+0+2+2\,] = 02$. Vidíme, že syndrom násobek prvního sloupce kontrolní matice H, platí $2(0, 1) = (0, 2)$, proto $S(\vec{y}_2) = 2\vec{h}_1$. Předpokládáme, že slovo $\vec{y}_3$ bylo přijato s jednou chybou 2 v prvním sloupci a dekódujeme $\vec{x}_3 = \vec{y}_3 - [\,2000\,] = 2221$.

✓

Kódování pomocí q-árního Hammingova kódu
Definice Hammingova kódu říká, jaký tvar má kontrolní matice. Pro kódování však potřebujeme sestavit generující matici Hammingova kódu, kterou získáme z kontrolní matice postupem dle Věty 6.5. Užitím ekvivalentních úprav můžeme z kontrolní matice Hammingova kódu odvodit příslušnou generující matici. Pokud využijeme pouze řádkové úpravy z Věty 4.3., dostaneme generující matici stejného kódu, pokud však využijeme i sloupcové úpravy, tak zpravidla dostaneme matici jiného ekvivalentního lineárního kódu.

Cvičení

7.4.1. Kolik existuje různých generujících matic q-árního Hammingova kódu Ham(r, q)*?*

7.4.2. Mějme prvočíslo $q = 191$. Dekódujte slovo a) $\vec{y}_1 = 5\,5 \ldots 5$ b) $\vec{y}_2 = a\,a \ldots a$, kde $a \in F_q$ pomocí q-árního Hammingova kódu Ham$(2, 191)$.

7.5. Zkrácení kódu

O zkrácení kódu (ne nutně lineárního!) jsme psali již v podkapitole 2.2. na straně 15. Zkrácení kódu je užitečným nástrojem pro sestavení kódu předepsané délky a minimální vzdálenosti, pokud známe vhodný kód větší délky se *stejnou* minimální vzdáleností.

Definice Mějme q-ární (n, M, d)-kód C. Zvolme libovolnou pevnou souřadnici j a zvolme libovolný pevný symbol λ z abecedy $0, 1, \ldots, q-1$. Vybereme všechna kódová slova, která mají na pozici j symbol λ a tuto souřadnici ze všech vybraných kódových slov vymažeme. Dostaneme $(n-1, M', d')$-kód C', pro který platí

- obecně $M' \leq M$, přičemž ve výjimečných případech může nastat rovnost,
- obecně $d' \geq d$, přičemž zpravidla nastává rovnost.

Kód C' se nazývá *zkrácený kód*.

Anglicky se pro zkrácení kódů používá termín „code puncturing“.

Lemma 7.9. *Mějme nějaký lineární q-ární $[n, k, d]$-kód C. Jestliže vymazaný symbol λ bude symbol 0, tak zkrácený kód C' bude lineární q-ární $[n-1, k-1, d']$-kód C', kde $d' \geq d$.*

Důkaz je ponechán jako Cvičení 7.5.1.

Není těžké si rozmyslet, proč požadujeme, aby vynechaný symbol λ ve zkráceném lineárním kódu byl symbol 0. Pokud minimální vzdálenost q-árního lineárního $[n, k, d]$-kódu C je alespoň $d \geq 2$ (minimální vzdálenost 1 znamená, že kód obecně neumí detekovat ani jednu chybu), tak do lineárního kódu C patří kódové slovo $\vec{0}$ a do C nepatří žádné kódové slovo váhy 1. Vybráním a vynecháním jiného symbolu než 0 tak vznikne zkrácený kód, který neobsahuje kódové slovo $\vec{0}$, a proto nemůže být lineární.

Příklad 7.15. Mějme lineární $[3, 2]$-kód C_2 z Příkladu 4.1. Kód C_2 z Příkladu 1.10. je lineární kód.

$$\text{kód } C_2 = \begin{cases} 000 \\ 011 \\ 101 \\ 110. \end{cases}$$

Vybereme-li libovolnou souřadnici j a vybereme-li pouze kódová slova, která na souřadnici j obsahují symbol 1, tak mezi vybranými slovy *nebude* kódové slovo $\vec{0}$. To znamená, že výsledný kód nebude lineární. Stejné pozorování můžeme udělat pro libovolný lineární kód.

Cvičení

7.5.1. Dokažte Lemma 7.9. Mějme nějaký lineární q-ární $[n, k, d]$-kód C. Sestavíme zkrácený kód tak, že zvolíme libovolnou souřadnici j kódu C, vybereme všechna kódová slova, která na souřadnici j obsahují symbol 0. Dokažte, že výsledný zkrácený kód C' bude lineární q-ární $[n-1, k-1, d]$-kód C', kde $d' \geq d$.

7.5.2. Dokážeme zobecnění Lemmatu 7.9. pro obecné (nejen lineární) kódy. Mějme nějaký q-ární (n, M, d)-kód C. Sestavíme zkrácený kód tak, že zvolíme libovolnou souřadnici j kódu C, vybereme všechna kódová slova, která na souřadnici j obsahují stejný symbol λ. Dokažte, že výsledný zkrácený kód C' bude q-ární $(n-1, M', d']$-kód C', kde $d' \geq d$.

Rejstřík

Kurzívou jsou v rejstříku označeny stránky, kde najdete definici příslušného pojmu.

Literatura

[A] Jiří Adámek, Kódování, SNTL – Nakladatelství technické literatury, Praha, (1989).

[LR] C.C. Lindner, C.A. Rodger, Design Theory, CRC Press, Boca Raton FL, (1997), ISBN 0-8493-3986-3.

[S] N.J.A. Sloane, Error-correcting Codes and Cryptography, Mathematical Gardner (ed. David A. Klarner), California, Wadsworth, (1981), ISBN 978-1-4684-6686-7.

[H] R. Hill, A First Course in Coding Theory, Clarendon Press, Oxford, (2009), ISBN 9780198538035.

Přehled použitých symbolů

$A_q(n, d)$ největší velikost q-árního (n, M, d)-kódu (str. 10)
$\dim(C)$ dimenze prostoru C (str. 39)
$\mathrm{dist}(\vec{u}, \vec{v})$ vzdálenost slov $\vec{u}$ a $\vec{v}$ (str. 6)
$\mathrm{dist}(C)$ minimální vzdálenost kódu C (str. 7)
F_q q-ární abeceda (str. 3)
$\mathrm{Ham}(r, 2)$ Hammingův binární kód (str. 71)