TEORETICKÁ INFORMATIKA 2. ČÁST

Teorie informace a kódování

Václav Vais

Obsah

Předmluva

Tento materiál se snaží být v jistém smyslu uceleným textem o *základech teorie informace a kódování*, psaným sice „z pozic“ teoretické informatiky, ale pro studenty inženýrských, zejména informatických oborů. Text mimo jiné pokrývá dva z pěti tématických okruhů, kterým se věnuje předmět KIV/TI ve druhém ročníku bakalářského studia studijního programu Inženýrská informatika.

Stejně jako v předmluvě k prvnímu dílu musím konstatovat, že t**ext není záznamem přednášek, rozsahem i podrobnostmi značně přesahuje objem, který obvykle bývá k danému tématu odpřednášen (a který také bývá zkoušen).**

Při tvorbě tohoto textu jsem se snažil zejména o následující:

- v teoretických partiích neustoupit z matematické přesnosti, ale prezentované pojmy a postupy doplnit podrobným slovním vysvětlením (tam, kde jsem to považoval za vhodné, jsem někdy přistoupil i k méně přesným, nicméně názornějším formulacím; takové formulace jsou obvykle uvedeny v uvozovkách),
- snažil jsem se ukázat principy a postupy, s jejichž pomocí lze kódování a dekódování realizovat.

U zkoušky bude vyžadována znalost v rozsahu přednášek, tj. v rozsahu zveřejněných přednáškových slajdů, tedy ne v rozsahu tohoto rozšiřujícího textu. Jeho prostudování ovšem může student získat širší nadhled nad přednášenou problematikou, a to i v kontextu reálných aplikací a realizací. Kromě toho by měly „vysvětlující“ části textu napomoci k pochopení těch teoretických partií, které mohou být pro posluchače při dnešní úrovni obecných matematických znalostí a intenzitě návštěv kontaktních výukových akcí obtížnější.

Václav Vais
listopad 2018

1. Úvod do teorie informace

1.1. Historický exkurs

Teorie informace je obor, který se z kvantitativního (tj. nikoli z technologického) hlediska zabývá měřením, kódováním, přenosem, ukládáním a následným zpracováním informací. První publikace na téma *informace* se začínají objevovat od 20. let dvacátého století. Zpočátku byla ovšem informace vnímána pouze jako psychofyziologický jev.

Takový náhled na informaci odmítl Léon Brillouin (1889 – 1969), který definoval informaci jako objektivní obsah komunikace mezi souvisejícími hmotnými objekty, projevující se změnou stavu těchto objektů. Podle Brillouina už informace není psychologický vjem či pocit, ale je to (objektivní) forma komunikace mezi objekty, které spolu mají určitou podobnost (přinejmenším v tom smyslu, že jsou spolu schopny komunikovat) a dochází u nich k objektivním měřitelným změnám, jež lze vyhodnocovat.

Brillouinův přístup dále posunul Norbert Wiener (1894 – 1964) který v knize *Kybernetika a společnost* popsal informaci jako název pro obsah toho, co si vyměňujeme s vnějším světem, když se mu přizpůsobujeme a působíme na něj svým přizpůsobováním.

Podle Toma Stoniera (1927 - 1999) informace existuje ve fyzikální podobě jako součást vnitřní struktury objektů. Projevem této informace je podle Stoniera organizovanost objektu. Tento přístup teorii informaci v jistém smyslu přibližuje termodynamice. Ta pracuje s pojmem *entropie*, kterou chápe jako veličinu udávající „míru neuspořádanosti“ (míru neurčitosti) systému. Později uvidíme, že *entropie* je klíčovou veličinou také v teorii informace a má v ní podobný význam.

Za zakladatele matematické teorie informace je pokládán Claude Elwood Shannon (1916 – 2002). Jeho prvotní motivací bylo zjištění limitů soudobých komunikačních kanálů (telefon, telegraf, televize) s cílem hledání cest k jejich optimálnímu využití. Proto začal matematicky popisovat vztahy mezi dobou přenosu, šířkou frekvenčního pásma, šumem a množstvím přenesené informace.

Z obecného hlediska ovšem Shannon pojal informaci jako veličinu nezávislou na jejím hmotném nosiči a oddělil ji od sémantického obsahu. Následně pak řešil, jak informaci optimálně kódovat, uchovávat a přenášet. Zprávu (sdělení) budeme v Shannonově pojetí chápat jako posloupnost znaků, které odesílatel volí z určitého předem daného souboru znaků. Předpokladem je, že jednotlivým znakům souboru jsou přiřazeny významy, které jsou známy odesílateli i příjemci. Informace je tedy vymezena ve vztahu k jiné, přesně definované veličině (např. stavu, zvuku, obrazu, teplotě, …..) jako míra schopnosti o této veličině vypovídat, tedy snížit nejistotu (neurčitost), kterou o hodnotě této veličiny má příjemce informace. Zjednodušeně řečeno – informace je poznatek, který zmenšuje nebo odstraňuje nejistotu týkající se výskytu určitého jevu ze známé množiny možných jevů.

Aby bylo možné informace přenášet sdělovacím kanálem nebo ukládat na nějaké paměťové médium, musí být informace nějakým způsobem zakódována, tedy přizpůsobena možnostem tohoto sdělovacího kanálu či média. Kódování kromě toho může mít i řadu dalších funkcí. Může sloužit ke kompresi dat, může dokonce umožňovat opravy přenosových chyb. Za specifický typ kódování lze považovat i šifrování dat.

1.2. Entropie a informace

Motivační příklad:

Máme tři osudí (jinými slovy – tři náhodné veličiny). V každém z nich je 10 lístků – některé z nich jsou označeny symbolem „A“, ostatní symbolem „B“ (jinými slovy – u náhodné veličiny může nastat jedna ze dvou realizací – „A“ nebo „B“).

V osudí 1 je	5 lístků s písmenem „A“	a	5 lístků s písmenem „B“
V osudí 2 je	9 lístků s písmenem „A“	a	1 lístek s písmenem „B“
V osudí 3 je	10 lístků s písmenem „A“	a	0 lístků s písmenem „B“

Z každého osudí vytáhneme jeden lístek. Pokusme se intuitivně porovnávat „míru překvapení“ z výsledku každého z tahů. Je zřejmé, že v případě osudí 1 budeme z výsledku „A“ i „B“ „překvapeni stejně“. V případě osudí 2 nás výsledek „A“ příliš nepřekvapí (víceméně jsme ho očekávali), o to více budeme překvapeni z výsledku „B“ (vždyť tento výsledek měl šanci jen 10%). Z výsledku tahu z osudí 3 nebudeme překvapeni vůbec (nic jiného než „A“ nebylo možné vytáhnout).

Je zřejmé, že „míra překvapení“ z konkrétního výsledku losování souvisí s pravděpodobností, s jakou tento výsledek může nastat – čím menší pravděpodobnost, tím větší překvapení a naopak. Kvantifikovat velikost překvapení v případě nemožného výsledku nemá smysl (ledaže bychom je prohlásili za „nekonečně velké“). Tuto zatím intuitivně chápanou „míru překvapení z konkrétního výsledku“ zavedeme jako seriózní veličinu - *elementární entropii písmena* x_i *(elementární neurčitost písmena* x_i*)*, o které si později ukážeme, že souvisí s kvantifikací informace.

Elementární entropie písmene $H(x_i)$ bude zřejmě funkcí pravděpodobnosti tohoto písmene, tedy

$$H(x_i) = f\big(p(x_i)\big)$$

a bude funkcí klesající (čím větší pravděpodobnost, tím menší neurčitost), tedy

$$p_1 < p_2 \Rightarrow f(p_1) > f(p_2)\,.$$

Uvědomme si, že v případě nezávislých jevů musí být neurčitost aditivní, musí tedy platit

$$f(p_1 \cdot p_2) = f(p_1) + f(p_2)\,,$$

protože pravděpodobnost toho, že dva nezávislé jevy nastanou současně, je rovna součinu jejich pravděpodobností.

Výše uvedeným podmínkám vyhovuje funkce $f(x) = -\log(x)$ při libovolném základu větším než 1.

Elementární entropii $H(x_i)$ písmena x_i definujeme jako

$$H(x_i) = -\log_2 p(x_i)$$

Jednotkou entropie je 1 bit.

Poznámka 1: Je třeba rozlišovat 1 bit jako jednotku entropie („teoretický bit“) a bit jako elementární paměťový prvek v paměti počítače („technický bit“). Později uvidíme, že 1 („teoretický“) bit bude i jednotku informace. Někteří autoři místo jednotky 1 bit používají jednotku 1 Shannon. Její význam je stejný, 1 bit = 1 Shannon, ale nový název se zatím příliš neujímá.

Poznámka 2: Někteří autoři definovali entropii pomocí logaritmu s jiným základem, a tedy zavedli i jiné jednotky entropie, nicméně ani tyto pokusy se neujaly. Zmínit můžeme 1 nat - jednotku informace, pokud byl v definici použit přirozený logaritmus $\ln x$ a 1 Hartley (byl-li použit dekadický logaritmus $\log_{10} x$).

Poznámka 3 (pro ty, kteří na kalkulačce nemají funkci logaritmu s obecným základem): K výpočtu lze použít vztah dřívějším generacím známý ze střední školy:

$$\log_2 x = \frac{\log_{10} x}{\log_{10} 2}$$

Elementární entropie se vztahuje k jednomu konkrétnímu písmenu, pravděpodobnosti ostatních písmen její velikost neovlivňují. Vraťme se nyní k výše uvedenému motivačnímu příkladu a zkusme intuitivně porovnat, „jak zajímavé“ může být dlouhodobé sledování losování z osudí 1, 2 a 3.

Je zjevné, že losování z osudí 3 je naprosto nezajímavé, protože nemůže být vylosováno nic jiného než lístek s písmenem „A“. Naopak losování z osudí 1 je nejzajímavější ze všech, protože žádnému výsledku nemůžeme dávat větší šanci, oba mají stejnou pravděpodobnost 0,5. Losování z osudí 2 bude méně zajímavé, protože očekávaným výsledkem bude vytažení lístku se symbolem „A“. Lístek „B“ sice může být také vytažen, ale pravděpodobnost tohoto jevu je jen 0,1, tedy dost malá. Tuto zatím jen intuitivně chápanou „míru zajímavosti ze sledování losování z konkrétního osudí“ bude reprezentovat veličina $H(X)$ - *střední entropie (střední neurčitost*), která bude střední hodnotou elementárních entropií všech písmen. Formálně:

Střední entropii $H(X)$ diskrétní náhodné veličiny X definujeme jako

$$H(X) = -\sum_{i=1}^{r} p(x_i) \log_2 p(x_i)$$

kde r představuje počet realizací náhodné veličiny X .

Jednotkou střední entropie je 1 bit.

Poznámka: V případě, že má některé z písmen nulovou pravděpodobnost, nemá výraz $\log_2 p(x_i)$ uvnitř sumy smysl. V tom případě člen $p(x_i) \cdot \log_2 p(x_i)$ nahradíme jeho limitou k nule zprava, tento člen součtu tedy bude nulový („náhrada“ $0 . \log_2 0 \approx 0$ je ovšem korektní pouze pro účely naší definice):

$$p(x_i) = 0 \Rightarrow p(x_i) \cdot \log_2 p(x_i) \approx \lim_{x \to 0+} (x \cdot \log_2 x) = 0$$

Návrat k motivačnímu příkladu:

Osudí 1: $p(x_1) = 0{,}5, \; p(x_2) = 0{,}5$
$H(X) = -(0{,}5. \log_2 0{,}5 + \; 0{,}5. \log_2 0{,}5) = -\; \log_2 0{,}5 = \; -(-1) = 1 \; [bit]$

Osudí 2: $p(x_1) = 0{,}9, \; p(x_2) = 0{,}1$
$H(X) = -(0{,}9. \log_2 0{,}9 + \; 0{,}1. \log_2 0{,}1) = -[0{,}9. (-0{,}152) + 0{,}1. (-3{,}322)] =$
$= -(-0{,}137 - 0{,}332) = 0{,}469 \; [bit]$

Osudí 3: $p(x_1) = 1, \; p(x_2) = 0$
$H(X) = -(1 \; \log_2 1 + \; 0. \log_2 0) = -(0 + 0) = \; 0 \; [bit]$

(Konec motivačního příkladu).

Velikost střední entropie je omezena, tj. střední entropie nemůže nabývat libovolných hodnot. Platí nerovnost

$$0 \; \leq H(X) \leq \; \log_2 r \qquad [bit]$$

kde r představuje počet realizací náhodné veličiny, tj. počet písmen v abecedě zdroje.

Mezních hodnot dosahuje střední entropie ve speciálních případech:

$H(X) = 0$ právě tehdy, když může nastat pouze jedna realizace (může být vygenerováno pouze jedno písmeno). Pak právě pro toto písmeno platí $p(x_i) = 1$, pravděpodobnosti všech ostatních písmen je nulová. Pro všechna písmena s nulovou pravděpodobností tí $p(x_i) \cdot \log_2 p(x_i) \approx \lim_{x \to 0+} x \cdot \log_2 x = 0$, pro realizaci s pravděpodobností $p(x_i) = 1$ pak je $\log_2 1 \; = 0$, takže všechny členy v sumě z definičního vztahu pro $H(X)$ jsou nulové.

$H(X) = \log_2 r$ právě tehdy, když mají všechna písmena stejnou pravděpodobnost $p(x_i) = \frac{1}{r} \quad \forall \, i = 1, \dots, r.$ Pak $H(X) = \; -\sum_{i=1}^{r} \frac{1}{r} \log_2 \frac{1}{r} = \; -\log_2 \frac{1}{r} = \; \log_2 r$

Všechny úvahy v této kapitole pracovaly s pojmem entropie (neurčitost). Informaci budeme vnímat jako veličinu, která zmenšuje (v ideálním případě odstraňuje) neurčitost. Přesněji: velikost informace, kterou nám přinesl náhodný jev, jenž nastal (např. konkrétní výsledek

nějakého experimentu, písmeno vygenerované zdrojem informace, …..), je roven rozdílu neurčitosti ve sledované veličině **před** tím, než jev nastal **a po** tom, co jev nastal.

U osudí z motivačního příkladu (i u jiných zdrojů informace) má smysl hledat odpovědi na dvě otázky:

a) (po vylosování písmene) Kolik informace **přineslo** písmeno, které **bylo** vytaženo?
b) (před vylosováním písmene) Kolik informace **může přinést** písmeno, které **bude** vytaženo?

V situaci ad a) **známe** písmeno x_i, které bylo taženo, neurčitost už je tedy nulová (nenulová neurčitost je až u písmena, které bude taženo příště). Neurčitost v písmenu x_i **před** jeho vylosováním byla rovna jeho elementární neurčitosti $H(x_i)$. Velikost informace, kterou nám písmeno x_i přineslo, tj. $I(x_i) = H(x_i) - 0 = H(x_i)$ nazveme elementární informací.

Elementární informaci $I(x_i)$ písmena x_i definujeme jako

$$I(x_i) = H(x_i) = -\log_2 p(x_i)$$

Jednotkou elementární informace je 1 bit.

V situaci ad b) nevíme, jaké písmeno bude vygenerováno. Nebudeme-li z nějakého důvodu předjímat, jaké písmeno by **mělo být** taženo, můžeme velikost „očekávané" informace vyjádřit pouze jako střední hodnotu elementárních informací přes všechna písmena abecedy. Neurčitost v taženém písmenu je tedy rovna střední entropii $H(X)$.

Střední informaci $I(X)$ připadající na jedno písmeno definujeme jako

$$I(X) = H(X) = -\sum_{i=1}^{r} p(x_i) \log_2 p(x_i)$$

Jednotkou střední informace je 1 bit.

V tomto případě odpověď na otázku „Kolik informace může přinést písmeno, které bude vytaženo?" bude znít „Ve střední hodnotě jedno písmeno přinese informaci o velikosti $I(X)$".

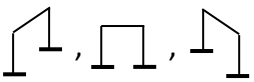
V situaci ad b) někdy může být použít i jiný přístup, kdy otázku trochu pozměníme: „Kolik informace může **maximálně** přinést písmeno, které bude vylosováno?" Maximální informaci přinese písmeno s nejmenší pravděpodobností výskytu, tedy s nejvyšší elementární entropií. Budeme tedy předjímat, že bude taženo právě takové písmeno, a pro toto písmeno vypočítáme elementární informaci $I(x_i)$. Odpověď v tomto případě bude znít „Jedno písmeno může přinést **maximálně** informaci o velikosti $I(x_i)$", kde x_i je písmeno s nejmenším pravděpodobností výskytu.

Odpověď na otázku b) může být klíčová při rozhodování, například při volbě vhodného experimentu, kterým se snažíme potvrdit či vyloučit nějakou hypotézu nebo dospět k jedinému správnému řešení z množiny možných řešení nějaké úlohy. Kdy v takových případech máme při predikci pracovat se střední informací $I(X)$, kdy s maximální elementární informací $I(x_i)$?

- střední informaci $I(X)$ použijeme k predikci tehdy, je-li naše strategie taková, že se „**ve střední hodnotě**“ snažíme „minimalizovat náklady“ nebo „ve střední hodnotě maximalizovat zisk“
- elementární informaci $I(x_i)$ nejméně pravděpodobného písmena použijeme k predikci, pokud je naší strategií maximální riziko; tuto strategii lze popsat tak, že „volíme cestu, která může vést k maximálnímu možnému zisku“ (a přitom nám nevadí, že pravděpodobnost, že takového zisku dosáhneme, je minimální)

Motivační příklad – strategie výběru experimentu:

Zadání: Máme dvanáct okem nerozlišitelných stejných mincí. Jedna z nich je falešná, od pravých se liší pouze hmotností. Není ale známo, zda je lehčí nebo těžší než pravé mince.

K dispozici máme rovnoramenné váhy (dokáží rozlišit tři stavy : ).

Úkoly:

- Kolik vážení potřebujeme k tomu, abychom zjistili odpovědi na dvě otázky
  - Která mince je falešná?
  - Je falešná mince lehčí, nebo těžší než pravé mince?
- Dokážeme to s takovým počtem vážení vždycky?

Řešení:

Nejprve vyhodnotíme neurčitost $H(X)$, která je v úloze. Pokud si mince očíslujeme od 1 do 12, může být řešením například tvrzení "falešná je mince č. 5 a je lehčí než pravá". Možných odpovědí zřejmě existuje 24, tj. dvanáct mincí krát dvě možnosti (lehčí, těžší). Není důvod předjímat, že některá z odpovědí je pravděpodobnější než ostatní, proto rozložení pravděpodobností možných odpovědí považujeme za rovnoměrné a neurčitost v úloze tedy je $H(X) = \log_2 24 = 4{,}585$ bitů (hledáme jednu z 24 možností).

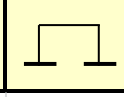
Informaci nám může poskytovat pouze vážení na rovnoramenných vahách. Jaké množství informace nám může poskytnout jedno vážení? Mohou nastat tři různé výsledky vážení: rovnováha, vychýlení vah vpravo, vychýlení vah vlevo. Pokud postavíme vážení tak, že pravděpodobnost všech tří možných výsledků bude stejná, dává nám toto vážení (ve střední hodnotě) maximální informaci, tj $I(V) = \log_2 3 = 1{,}585$ bitů. Je zřejmé, že smysl mají pouze taková vážení, kde je na obou miskách vah stejný počet mincí. Pokud by tomu tak nebylo a na levé misce bylo více mincí než na pravé, jsou pravděpodobnosti tří možných výsledků (1, 0, 0), což dává střední entropii $H(X) = 0$ a znamená to, že vážení poskytne nulovou informaci.

Předpokládejme, že se každé měření povede navrhnout tak, aby při jakémkoli výsledku poskytlo informaci $I(V) = \log_2 3$. Pro hledaný počet vážení n pak musí platit

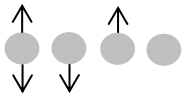
$$n \cdot I(V) \geq H(X), \text{ tedy } n \geq \frac{\log_2 24}{\log_2 3} = \frac{4{,}585}{1{,}585} = 2{,}893$$

Počet potřebných vážení by tedy **mohl být** 3, ale (POZOR!) pouze za předpokladu, že se nám podaří jednotlivá vážení navrhnout tak, aby v součtu poskytla požadovanou informaci o velikosti $\log_2 24 = 4{,}585$ bitů. Je vidět, že požadavek na to, aby ve všech váženích měly všechny tři možné výsledky stejnou pravděpodobnost 1/3, je v našem případě zbytečně přísný. Tři ideálně postavená vážení by poskytla celkovou informaci o velikosti $3 \cdot \log_2 3 = 4{,}755$ bitu, neurčitost v úloze je $\log_2 24 = 4{,}585$ bitů, existuje tedy „rezerva na nerovnoměrnost" o velikosti rozdílu $3 \cdot \log_2 3 - \log_2 24 = 4{,}755 - 4{,}585 = 0{,}17$ bitu, nicméně je zřejmé, že na to, aby stačila tři vážení, se musí rozložení pravděpodobností navrhovaných vážení co nejvíce blížit rozložení rovnoměrnému a v tuto chvíli stále ještě nemůžeme předjímat, zda se všechna vážení povede postavit takovým způsobem. Jediným způsobem, jak rozhodnout, zda ohadnutý počet 3 vážení stačí k nalezení řešení při libovolné „konstelaci" je detailní analýza procesu, který bude navrhovat jednotlivá vážení.

Stanovení prvního vážení:
Jak postavit první vážení? Prozkoumejme systematicky všechny možnosti, které mají na obou miskách vah stejný počet mincí. Nastane-li na vahách rovnováha, znamená to, že se falešná mince nezúčastnila vážení, pravděpodobnost rovnováhy je tedy úměrná počtu mincí, které zůstanou mimo vážení. Pravděpodobnosti výchylky vah na jednu či druhou stranu jsou stejné (v tuto chvíli ještě nemáme důvod cokoli předpokládat o umístění ani o váze falešné mince).
Všechny možnosti připadající do úvahy pro výběr prvního vážení popisuje tabulka:

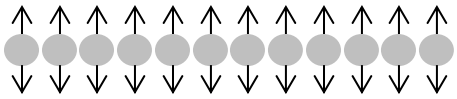
Počet mincí		**P-sti možných výsledků**			**H(X)**
vlevo	**vpravo**	1 	2 	3 	
0	0	0/12	12/12	0/12	0,000
1	1	1/12	10/12	1/12	0,817
2	2	2/12	8/12	2/12	1,252
3	3	3/12	6/12	3/12	1,500
4	4	4/12	4/12	4/12	1,585
5	5	5/12	2/12	5/12	1,483
6	6	6/12	0/12	6/12	1,000

Chceme-li dokázat, že tři vážení stačí v každém případě, budeme vybírat vždy vážení s maximální střední entropií, v našem případě tedy 4 mince na každé misce vah.

„Stav řešení úlohy" budeme znázorňovat obrázky, které ukazují, kolik mincí je „podezřelých" z jaké „odlišnosti". Šipka nahoru ukazuje, že mince může (ale nemusí) být lehčí, šipka dolů, že může (ale nemusí) být těžší. Po celou dobu dále popisovaného postupu musí být o každé minci zřejmé, do které ze čtyř kategorií () na základě dosud získané informace patří. Pokud bychom mince různých kategorií během řešení úlohy pomíchali, připravili bychom se tak o informaci získanou dosud provedenými váženími.

Podívejme se na neurčitost v úloze **před** provedením prvního vážení a **po** prvním vážení, tedy s informací, kterou nám přinesla znalost jeho výsledku.

Neurčitost před prvním vážením:



$H(X) = \log_2 24$ (tj. $H(X)$ před experimentem)

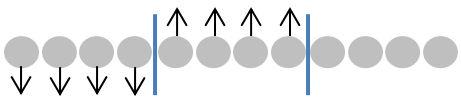
Neurčitost po prvním vážení po výsledku 2 ⊓⊓**:**

$H(X) = \log_2 8 = 3$ (tj. $H(X)$ po experimentu)

Vyjádřeme neurčitost v úloze **po** znalosti výsledku 2 prvního vážení jako neurčitost **před** experimentem zmenšenou o velikost elementární informace, kterou přinesl **konkrétní výsledek** experimentu:

$$H(X)\text{po} = H(X)\text{před} - I(v_2) = \log_2 24 - (-\log_2 \tfrac{4}{12}) = \log_2 24 + \log_2 4 - \log_2 12 =$$
$$= \log_2 \tfrac{24 \cdot 4}{12} = \log_2 8 = 3 \text{ [bit]}.$$

Neurčitost po prvním vážení po výsledku 1 ⊓⊔ **nebo 3** ⊔⊓**:**



$H(X) = \log_2 8 = 3$ (tj. $H(X)$ po experimentu)

Vzhledem k tomu, že pravděpodobnosti $p(v_1), p(v_2), p(v_3)$ všech výsledků prvního vážení byly stejné ($\frac{4}{12} = \frac{1}{3}$), bude stejný i výsledek výpočtu neurčitosti v úloze **po** znalosti výsledku, tj. 3 [bit] (viz výše).

Stanovení druhého vážení po výsledku 2 ⊓⊓**:**

Máme k dispozici 4 mince „podezřelé na obě strany“ a 8 mincí, u kterých je jistota, že jsou pravé. Prozkoumejme systematicky všechny možnosti umístění „podezřelých“ mincí na misky vah s tím, že na misky budeme muset doplňovat i pravé mince, aby na obou stranách byl stejný počet mincí. Pravděpodobnost rovnováhy bude úměrná počtu „podezřelých“ mincí,

které jsou mimo vážení, pravděpodobnosti výchylky vah na jednu či druhou stranu jsou opět stejné. Všechny možnosti připadající v úvahu pro výběr druhého vážení popisuje tabulka:

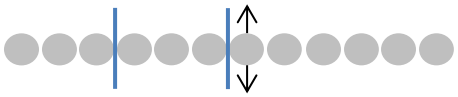
Počet mincí				P-sti možných výsledků			H(X)
vlevo		vpravo		a	b	c	
4	0	0	4	1/2	0/4	1/2	1,000
3	0	1	2	1/2	0/4	1/2	1,000
2	0	2	0	1/2	0/4	1/2	1,000
3	0	0	3	3/8	1/4	3/8	1,561
2	0	1	1	3/8	1/4	3/8	1,561
2	0	0	2	1/4	1/4	1/4	1,500
1	0	1	0	1/4	1/2	1/4	1,500
1	0	0	1	1/8	3/4	1/8	1,061

Je zřejmé, že v tomto případě není možné postavit vážení s rovnoměrným rozložením pravděpodobností výsledků. Ze dvou možností s maximální střední entropií můžeme vybrat libovolnou z nich, my jsme pro další postup zvolili tu označenou.

Neurčitost před druhým vážením po výsledku 2:

$H(X) = \log_2 8$ (tj. $H(X)$ před experimentem)

Neurčitost po druhém vážení (po výsledku b ⊓⊔):



$H(X) = \log_2 2 = 1$ (tj. $H(X)$ po experimentu)

L P MIMO

Neurčitost v úloze **po** znalosti výsledku **b** druhého vážení:

$$H(X)\text{po} = H(X)\text{před} - I(v_b) = \log_2 8 - (-\log_2 \tfrac{1}{4}) = \log_2 8 + \log_2 1 - \log_2 4 =$$
$$= \log_2 \tfrac{8}{4} = \log_2 2 = 1 \text{ [bit]}.$$

Úlohu v tomto případě dořešíme třetím vážením, v němž proti poslední podezřelé minci položíme na druhou misku minci pravou. Podle toho, zda je miska s podezřelou mincí těžší či lehčí, rozhodneme o váze podezřelé mince. Rozložení pravděpodobností při tomto posledním vážení je pak (1/2, 0, 1/2).

Neurčitost v úloze **po** znalosti výsledku třetího vážení (pro oba možné výsledky) v tomto případě:

$$H(X)\text{po} = H(X)\text{před} - I(v_i) = \log_2 2 - (-\log_2 \frac{1}{2}) = \log_2 2 - \log_2 2 = 0$$

Po třetím vážení je neurčitost v úloze nulová, úloha je vyřešena.

Neurčitost po druhém vážení (po výsledku c):

L P MIMO

$H(X) = \log_2 3$ (tj. $H(X)$ po experimentu)

$$H(X)\text{po} = H(X)\text{před} - I(v_c) = \log_2 8 - (-\log_2 \frac{3}{8}) = \log_2 8 + \log_2 3 - \log_2 8 =$$
$$= \log_2 3 \text{ [bit]}$$

Neurčitost po druhém vážení po výsledku **a** bude stejná, výpočet bude analogický.

Úlohu i v těchto případech dořešíme třetím vážením. U všech tří zbývajících „podezřelých mincí“ víme, zda mohou být lehčí nebo těžší. Na váhy proti sobě postavíme dvě ze tří „podezřelých“ mincí. Pokud se váhy vychýlí, prohlásíme za falešnou tu minci, která vyhovuje předpokladu o váze. Pokud se váhy nevychýlí, je falešnou mincí ta, co byla mimo vážení, její váhovou odchylku známe. Rozložení pravděpodobností při posledním vážení je tedy (1/3, 1/3, 1/3).

Neurčitost v úloze **po** znalosti výsledku třetího vážení (pro všechny tři možné výsledky):

$$H(X)\text{po} = H(X)\text{před} - I(v_i) = \log_2 3 - (-\log_2 \frac{1}{3}) = \log_2 3 - \log_2 3 = 0$$

Po třetím vážení je neurčitost v úloze nulová, víme vše, co bylo požadováno, úloha je vyřešena.

Stanovení druhého vážení po výsledku 1 nebo 3 :

Máme k dispozici 4 mince „podezřelé“ z toho, že jsou těžší, 4 mince „podezřelé“ z toho, že jsou lehčí a 4 mince, u kterých je jistota, že jsou pravé. Opět budeme zkoumat všechny možnosti umístění „podezřelých“ mincí na misky vah s tím, že na misky budeme muset doplňovat i pravé mince tak, aby na obou stranách byl stejný počet mincí. Pravděpodobnost rovnováhy bude úměrná počtu „podezřelých“ mincí, které jsou mimo vážení, pravděpodobnosti výchyl-

ky vah na jednu či druhou stranu budou úměrné počtu mincí, které jsou podezřelé z toho, že tento stav mohly navodit. Pravděpodobnosti vychýlení na jednu či druhou stranu už tedy nemusí být stejné. Možností, které připadají v úvahu, je mnoho, popíšeme proto obecný princip, který umožní vytvořit všechny možná vážení a nalezením toho z nich, které má maximální střední entropii pak dojde k nalezení optimálního vážení. Výsledek bude výstupem programátorkého řešení.

Abychom mohli popsat všechna vážení, která připadají v úvahu, zavedeme tuto konvenci: písmenem L budeme označovat počty mincí na levé misce vah, písmenem P počty mincí na pravé misce. Typ mince z hlediska již získané informace pak budeme vyjadřovat indexem – T (podezřelá z toho, že je těžší), L (podezřelá z toho, že je lehčí, nebo N (pravá mince). Na obě strany vah můžeme umisťovat všechny typy mincí, samozřejmě musíme dodržet to, že na obou miskách je stejný celkový počet mincí a že mince nepomícháme. Každé z možných vážení bude reprezentovat šestice $(L_T, L_L, L_N, P_T, P_L, P_N)$, která splňuje následující podmínky:

$$L_T + L_L + L_N = P_T + P_L + P_N \;\; ; \;\; L_T + P_T \leq 4 \; ; \;\; L_L + P_L \leq 4 \;\; ; \;\; L_N + P_N \leq 4$$

Počet různých sestav vážení, které vyhovují uvedeným podmínkám, je větší než 100.

Nyní pro obecné vážení $(L_T, L_L, L_N, P_T, P_L, P_N)$ vyjádříme pravděpodobnosti možných výsledků:

Výsledek  může způsobit těžší mince na levé misce nebo lehčí mince na pravé misce. Pravděpodobnost tohoto výsledku je tedy úměrná součtu $L_T + P_L$.

Výsledek  může způsobit lehčí mince na levé misce nebo těžší mince na pravé misce. Pravděpodobnost tohoto výsledku je tedy úměrná součtu $L_L + P_T$.

Výsledek  může způsobit lehčí mince nebo těžší mince mimo vážení. Pravděpodobnost tohoto výsledku je tedy úměrná rozdílu $8 - (L_T + L_L + P_T + P_L)$.

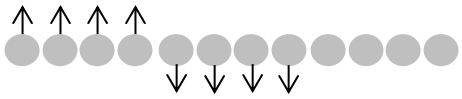
Protože součet všech tří pravděpodobností musí být roven 1, znormujeme tři výše uvedené výrazy jejich celkovým součtem, tedy 8, a dostaneme pravděpodobnosti výsledků:

$$p_{\sqcap} = \frac{L_T + P_L}{8} \quad ; \quad p_{\sqcap\!\sqcap} = \frac{8 - (L_T + L_L + P_T + P_L)}{8} \quad ; \quad p_{\sqcap\!\backslash\!\sqcup} = \frac{L_L + P_T}{8}$$

Výstupem zmíněného programu je 24 různých šestic $(L_T, L_L, L_N, P_T, P_L, P_N)$, které reprezentují vážení s maximální střední entropií o velikosti 1,561 bitů. Pravděpodobnostní rozložení výsledků těchto vážení obsahuje hodnoty 1/4, 3/8, 3/8 (v různém pořadí). Pro další postup vybereme vážení $(0,1,3,1,3,0)$, znázorněné v následující tabulce, nicméně mohli bychom vybrat libovolnou ze zmíněných 24 možností.

Počet mincí						P-sti možných výsledků			H(X)
vlevo			vpravo			d	e	f	
L_T	L_L	L_N	P_T	P_L	P_N				
0	1	3	1	3	0	3/8	3/8	1/4	1,561

Neurčitost před druhým vážením:



$H(X) = \log_2 8$ (tj. $H(X)$ před experimentem)

Neurčitost po druhém vážení (po výsledku e):

$H(X) = \log_2 3$ (tj. $H(X)$ po experimentu)

Neurčitost v úloze **po** znalosti výsledku druhého vážení:

$$H(X)\text{po} = H(X)\text{před} - I(v_e) = \log_2 8 - (-\log_2 \tfrac{3}{8}) = \log_2 8 + \log_2 3 - \log_2 8 = \\ = \log_2 3 \text{ [bit]}$$

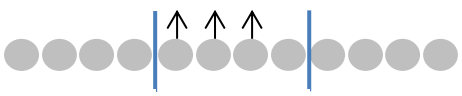
Úlohu v tomto případě dořešíme třetím vážením, v němž proti sobě položíme po jedné z podezřelých mincí. Víme, že falešná mince je těžší. Pokud nastane rovnováha, je falešná mince mimo vážení (a víme, že je těžší). Pokud se váhy vychýlí, je falešná mince ta, která je na nižší misce. Rozložení pravděpodobností při tomto posledním vážení je (1/3, 1/3, 1/3).

Neurčitost v úloze **po** znalosti výsledku třetího vážení (pro všechny možné výsledky):

$$H(X)\text{po} = H(X)\text{před} - I(v_i) = \log_2 3 - (-\log_2 \tfrac{1}{3}) = \log_2 3 - \log_2 3 = 0$$

Po třetím vážení je neurčitost v úloze nulová, úloha je vyřešena.

Neurčitost po druhém vážení (po výsledku d):



$H(X) = \log_2 3$ (tj. $H(X)$ po experimentu)

$$H(X)\text{po} = H(X)\text{před} - I(v_d) = \log_2 8 - (-\log_2 \tfrac{3}{8}) = \log_2 8 + \log_2 3 - \log_2 8 =$$
$$= \log_2 3 \quad \text{[bit]}.$$

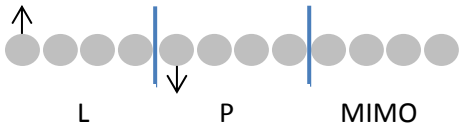
Úlohu i v tomto případu dořešíme třetím vážením. O všech tří zbývajících „podezřelých mincí“ víme, že mohou být jen lehčí. Na váhy proti sobě postavíme dvě ze tří „podezřelých“ mincí. Pokud se váhy vychýlí, prohlásíme za falešnou tu minci, která je na hořejší misce, pokud se váhy nevychýlí, je falešnou mincí ta, co byla mimo vážení a je lehčí. Rozložení pravděpodobností při posledním vážení je tedy (1/3, 1/3, 1/3).

Neurčitost v úloze **po** znalosti výsledku třetího vážení (pro všechny tři možné výsledky):

$$H(X)\text{po} = H(X)\text{před} - I(v_i) = \log_2 3 - (-\log_2 \frac{1}{3}) = \log_2 3 - \log_2 3 = 0$$

Po třetím vážení je neurčitost v úloze nulová, úloha je vyřešena.

Neurčitost po druhém vážení (po výsledku f):



$H(X) = \log_2 2$ (tj. $H(X)$ po experimentu)

$$H(X)\text{po} = H(X)\text{před} - I(v_f) = \log_2 8 - (-\log_2 \tfrac{1}{4}) = \log_2 8 - \log_2 4 = \log_2 \tfrac{8}{4} =$$
$$= \log_2 2 = 1 \quad \text{[bit]}.$$

Úlohu i v tomto případu dořešíme třetím vážením. Na váhy proti sobě postavíme jednu „podezřelou“ minci, třeba tu, která může být lehčí, a proti ní postavíme pravou minci. Pokud se váhy vychýlí, prohlásíme za falešnou tu minci, která je na hořejší misce, pokud se váhy nevychýlí, je falešnou mincí ta poslední podezřelá, co byla mimo vážení a je těžší. Rozložení pravděpodobností při posledním vážení je tedy (0,1/2, 1/2).

Neurčitost v úloze **po** znalosti výsledku třetího vážení (pro oba možné výsledky):

$$H(X)\text{po} = H(X)\text{před} - I(v_i) = \log_2 2 - (-\log_2 \frac{1}{2}) = \log_2 2 - \log_2 2 = 0$$

I v tomto případě je po třetím vážení neurčitost v úloze nulová, víme vše, co bylo požadováno, úloha je vyřešena.

Výše uvedeným postupem jsme dokázali, že k nalezení falešné mince a určení její odchylky stačí tři vážení a stačí ve všech případech. To vše samozřejmě za předpokladu, že během tří vážení maximálně využíváme již získanou informaci, tj, skupiny „stejně podezřelých“ mincí udržíme během celého procesu oddělené a nepomícháme je.

Představme si teď, že bychom měli úkol zadaný jinak: **Najděte odpovědi na obě otázky s menším počtem než tři vážení** ……. například s dodatkem „pokud se vám to podaří, získáte doživotní (a doživotně valorizovanou) rentu“ nebo (jak je běžné v orientálních pohádkách) „pokud se vám to nepodaří, přijdete o hlavu“.

Je zřejmé, že výše uvedený postup, který garantoval, že jsme k odpovědím došli v každém případě po třech váženích, je v tomto případě nepoužitelný. Je vůbec možné dobrat se k odpovědím na méně než tři vážení?

Vraťme se k tabulce, na jejímž základě jsme určili sestavu prvního vážení:

Počet mincí		P-sti možných výsledků			H(X)
vlevo	vpravo	1 	2 	3 	
0	0	0/12	12/12	0/12	0,000
1	1	1/12	10/12	1/12	0,817
2	2	2/12	8/12	2/12	1,252
3	3	3/12	6/12	3/12	1,500
4	4	4/12	4/12	4/12	1,585
5	5	5/12	2/12	5/12	1,483
6	6	6/12	0/12	6/12	1,000

Nyní označený řádek tabulky se vyznačuje tím, že obsahuje **nejmenší nenulovou pravděpodobnost** (1/12) z celé tabulky. Pokud bychom postavili takové vážení a jeho výsledkem by byla nerovnováha, přineslo by nám toto vážení informaci větší, než vážení s rovnoměrným rozložením pravděpodobností.

Neurčitost poté, co při vážení „jedna proti jedné“ nastala nerovnováha ( nebo ):

L(P)P(L) MIMO

$H(X) = \log_2 2$ (tj. $H(X)$ po experimentu)

$I(v_1) = I(v_3) = -\log_2 \frac{1}{12} = \log_2 12 = 3{,}585 \Rightarrow$

$H(X)\text{po} = H(X)\text{před} - I(v_1) = \log_2 24 - (-\log_2 \frac{1}{12}) = \log_2 24 - \log_2 12 =$

$= \log_2 \frac{24}{12} = \log_2 2 = 1$ [bit].

Úlohu v tomto případě dořešíme druhým vážením. Na váhy proti sobě postavíme jednu „podezřelou“ minci, třeba tu, která může být lehčí, a proti ní postavíme pravou minci. Pokud se váhy vychýlí, prohlásíme za falešnou tu minci, která je na hořejší misce, pokud se váhy nevy-

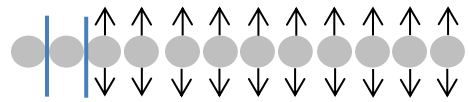
chýlí, je falešnou mincí druhá podezřelá, co byla mimo vážení, a je těžší. Rozložení pravděpodobností při posledním vážení je tedy (0, 1/2, 1/2).

$$H(X)\text{po} = H(X)\text{před} - I(v_i) = \log_2 2 - (-\log_2 \frac{1}{2}) = \log_2 2 - \log_2 2 = 0$$

I v tomto případě je úloha vyřešena a to dokonce po dvou váženích.

Tento postup odpovídá strategii maximálního rizika, která v tomto případě může přinést výsledek již po dvou váženích, ale pravděpodobnost toho, že se to podaří, je omezena výsledkem prvního vážení.

S pravděpodobností 10/12 výsledek prvního vážení nebude takový, „jaký potřebujeme“:



L(P)P(L) MIMO

$H(X) = \log_2 20$ (tj. $H(X)$ po experimentu)

$$I(v_2) = -\log_2 \frac{10}{12} = \log_2 12 - \log_2 10 = 0{,}263 \Rightarrow$$

$$H(X)\text{po} = H(X)\text{před} - I(v_1) = \log_2 24 - (-\log_2 \frac{10}{12}) = \log_2 24 - \log_2 12 + \\ + \log_2 10 = \log_2 \frac{24 \cdot 10}{12} = \log_2 20 = 4{,}322 \text{ [bit]}.$$

Jediná možnost, která v tomto případě ještě dává naději na vyřešení úlohy druhým vážením je taková, že na jednu misku dáme jednu z deseti „podezřelých“ mincí a na druhou jednu ze dvou mincí pravých. Nastane–li nerovnována, máme vyhráno – falešná mince je na konkrétní misce vah a její váhová odchylka je zřejmá. Pravděpodobnost, že nenastane rovnováha, je ale pouze 1/10 (každé z obou vychýlení má p-st 1/20). Pak

$$I(v_i) = -\log_2 \frac{1}{20} = \log_2 20 \Rightarrow$$

$$H(X)\text{po} = H(X)\text{před} - I(v_i) = \log_2 20 - \log_2 20 = 0 \text{ [bit]}.$$

S pravděpodobností 9/10 ovšem nastane rovnováha, takže už o 3 mincích budeme vědět, že jsou pravé, zbylých 9, které se ještě nezúčastnily žádného vážení, bude „oboustranně podezřelých“. Neurčitost v úloze tedy bude $\log_2 18$ [bit].

$$I(v_i) = -\log_2 \frac{9}{10} = \log_2 \frac{10}{9} = 0{,}152 \text{ [bit]} \Rightarrow$$

$$H(X)\text{po} = H(X)\text{před} - I(v_i) = \log_2 20 - (-\log_2 \frac{9}{10}) = \log_2 20 + \log_2 9 - \\ - \log_2 10 = \log_2 \frac{20 \cdot 9}{10} = \log_2 18 = 4{,}170 \text{ [bit]}.$$

V tomto případě jsme tedy dvěma „nevydařenými“ váženími počáteční neurčitost v úloze v hodnotě 4,585 [bit] snížili o „pouhých“ 0,415 [bit].

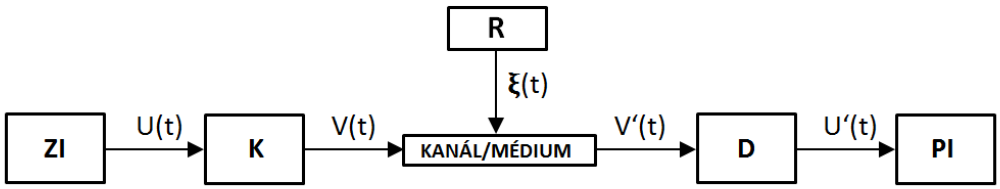
Závěr k modifikovanému zadání „najděte odpovědi na obě otázky s menším počtem než tři vážení“ : Dokázali jsme, že při volbě takové strategie, která bude volit experiment, jehož některý výsledek nabízí nejmenší nenulovou pravděpodobnost, mohou nastat situace, které umožní nalezení odpovědí po dvou váženích. To, že se povede otázky zodpovědět po dvou váženích, ovšem není jisté, pravděpodobnost toho je pouze $\frac{2}{12} + \frac{10}{12} \cdot \frac{1}{10} = 0{,}25$.

(Konec příkladu strategie výběru experimentu)

1.3. Matematický model sdělovací soustavy

Informace může být vyjádřena různou formou (text, obraz, řečový signál, hodnoty nějaké fyzikální veličiny, řídicí povel pro nějaké zařízení, ….). Cílem může být tuto informaci „přenést v prostoru“ (pak mluvíme o přenosu dat) nebo „přenést v čase“ (pak mluvíme o záznamu dat na paměťové médium). Informace proto musí být vhodným způsobem zakódována a reprezentována nějakou fyzikální veličinou, která umožní dálkový přenos nebo zápis do paměťového média. Může to být např. elektrický nebo světelný signál nebo elektromagnetické vlnění (pro přenos dat) nebo elektrický náboj, zmagnetizovaná doména či prohlubeň nebo ploška na odrazné ploše optického disku (pro uchovávání dat). Fyzikální podstata zdroje zpráv i sdělovacího kanálu pro nás ale v tuto chvíli nebude podstatná.

Následující obecné schéma sdělovací soustavy představuje abstraktní model, který vyhovuje jak přenosu dat, tak i procesu zápisu na paměťové médium a následného čtení uložené informace.



Významy jednotlivých bloků ve schématu:

ZI ……. model zdroje informace
K ……. kodér
kanál/médium …….model sdělovacího nebo záznamového prostředí
D ……. dekodér
PI …… příjemce informace
R …… model rušení (nežádoucí vnější působení na sdělovací kanál/záznamové prostředí)
U(t), V(t), V‘(t), U‘(t), ξ(t) …….. matematické modely průběhů příslušných signálů (obecně jsou to *náhodné procesy*)

Pouze v případě nulového rušení $\xi(t)$ platí $V'(t) = V(t)$. Pokud je rušení nenulové, rovnost neplatí. Cílem přenosu je samozřejmě to, aby platilo $U'(t) = U(t)$. Součástí kodéru a dekodéru proto bývají vhodné mechanizmy pro eliminaci (nebo alespoň minimalizaci) důsledků rušení.

Poznámka: *Náhodný proces (stochastický proces)* je matematický pojem, jehož přesná definice přesahuje rámec tohoto materiálu. K pochopení výše uvedeného modelu nám postačí představa, že náhodný proces je zobecněním pojmu náhodná veličina, známého např. z předmětu KMA/PSA nebo (ve speciálním případě) z kapitoly 1.3.1 tohoto textu. Realizací náhodné veličiny je číslo (obecněji prvek z nějaké množiny možných realizací), realizací náhodného procesu je (spojitá nebo diskrétní) funkce jedné proměnné (času). Náhodný proces je tedy funkcí „náhody" a „času". „Zastavíme-li" náhodný proces v čase (tj. zvolíme-li pevnou hodnotu nezávislé proměnné „čas"), získáme tím náhodnou veličinu. Náhodné procesy se obecně popisují nástroji pravděpodobnosti a statistiky, jako jsou např. distribuční nebo autokovarianční funkce. Čas může být chápan spojitě (v tom případě je realizací náhodného procesu spojitá funkce) nebo diskrétně (realizací je pak posloupnost), stejně tak mohou být diskrétní nebo spojité stavy náhodného procesu.

Zdroj informace může být *diskrétní* nebo *spojitý*. Diskrétní zdroj generuje informaci v diskrétních časových okamžicích, zpráva je pak reprezentována řetězcem prvků nad danou množinou. U spojitého zdroje je zpráva reprezentována spojitou funkcí času.

Také sdělovací kanál může být diskrétní nebo spojitý. Diskrétní sdělovací kanál přenáší pouze znaky z nějaké konečné množiny, zatímco spojitý sdělovací kanál je schopen přenášet spojitý signál s charakteristikou v určitém omezeném rozsahu (v případě, že modelujeme metalický spoj, je takovým omezením například frekvenční charakteristika kanálu). Paměťová média jsme dnes zvyklí vnímat jako diskrétní (paměť RAM, pevný disk, DVD, …..), ale nesmíme zapomínat na analogové záznamy na magnetických páskách (nejen zvuk a obraz, ale i záznamy z měřících magnetofonů).

Funkcí kodéru je transformovat zprávy generované zdrojem tak, aby byly přenositelné sdělovacím kanálem. To může představovat operace různého typu podle charakteru zdroje zpráv a přenosového kanálu. Mohou nastat tyto situace:

- diskrétní zdroj informace, diskrétní sdělovací kanál - množina znaků zdroje a množina znaků, které přenáší kanál, jsou obecně různé a nemusí mít stejný počet prvků. Kodér v tomto případě řeší kódování znaků abecedy zdroje do řetězců znaků abecedy kanálu. **Tímto typem kódování se budeme zabývat v kapitolách o kódech**.
- spojitý zdroj informace, spojitý kanál – frekvenční spektrum signálu generovaného zdrojem nemusí odpovídat frekvenčnímu pásmu přenášenému kanálem. Kodér v tomto případě řeší přeložení frekvenčního pásma – provádí spojitou analogovou modulaci (principielně může být amplitudová, frekvenční nebo fázová podle toho, jaký parametr nosného harmonického signálu se signálem zdroje moduluje).
- diskrétní zdroj informace, spojitý sdělovací kanál - kodér v tomto případě řeší modulaci „hranatého" signálu, reprezentujícího posloupnost znaků generovanou zdrojem do frenkvenční oblasti kanálu. Reálným příkladem takového kodéru (i dekodéru) byl modem pro připojení počítače na analogovou telefonní přípojku.

- spojitý zdroj informace – diskrétní sdělovací kanál – kodér v tomto případě řeší vzorkování (v čase) a kvantizaci (v úrovních) spojitého signálu (reálným příkladem takového kodéru je pulzně kódová modulace PCM). Nyquistův (Shannonův, Kotělnikovův) vzorkovací teorém říká, že dokonalá rekonstrukce spojitého signálu je možná pouze tehdy, když je vzorkovací frekvence větší než dvojnásobek maximální frekvence obsažené ve spektru vzorkovaného signálu. Počet úrovní, do kterých lze signál kvantovat, je pak omezen kapacitou sdělovacího kanálu.

  Příklad: PCM pro přenos signálu z analogového telefonního přístroje diskrétním sdělovacím kanálem s přenosovou rychlostí 64 kbit/s .

standardní frekvenční pásmo telefonního signálu	300 Hz – 3400 Hz
maximální frekvence obsažená v signálu	f_{max} = 3,4 kHz
dvojnásobek maximální frekvence	2 x f_{max} = 6,8 kHz
frekvence vzorkování stanovená normou pro PCM	f_{vz} = 8 kHz > 2 x f_{max}
perioda vzorkování	T_{vz} = 125 μs = 1/ f_{vz}

  To znamená, že každých 125 μs bude do sdělovacího kanálu odesílán jeden „zakódovaný vzorek“.

přenosová rychlost kanálu	c = 64 kbit/s
počet bitů, které lze přenést v jedné periodě vzorkování	n = 8 bitů = c/ f_{vz}
počet úrovní, do kterých lze signál kvantizovat	u = 256 = 2^n

  Příklad: Záznam signálu na audio CD podle specifikace Red Book.

vzorkovací frekvence	f_{vz} = 44,1 kHz
počet bitů v zakódovaném vzorku	n = 16 bitů
počet úrovní, do kterých lze signál kvantizovat	u = 65.536 = 2^n

  kapacita média potřebná k záznamu 1 minuty stereo nahrávky:

  c = 2 (kanály) x 60 (s) x 16 (bitů)/ 8 (bitů v bytu) x 44,1 x 10^3 (vzorků/s) =
  = 10.584.000 bytů = 10,584 MB

Funkcí dekodéru je principielně provést inverzní operaci ke kódování kodérem. V některých případech dekodér navíc může i minimalizovat až eliminovat dopady rušení.

V dalším textu v této kapitole se omezíme pouze na pro informatiky nejdůležitější a matematicky nejjednoduššeji popsatelný případ – model přenosu informace z diskrétního zdroje diskrétním sdělovacím kanálem bez paměti.

1.3.1. Model diskrétního zdroje informace

Diskrétním zdrojem informace bez paměti budeme rozumět takový zdroj, kde vysílání jednotlivých znaků tvoří nezávislé jevy. To, jaký znak je vyslán zdrojem v diskrétním časovém okamžiku T_n je statisticky nezávislé na tom, jaké znaky zdroj vyslal v okamžicích T_1 až T_{n-1} .

Takový zdroj informace budeme popisovat pomocí diskrétní náhodné veličiny X. Množinou realizací veličiny bude abeceda zdroje (tedy množina znaků, které zdroj může generovat). Každá realizace (tj. každé písmeno x_i abecedy zdroje) bude mít přiřazenu svoji pravděpodobnost $p(x_i)$, kde $0 \leq p(x_i) \leq 1$, s tím, že součet pravděpodobností přes všechny realizace bude roven 1 (zdroj některé z písmen abecedy zdroje určitě vygeneruje, sjednocením elementárních jevů je tedy jev jistý s pravděpodobností 1). Formálně:

Diskrétní zdroj informace X je popsán diskrétní náhodnou veličinou

$$X = \{x_1, x_2, \ldots., x_r\}\ ,\ \ P(X) = \big(p(x_1), p(x_2), \ldots., p(x_r)\big)\ \ ,\qquad \sum_{i=1}^{r} p(x_i) = 1$$

Elementární entropii $H(x_i)$ písmena x_i definujeme jako

$$H(x_i) = -\log_2 p(x_i)$$

Střední entropii $H(X)$ diskrétního zdroje informace X definujeme jako

$$H(X) = -\sum_{i=1}^{r} p(x_i) \log_2 p(x_i)$$

Elementární informaci $I(x_i)$ připadající na písmeno x_i definujeme jako

$$I(x_i) = H(x_i) = -\log_2 p(x_i)$$

Informační vydatnost $I(X)$ diskrétního zdroje informace X definujeme jako střední informaci připadající na jedno písmeno

$$I(X) = H(X) = -\sum_{i=1}^{r} p(x_i) \log_2 p(x_i)$$

Jednotkou entropie i informace je 1 bit.

Důležitou charakteristikou diskrétního zdroje informace je jeho *redundance (nadbytečnost).* Tato veličina je jednoznačně určena počtem písmen abecedy zdroje a jeho střední entropií.

Redundanci (nadbytečnost) ρ diskrétního zdroje informace X definujeme jako

$$\rho = 1 - \frac{H(X)}{\log_2 r}$$

Redundance je bezrozměrná veličina. Často se udává v procentech. Mezních hodnot nabývá v situacích, kdy dosahuje svých mezních hodnot střední entropie zdroje $H(X)$:

$\rho = 0$ právě tehdy, když má diskrétní náhodná veličina X rovnoměrné rozložení, tj. $H(X) = \log_2 r$. Pak $\rho = 1 - \frac{\log_2 r}{\log_2 r} = 1 - 1 = 0$.

$\rho = 1$ právě tehdy, když je diskrétní náhodná veličina X nabývá pouze jedné hodnoty, je tedy veličinou deterministickou, tj. $H(X) = 0$. Pak $\rho = 1 - \frac{0}{\log_2 r} = 1$.

Ilustrační příklad (redundance):

Máme zdroj informace generující písmena z abecedy $\{0,1\}$, každé z nich s pravděpodobností 0,5. Zpráva bude přenášena nespolehlivým kanálem s abecedou $\{0,1\}$ a pro zvýšení pravděpodobnosti jejího správného vyhodnocení příjemcem bude každý znak „zakódován" trojnásobným opakováním. Jaká je redundance zdroje a jaká je redundance po zakódování?

Před zakódováním: X = $\{0,1\}$, $p(x_1) = 0{,}5$, $p(x_2) = 0{,}5$, $H(X) = 1\ bit$ (viz příklad výše). Pak $\rho = 1 - \frac{1}{\log_2 2} = 1 - \frac{1}{1} = 0$. Nadbytečnost zdroje je tedy nulová.

Po zakódování: Znaky jsou kódovány do trojic. Nad dvouprvkouvou abecedou je takových trojic 8: Y = $\{000, 001, 010, 011, 100, 101, 110, 111\}$. Zakódováním ale můžeme získat pouze dvě trojice, a to 000 s $p(000) = 0{,}5$ a 111 s $p(111) = 0{,}5$. Pravděpodobnosti výskytu všech zbývajících trojic jsou nulové. Tedy $H(X) = 1\ bit$, $\rho = 1 - \frac{1}{\log_2 8} = 1 - \frac{1}{3} = \frac{2}{3}$
Hodnota výsledné redundance je „v souladu se zdravým rozumem", protože na první pohled vidíme, že dva ze tří znaků ve zprávách jsou nadbytečné (z hlediska zakódované informace, nikoli ovšem z pohledu zvýšení pravděpodobnosti správného vyhodnocení zprávy po přenosu nespolehlivým sdělovacím kanálem).

(Konec ilustračního příkladu)

1.3.2. Model diskrétního sdělovacího kanálu

Budeme uvažovat diskrétní stacionární sdělovací kanál bez paměti. Přenosové charakteristiky kanálu jsou konstantní, s časem se nemění (stacionarita). Znak na výstupu kanálu závisí pouze na znaku, jenž byl do kanálu vyslán a nezávisí na dříve přenesených znacích („bezpaměťovost").

Takový diskrétní kanál budeme popisovat pomocí *vstupní abecedy kanálu* , *výstupní abecedy kanálu* a *matice přenosu*. Matice přenosu udává, „jak dobře“ se jednotlivá písmena vstupní abecedy přenášejí kanálem, respektive s jakými pravděpodobnostmi jsou písmena vstupní abcedy vlivem šumu „kanálem přepsána“ na písmena výstupní abecedy. Řádky matice odpovídají písmenům vstupní abecedy, sloupce pak písmenům výstupní abecedy. V pozici (i,j) matice je podmíněná pravděpodobnost $p(y_j/x_i)$, tedy pravděpodobnost toho, že (pokud je do kanálu vysláno písmeno x_i) se na výstupu objeví písmeno y_j . Součet pravděpodobností v libovolném řádku je 1. Formálně:

Diskrétní sdělovací kanál je definován trojicí $K = (X, Y, P)$, kde

$X = \{x_1, x_2, \dots, x_r\}$ ..... vstupní abeceda kanálu
$Y = \{y_1, y_2, \dots, y_s\}$ ..... výstupní abeceda kanálu
$P(Y/X) = [p(y_j/x_i)]$..... matice přenosu kanálu, pro kterou platí

$$\sum_{j=1}^{s} p(y_j/x_i) = 1 \quad \forall\, i = 1, \dots, r$$

Ilustrační příklad: Je dán diskrétní sdělovací kanál

$X = \{1,2,3\}$ ..... vstupní abeceda kanálu
$Y = \{1,2,3,4\}$ ..... výstupní abeceda kanálu

$$P(Y/X) = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0{,}8 & 0{,}0 & 0{,}1 & 0{,}1 \\ 0{,}1 & 0{,}6 & 0{,}2 & 0{,}1 \\ 0{,}0 & 0{,}0 & 1{,}0 & 0{,}0 \end{bmatrix} \end{array}$$

matice přenosu kanálu

Hodnoty v matici přenosu interpretujeme takto:

Bude-li do kanálu na vstupu vložen znak 1, s pravděpodobností 0,8 bude na výstup přenesen správně, s pravděpodobností 0,1 bude v důsledku šumu přenesen na výstup jako znak 3 a se stejnou pravděpodobnosti bude přepsán na znak 4.

Bude-li do kanálu na vstupu vložen znak 2, bude přenesen správně s pravděpodobností pouze 0,6, s pravděpodobností 0,2 bude v důsledku šumu přepsán na znak 3, s prevděpodobnost 0,1 bude přepsán na 1 a se stejnou pravděpodobnosti na znak 4.

Bude-li do kanálu vložen znak 3, bude vždy přenesen bezchybně, na výstupu se vždy objeví znak 3.

Poznámka: Je zřejmé, že matice přenosu ideálního kanálu bez rušení je jednotková matice.

Úlohou „pozorovatele na výstupu kanálu“ je z písmena y na výstupu zpětně určit písmeno x , které bylo do kanálu vloženo. Z matice přenosu je zřejmé, že pouze v případě, že se na výstupu kanálu objeví znak 2, určí pozorovatel vložené písmeno jednoznačně (znaku 2 na výstupu odpovídá druhý sloupec v matici přenosu; ten obsahuje jediný nenulový prvek, což

znamená, že 2 na výstupu mohla „vzniknout“ jediným možným způsobem; nenulový prvek je v řádku odpovídajícím znaku 2 na vstupu, vložen tedy byl znak 2). V případě znaků 1, 3, 4 na výstupu už pozorovatel nemůže rozhodnout jednoznačně, protože tato písmena mohou „vzniknout“ také v důsledku rušení při přenosu.

Je zřejmé, že znaky na výstupu kanálu nesou (nějakou) informaci o znacích, které jsou do kanálu vkládány, v našem případě to ale není informace úplná (ne vždy jsme schopni z písmena na výstupu jednoznačně určit písmeno, které bylo vloženo na vstupu).

(Přerušení ilustračního příkladu)

Budeme-li znát rozložení pravděpodobnosti písmen vstupní abecedy $P(X)$ (tedy popis zdroje, který znaky do kanálu vkládá), budeme moci z tohoto rozložení a z matice přenosu kanálu $P(Y/X)$ spočítat i *aposteriorní rozložení pravděpodobností* vstupní abecedy, tady pravděpodobnosti typu $p(x_i/y_j)$ (tj. pravděpodobnosti, že bylo do kanálu vloženo písmeno x_i za předpokladu, že se na výstupu objevilo písmeno y_j).

Na základě definice podmíněné pravděpodobnosti $p(y_j/x_i) = p(x_i, y_j)/p(x_i)$, kde $p(x_i, y_j)$ je *simultánní pravděpodobnost*, tedy pravděpodobnost, že bude **současně** na vstupu kanálu písmeno x_i a na výstupu písmeno y_j, můžeme spočítat *matici simultánních pravděpodobností* $P(X, Y)$ *:*

$$P(X, Y) = [p(x_i, y_j)] \quad \text{, kde } p(x_i, y_j) = p(x_i) \cdot p(y_j/x_i)$$

Řádkové součty matice $P(X, Y)$ pak odpovídají výchozímu pravděpodobnostnímu rozložení abecedy zdroje $P(X)$, sloupcové součty pak určují rozložení výstupní abecedy kanálu $P(Y)$:

$$\sum_{j=1}^{s} p(x_i, y_j) = p(x_i) \quad \forall i = 1, \dots, r \qquad \sum_{i=1}^{r} p(x_i, y_j) = p(y_j) \quad \forall j = 1, \dots, s$$

Součet všech prvků matice $P(X, Y)$ je roven 1:

$$\sum_{i=1}^{r} \sum_{j=1}^{s} p(x_i, y_j) = 1$$

Nyní již můžeme spočítat i *matici aposteriorních rozložení vstupní abecedy* $P(X/Y)$:

$$P(X/Y) = [p(x_i/y_j)] \quad \text{, kde } p(x_i/y_j) = p(x_i, y_j)/p(y_j) =$$

$$= p(x_i, y_j) \Big/ \sum_{i=1}^{r} p(x_i, y_j)$$

(Návrat k ilustračnímu příkladu)

Zadání rozšíříme o pravděpodobnostní rozložení abecedy zdroje $P(X)$:

$$P(Y/X) = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0{,}8 & 0{,}0 & 0{,}1 & 0{,}1 \\ 0{,}1 & 0{,}6 & 0{,}2 & 0{,}1 \\ 0{,}0 & 0{,}0 & 1{,}0 & 0{,}0 \end{bmatrix} \end{array} \qquad P(X) = \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ (0{,}5\ ,\ 0{,}3\ ,\ 0{,}2) \end{array}$$

Nejprve spočítáme matici simultánních pravděpodobností $P(X,Y)$:

Např. prvek $p(x_1, y_1)$ spočítáme jako $p(x_1, y_1) = p(x_1) \cdot p(y_1/x_1) = 0{,}5 \cdot 0{,}8 = 0{,}4$,
prvek $p(x_2, y_3)$ spočítáme jako $p(x_2, y_3) = p(x_2) \cdot p(y_2/x_3) = 0{,}3 \cdot 0{,}2 = 0{,}06$

Kompletní matice $P(X,Y)$ je pak

$$P(X,Y) = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0{,}4 & 0{,}0 & 0{,}05 & 0{,}05 \\ 0{,}03 & 0{,}18 & 0{,}06 & 0{,}03 \\ 0{,}0 & 0{,}0 & 0{,}2 & 0{,}0 \end{bmatrix} \end{array}$$

Kontrola: $\sum_{i=1}^{r} \sum_{j=1}^{s} p(x_i, y_j) = 1$

Sloupcové součty matice $P(X,Y)$ pak určují pravděpodobnostní rozložení abecedy na výstupu kanálu

$$P(Y) = \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ (0{,}43\ ,\ 0{,}18\ ,\ 0{,}31\ ,\ 0{,}08) \end{array}$$

Kontrola: $\sum_{i=1}^{s} p(y_j) = 1$

Nyní můžeme přistoupit k výpočtu matice aposteriorních pravděpodobností pravděpodobností $P(X/Y)$:

Např. prvek $p(x_1/y_1)$ spočítáme jako $p(x_1/y_1) = p(x_1, y_1)/p(y_1) = 0{,}4\ /0{,}43 = 0{,}93$,
prvek $p(x_2/y_3)$ spočítáme jako $p(x_2/y_3) = p(x_2, y_3)/p(y_3) = 0{,}06\ /0{,}31 = 0{,}19$

Kompletní matice $P(X/Y)$ je pak

$$P(X/Y) = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0{,}93 & 0{,}0 & 0{,}16 & 0{,}63 \\ 0{,}07 & 1{,}0 & 0{,}19 & 0{,}38 \\ 0{,}0 & 0{,}0 & 0{,}65 & 0{,}0 \end{bmatrix} \end{array}$$

Kontrola: 1,00 1,00 1,00 1,00 (sloupcové součty matice $P(X/Y)$)

j-tý sloupec matice $P(X/Y)$ pak představuje aposteriorní rozložení vstupní abecedy kanálu za předpokladu, že byl na výstupu kanálu zpozorován symbol y_j , a umožní tak zpětně odhadnout písmeno, které vstoupilo do kanálu.

(Přerušení ilustračního příkladu)

1.3.3. Přenos informace diskrétním sdělovacím kanálem

V dalším bude naším úkolem kvantifikovat množství informace, které kanál „přenese ze vstupu na výstup“.

Známe-li pravděpodobnostní rozložení abecedy zdroje $P(X)$, můžeme spočítat střední entropii zdroje připadající na jedno písmeno $H(X)$ (viz 1.3.1), která představuje neurčitost ve vstupní abecedě **před** vysláním písmene do sdělovacího kanálu.

Předpokládejme, že se po vyslání znaku x_i do kanálu na jeho výstupu objeví konkrétní písmeno y_j . Jak vyjádříme neurčitost v písmenu x_i na vstupu kanálu **po** objevení y_j na výstupu? Tak, že entropii spočítáme na základě aposteriorního rozdělení pravděpodobností vstupní abecedy, tedy na základě prvku $p(x_i/y_j)$ matice $P(X/Y)$:

Elementární entropie vstupního písmene x_i podmíněná výstupním písmenem y_j je definována jako

$$H(x_i/y_j) = -\log_2 p(x_i/y_j)$$

Její střední hodnotou přes vstupní abecedu X je *podmíněná entropie vstupní abecedy X při známém výstupu y_j* . Hodnota $H(X/y_j)$ se spočítá z *j*-tého sloupce matice $P(X/Y)$. Udává neurčitost, která zbývá ve vstupní abecedě X po objevení konkrétního písmena y_j na výstupu kanálu.

$$H(X/y_j) = -\sum_{i=1}^{r} p(x_i/y_j)\log_2 p(x_i/y_j)$$

Spočítáme-li střední hodnotu podmíněné entropie vstupní abecedy *X* přes všechna písmena výstupní abecedy, získáme *střední podmíněnou entropii vstupní abecedy při známém výstupu,* kterou označujeme jako $H(X/Y)$ *:*

$$H(X/Y) = \sum_{j=1}^{s} p(y_j)\cdot H(X/y_j) = \sum_{j=1}^{s} p(y_j)\cdot\left(-\sum_{i=1}^{r} p(x_i/y_j)\cdot\log_2 p(x_i/y_j)\right) =$$

$$= -\sum_{j=1}^{s}\sum_{i=1}^{r} p(y_j)\cdot p(x_i/y_j)\log_2 p(x_i/y_j) = -\sum_{i=1}^{r}\sum_{j=1}^{s} p(y_j)\cdot p(x_i/y_j)\log_2 p(x_i/y_j) =$$

$$= -\sum_{i=1}^{r}\sum_{j=1}^{s} p(x_i,y_j)\cdot\log_2 p(x_i/y_j)$$

$H(X/Y)$ udává průměrnou neurčitost ve vstupním písmenu při znalosti písmena na výstupu kanálu.

Na základě úvah prezentovaných v kapitole 1.2 budeme informaci, kterou nám přináší výstup kanálu o jeho vstupu, chápat jako rozdíl neurčitostí ve vstupu **před** znalostí výstupu a **po** znalosti výstupu.

Elementární vzájemnou informaci $I(x_i, y_j)$ tedy vyjádříme jako rozdíl elementární entropie písmene x_i vstupní abecedy **před** objevením písmene na výstupu a **po** objevení konkrétního písmene y_j na výstupu, tedy jako

$$I(x_i, y_j) = H(x_i) - H(x_i/y_j)$$

Tato elementární vzájemná informace nám vyjadřuje velikost informace, kterou přináší konkrétní písmeno y_j na výstupu kanálu o tom, že bylo do kanálu vloženo písmeno x_i .

Následující odvození dokazuje, že je tato informace vzájemná, tj. platí $I(x_i, y_j) = I(y_j, x_i)$:

$$I(x_i, y_j) = H(x_i) - H(x_i/y_j) = -\log_2 p(x_i) - (-\log_2 p(x_i/y_j)) =$$

$$= -\log_2 p(x_i) + \log_2 \frac{p(x_i, y_j)}{p(y_j)} = -\log_2 p(x_i) + \log_2 p(x_i, y_j) - \log_2 p(y_j) =$$

$$= -\log_2 p(y_j) + \log_2 \frac{p(x_i, y_j)}{p(x_i)} = -\log_2 p(y_j) - (-\log_2 p(y_j/x_{ji})) =$$

$$= H(y_j) - H(y_j/x_i) = I(y_j, x_i)$$

Písmeno na výstupu kanálu tedy nese o vstupním písmenu stejné množství informace, jako vstupní písmeno o výstupním písmenu.

Spočítáme-li střední hodnotu elementární vzájemné informace $I(x_i, y_j)$ přes X i přes Y, získáme *střední vzájemnou informaci* $I(X, Y)$*:*

$$I(X, Y) = \sum_{i=1}^{r} \sum_{j=1}^{s} p(x_i, y_j) \cdot I(x_i, y_j)$$

Střední vzájemná informace představuje průměrnou hodnotu „užitečné“ informace přenesené ze vstupu na výstup jedním přeneseným písmenem. Protože jsou elementární vzájemná informace i simultánní pravděpodobnost komutativní, tj. $I(x_i, y_j) = I(y_j, x_i)$ a $p(x_i, y_j) = p(y_j, x_i)$, je komutativní i střední vzájemná informace, platí tedy

$$I(X, Y) = I(Y, X) .$$

Střední vzájemnou informaci lze (samozřejmě) také vyjádřit „standardním způsobem“, tedy jako rozdíl středních entropií „**před** a **po**“ :

$$I(X,Y) = H(X) - H(X/Y) \quad \text{, respektive} \quad I(X,Y) = H(Y) - H(Y/X)$$

V kapitole 1.3.1 jsme konstatovali, že jedno písmeno vygenerované zdrojem informace nese (ve střední hodnotě) informaci o velikosti $I(X) = H(X)$. Co se s touto informací stane při přenosu sdělovacím kanálem? Dosadíme-li do prvního ze vztahů v předchozím odstavci $I(X)$ za $H(X)$ a (zatím jen formálně, bez vysvětlení) $I(X/Y)$ za entropii $H(X/Y)$, dostaneme po úpravě vztah

$$I(X) = I(X,Y) + I(X/Y)\,,$$

jenž budeme iterpretovat takto: z informace $I(X)$, kterou nese písmeno vstupující do kanálu, se na výstup přenese pouze její část $I(X,Y)$, část informace o velikosti $I(X/Y)$ se při přenosu ztratí. Tuto informaci $I(X/Y)$ proto nazýváme střední ztracená informace. Tato ztracená informace odpovídá „zbytkové neurčitosti“, tedy neurčitosti, která zůstává ve znalosti vstupu při znalosti výstupu kanálu, tedy právě entropii $H(X/Y)$.

Analogicky můžeme přepsat i druhý vztah jako

$$I(Y) = I(X,Y) + I(Y/X)\,,$$

a konstatovat, že výstupní informace $I(Y)$, kterou nese písmeno na výstupu kanálu, není tvořena pouze „užitečnou přenesenou informací“ $I(X,Y)$, jež vypovídá o vstupu, ale skládá se i z informace o velikosti $I(Y/X)$, která je důsledkem rušení. Tuto informaci $I(Y/X)$ proto nazýváme *střední rušivá informace* a odpovídá neurčitosti, která je na výstupu kanálu při znalosti jeho vstupu.

(Návrat k ilustračnímu příkladu)

Rekapitulace zadání + dosud spočítané veličiny:

$$P(Y/X) = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0{,}8 & 0{,}0 & 0{,}1 & 0{,}1 \\ 0{,}1 & 0{,}6 & 0{,}2 & 0{,}1 \\ 0{,}0 & 0{,}0 & 1{,}0 & 0{,}0 \end{bmatrix} \end{array} \qquad P(X) = \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ (0{,}5\,,\ 0{,}3\ ,\ 0{,}2) \end{array}$$

$$P(X,Y) = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0{,}4 & 0{,}0 & 0{,}05 & 0{,}05 \\ 0{,}03 & 0{,}18 & 0{,}06 & 0{,}03 \\ 0{,}0 & 0{,}0 & 0{,}2 & 0{,}0 \end{bmatrix} \end{array}$$

$$P(Y) = \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ (0{,}43\,,\ 0{,}18\ ,\ 0{,}31\,,\ 0{,}08) \end{array}$$

$$P(X/Y) = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0{,}93 & 0{,}0 & 0{,}16 & 0{,}63 \\ 0{,}07 & 1{,}0 & 0{,}19 & 0{,}38 \\ 0{,}0 & 0{,}0 & 0{,}65 & 0{,}0 \end{bmatrix} \end{array}$$

Z těchto matic a vektorů již můžeme na základě definic pojmů z této kapitoly spočítat všechny výše uvedené střední entropie a informace:

$$I(X) = 1{,}4855 \quad I(Y) = 1{,}7842 \quad I(X,Y) = 0{,}8519 \quad I(X/Y) = 0{,}6335 \quad I(Y/X) = 0{,}9322$$

Platnost vztahů $I(X,Y) = H(X) - H(X/Y)$ a $I(X,Y) = H(Y) - H(Y/X)$ je snadné ověřit.

(Konec ilustračního příkladu)

Souvislost rušivé a ztracené informace s maticí přenosu názorně ukáže následující příklad.

Ilustrační příklad (souvislost ztracené a rušivé informace s maticí přenosu)

Sdělovacími kanály s maticemi přenosu $P_1(Y/X)$ a $P_2(Y/X)$ jsou přenášeny znaky generované zdrojem s rozložením pravděpodobností $P(X)$:

$$P_1(Y/X) = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 1{,}0 & 0{,}0 & 0{,}0 & 0{,}0 \\ 0{,}0 & 1{,}0 & 0{,}0 & 0{,}0 \\ 0{,}0 & 0{,}0 & 0{,}5 & 0{,}5 \end{bmatrix} \end{array} \qquad \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ P(X) = (0{,}5\ ,\ 0{,}25\ ,\ 0{,}25) \end{array}$$

Výpočtem podle výše uvedených vzorců dospějeme k následujícím výsledkům:

$$I(X) = 1{,}5 \quad I(Y) = 1{,}75 \quad I(X,Y) = 1{,}5 \quad I(X/Y) = 0 \quad I(Y/X) = 0{,}25$$

Z těchto hodnot je zřejmé, že je ztracená informace $I(X/Y)$ nulová, veškerá vstupní informace se tedy dostane na výstup. Lze to ukázat i na matici přenosu, z níž je zřejmé, že lze z výstupního písmena vždy jednoznačně určit písmeno vstupní (v každém sloupci je právě jedna nenulová hodnota).

$$P_2(Y/X) = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cc} 1 & 2 \end{array} \\ \begin{bmatrix} 1{,}0 & 0{,}0 \\ 0{,}0 & 1{,}0 \\ 0{,}0 & 1{,}0 \end{bmatrix} \end{array} \qquad \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ P(X) = (0{,}5\ ,\ 0{,}25\ ,\ 0{,}25) \end{array}$$

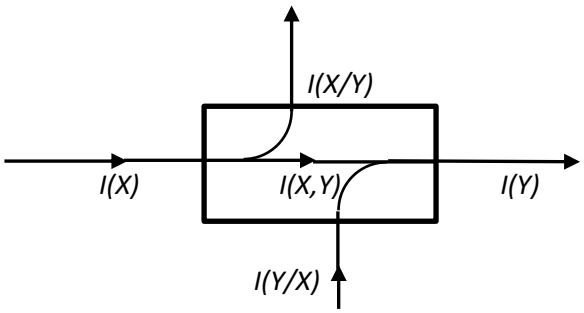
V případě tohoto kanálu výpočtem zjistíme:

$$I(X) = 1{,}5 \quad I(Y) = 1{,}0 \quad I(X,Y) = 1{,}0 \quad I(X/Y) = 0{,}5 \quad I(Y/X) = 0$$

V tomto případě je nulová rušivá informace $I(Y/X)$, což je dáno tím, že lze vždy ze vstupního písmena jednoznačně určit písmeno výstupní (v každém řádku matice přenosu je právě jedna jednička). Část vstupní informace se ovšem ztratí, protože z výstupu nelze jednoznačně určit vstup.

(Konec ilustračního příkladu)

Přenos informace diskrétním sdělovacím kanálem můžeme znázornit jednoduchým schématem:



1.3.4. Kapacita diskrétního sdělovacího kanálu

Přenos informace od zdroje k příjemci, by měl být co nejefektivnější, což lze vyjádřit požadavkem přenést za jednotku času maximální množství informace při minimálním počtu chyb. Zaveďme do našich úvah čas prostřednictvím konstanty τ [sec], která bude představovat čas potřebný k přenesení jednoho písmena sdělovacím kanálem.

Maximální množství informace, které může být kanálem přeneseno za jednotku času, pak lze vyjádřit jako

$$C = \frac{1}{\tau}\max_{P(X)} I(X,Y) \quad = \frac{1}{\tau}\max_{P(X)}[H(Y) - H(Y/X)] \qquad \text{[bit/sec]} ,$$

kde se maximum hledá přes všechna pravděpodobnostní rozložení vstupní abecedy kanálu. Hodnota výrazu v hranatých závorkách totiž závisí jak na parametrech kanálu (tj. na matici přenosu $P(Y/X)$), tak i na pravděpodobnostním rozložení vstupní abecedy kanálu $P(X)$.

Zamysleme se nad obecnou úlohou, jak najít k danému sdělovacímu kanálu takové rozložení vstupní abecedy, které by za jednotku času umožnilo přenést maximální množství informace. V kapitole 1.3.1 jsme ukázali, že (ve střední hodnotě) maximální množství informace vygeneruje zdroj s rovnoměrným rozložením pravděpodobností písmen abecedy. Je takové rozložení vstupní abecedy kanálu řešením naší úlohy? **Obecně nikoli,** protože sdělovací kanál nemusí přenášet všechny znaky stejně kvalitně, může to být *nesymetrický sdělovací kanál*. Pak by zřejmě přenesení většího množství informace umožnilo takové vstupní rozložení, které (velice volně řečeno) „více využívá písmena, jež kanál přenáší lépe, a naopak“. Rozložení vstupní abecedy kanálu lze ovlivnit vhodným kódováním.

Symetrickým kanálem rozumíme kanál, jehož matice přenosu splňuje obě následující podmínky:

- všechny **řádky** matice obsahují stejný soubor hodnot pravděpodobností, řádky se navzájem liší pouze pořadím, v jakém jsou hodnoty seřazeny
- všechny **sloupce** matice obsahují stejný soubor hodnot pravděpodobností, sloupce se navzájem liší pouze pořadím v jakém jsou hodnoty seřazeny

U symetrického kanálu se maximálního množství přenesené informace dosáhne právě při rovnoměrném rozložení pravděpodobností písmen na vstupu kanálu.

Obecné řešení úlohy nalézt optimální rozložení vstupní abecedy pro nesymetrický kanál není jednoduché. Pro nesymetrický kanál, který přenáší pouze dva znaky, lze úlohu poměrně jednoduše vyřešit numericky.

Ilustrační příklad (optimální rozložení vstupních písmen pro nesymetrický binární kanál)

Je dán sdělovací kanál s maticí přenosu P(Y/X). Najděte takové rozložení pravděpodobností vstupních písmen kanálu, které umožní přenos maximálního množství informace $I(X,Y)$.

$$P_1(Y/X) = \begin{matrix} \\ 1 \\ 2 \end{matrix} \begin{matrix} \begin{matrix} 1 & 2 \end{matrix} \\ \begin{bmatrix} 0{,}9 & 0{,}1 \\ 0{,}2 & 0{,}8 \end{bmatrix} \end{matrix} \qquad P(X) = (p(x_1) = p,\ p(x_2) = 1 - p)$$

Na základě již dříve používaných definičních vztahů dospějeme numerickou optimalizací (např. v Excelu) k následujícím výsledkům:

$p_1 = 0{,}51755742 \qquad p_2 = 0{,}48244258$

$I(X) = 0{,}9991 \quad I(Y) = 0{,}9888 \quad I(X,Y) = 0{,}3978 \quad I(X/Y) = 0{,}6014 \quad I(Y/X) = 0{,}5910$

Výsledky „nejsou v rozporu se zdravým rozumem“. Vstupní rozložení je „lehce vychýleno“ ve prospěch znaku 1, což respektuje, že znak 1 je kanálem přenášen „o něco lépe“ než znak 2.

Pokud bychom do stejného kanálu vkládali písmena s rovnoměrným rozložením pravděpodobností, získali bychom tyto výsledky:

$p_1 = 0{,}5 \qquad p_2 = 0{,}5$

$I(X) = 1{,}0000 \quad I(Y) = 0{,}9928 \quad I(X,Y) = 0{,}3973 \quad I(X/Y) = 0{,}6027 \quad I(Y/X) = 0{,}5955$

Z porovnání výsledků při dvou různých vstupních rozloženích je vidět, že rovnoměrné rozložení písmen na vstupu sice do kanálu vloží ve střední hodnotě více informace na jedno písmeno než nalezené optimální rozložení, nicméně je zřejmé, že v důsledku nesymetrické chybovosti kanálu je v tomto případě větší jak ztracená, tak i rušivá informace připadající na jedno písmeno, takže ve výsledku se při rovnoměrném rozložení jedním písmenem v průměru přenese méně užitečné informace ze vstupu, než při nalezeném sice nerovnoměrném, nicméně pro tento kanál optimálním vstupním rozložením.

(Konec ilustračního příkladu)

2. Kódování

2.1. Účel kódování

V této kapitole se budeme zabývat výhradně kódováním znakových řetězců pro přenos diskrétním sdělovacím kanálem, jak již bylo naznačeno v 1.3. Primárním důvodem pro kódování je **přizpůsobení zdrojových řetězců vstupní abecedě kanálu**. Typickým příkladem kódu, který plní tento úkol, je kód ASCII (American Standard Code for Information Interchange), jenž reprezentuje znaky anglické abecedy a řadu dalších (nejen) textových znaků pomocí sedmibitových (v rozšířených variantách osmibitových) značek tvořených znaky 0 a 1. ASCII kód tak (stejně jako řada podobných dříve či později standardizovaných kódů) umožňuje textovou informaci přenášet prostřednictvím binárních sdělovacích kanálů a ukládat v dvoustavových paměťových prvcích počítače.

Kromě zmíněného přizpůsobení zdrojové abecedy sdělovacímu kanálu může kódování souběžně řešit i další úlohy, jako je například efektivnější využití přenosového kanálu či paměťového média nebo zvýšení odolnosti přenášených nebo ukládaných zpráv proti rušení. Za specifický druh kódování pak lze považovat šifrování zpráv.

Teorie kódování je zajímavou aplikací řady matematických disciplin, jako je např. kombinatoríka, lineární algebra, teorie čísel, teorie grup, Galoisova teorie těles, apod.

V následujícím textu se budeme zabývat dvěma skupinami kódů, a to

- kódy pro kanál bez šumu (tj. pro „stoprocentně spolehlivý“ kanál bez rušení)
- kódy pro kanál se šumem (tj. pro sdělovací kanál, který není „stoprocentně spolehlivý“)

Kódy pro kanál bez šumu jsou konstruovány tak, aby byla efektivně využita kapacita sdělovacího kanálu, respektive paměťového média. Takové kódy umožňují bezeztrátovou kompresi dat. Mají smysl tam, kde jsou data produkována zdroji s větší redundancí. Princip takových kódování spočívá ve snižování redundance, takže je v zakódované zprávě nižší redundance než ve zprávě zdrojové. Dekódovací algoritmus pak je schopen zdrojovou zprávu zrekonstruovat, a to vždy, protože (podle předpokladu) je kanál bezšumový. POZOR! – nezaměňovat s kompresemi, jež se používají pro ukládání obrazových a zvukových záznamů, které jsou ztrátové, což znamená, že dekodér není schopen zrekonstruovat vysílanou/ukládanou zprávu „jedna ku jedné“. V případě zmíněných aplikací je ale ztráta určitého objemu informace (smyslovým vnímáním obvykle jen těžko rozpoznatelná) vyvážena výrazným zmenšením objemu komprimovaných dat.

Kódy pro kanál se šumem naopak do zpráv přidávají redundandní informace, které (za určitých pravděpodobnostních předpokladů) dekodéru umožní přenosové chyby detekovat (tj. zjistit, že při přenosu došlo k nějaké chybě - *detekční kódy*) nebo dokonce tuto chybu lokalizovat a opravit (*samoopravné kódy*). V zakódovaných zprávách je tedy větší redundance než ve zprávách zdrojových. Skupina kódů pro kanál se šumem bývá souhrnně nazývána *bezpečnostní kódy* .

2.2. Kódy pro kanál bez šumu

2.2.1. Kódování znaků, jednoznačná dekódovatelnost řetězců

Motivační příklad: Binární kódování dekadických číslic.

$A = \{0,1,2,3,4,5,6,7,8,9\}$ …… abeceda zdroje
$B = \{0,1\}$ …… (vstupní) abeceda kanálu

Kódování každému zdrojovému znaku přiřadí řetězec tvořený znaky z abecedy kanálu.

Několik standardizovaných řešení představuje následující tabulka:

Zdrojový znak	Zakódovaný znak (kódová značka)			
	BCD 8421	**BCD 2421**	**Excess 3**	**POSTNET 74910**
0	0000	0000	0011	11000
1	0001	0001	0100	00011
2	0010	0010	0101	00101
3	0011	0011	0110	00110
4	0100	0100	0111	01001
5	0101	1011	1000	01010
6	0110	1100	1001	01100
7	0111	1101	1010	10001
8	1000	1110	1011	10010
9	1001	1111	1100	10100

„Řešením prvního nápadu“ je samozřejmě kód BCD 8421, který každé dekadické číslici přiřadí její přímou čtyřbitovou binární reprezentaci (bez znaménka). Tento kód je *kód váhový*, rozšiřující část názvu (8421) představuje váhy jednotlivých bitů v kódové značce.

Dekadickou hodnotu zdrojového znaku můžeme u váhového kódu vyjádřit jako $a = \sum_i b_i v_i$. Značka 0111 v BCD 8421 tedy představuje znak, jehož dekadickou hodnotu spočítáme jako $a = b_1 v_1 + b_2 v_2 + b_3 v_3 + b_4 v_4 = 0 \cdot 8 + 1 \cdot 4 + 1 \cdot 2 + 1 \cdot 1 = 7$. Modré znaky představují jednotlivé bity značky, červené jsou váhy jednotlivých pozic váhového kódu.

Kód BCD 2421 je rovněž váhový a je modifikací předchozího kódu. Díky nastavení vah má tu vlastnost, že obrazy komplementárních dekadických cifer (tj. dvojic cifer, jejichž součet je 9) jsou komplementární. Např. znak 3 je zakódován jako 0011. Kódová značka odpovídající znaku 6 (tj. „doplňku“ 3 do 9) proto bude „negací bit po bitu“ kódové značky 0011, tedy 1100.

Kód Excess-3 (někdy označovaný jako BCD + 3) váhový není. Kódová značka se vytvoří tak, že se binárně zakóduje dekadické číslo o 3 větší než kódovaný znak. Např. znak 8 zakódujeme tak, že k 8 přičteme 3 , tedy 8 + 3 = 11 a číslo 11 vyjádříme v přímé čtyřbitové binární reprezentaci jako 1011.

BCD (Binary Coded Decimal) kód (i jeho výše zmiňované modifikace) byl používán zejména na počátku vývoje samočinných počítačů, kdy bylo jejich klíčovým použitím zpraco-

vání finančních dat. Instrukční soubory sálových počítačů, jako byly IBM/360, PDP, VAX ale třeba i mikroprocesorů Motorola 68000 obsahovaly instrukce pro operace s daty ve formátu BCD, který (na rozdíl od binárních reprezentací ve formátu pohyblivé řádové čárky) mimo jiné odstraňoval komplikace vyplývající ze zaokrouhlování desetinných částí.

Kód POSTNET 74910 je příkladem kódu, který lze charakterizovat jako kód „2 z 5“. V každé značce jsou právě dvě jedničky a 3 nuly. Značek s touto vlastností lze vytvořit $\binom{5}{2} = \frac{5!}{2!\cdot 3!} = \frac{5\cdot 4\cdot 3\cdot 2\cdot 1}{2\cdot 1\cdot 3\cdot 2\cdot 1} = \frac{5\cdot 4}{2} = 10$, takže každé dekadické číslici lze přiřadit jedinečnou kódovou značku a naopak každá značka se dvěma jedničkami bude mít přiřazenu svoji zdrojovou dekadickou číslici. Kód „2 z 5“ má délku značky 5, zatímco všechny předchozí kódy měly délku značky 4. Za to ovšem získáváme určitou výhodu - je zřejmé, že při přenosové chybě v jednom bitu značky příjemce pozná, že při přenosu došlo k chybě (přijatá pětice bude mít buď jednu nebo tři jedničky, nesplňuje tedy vlastnost „2 z 5“). V případě kanálu bez šumu je tato vlastnost samozřejmě nadbytečná, nicméně tento fakt nijak nesnižuje to, že i kód „2 z 5“ je řešením našeho zadání.

Výše uvedená tabulka definuje (pro každý ze zmiňovaných kódů) *kódování znaků* , Je zřejmé, že každému zdrojovému znaku musí být přiřazena jedinečná značka. Jak bude prováděno *kódování řetězců*? Tak, že budeme řetězec kódovat „znak po znaku“. Takže např. vstupní řetězec 2591 v kódu BCD 8421 zakódujeme jako 0010010110010001.

Dekódování řetězce bude probíhat tak, že příjemce přijatý řetězec „rozseká“ na čtveřice bitů a ke každé čtveřici přiřadí zdrojový znak podle kódovací tabulky.

Všechny výše zmiňované kódy patří mezi *blokové kódy*, tedy mezi kódy, kde mají všechny kódové značky stejnou délku. To ale není nutnou podmínkou. V zadání příkladu jsme se nedozvěděli nic o pravděpodobnostním rozložení písmen zdrojové abecedy, předpokládali jsme tedy, že je rovnoměrné. Pokud bychom například věděli, že číslice 0 má větší pravděpodobnost výskytu než ostatní číslice a naopak číslice 8 a 9 mají pravděpodobnost výskytu ve zprávách menší, než ostatní, mělo by smysl, aby kódování vypadalo například takto:

Zdrojový znak	**Kódová značka**	
	Kód 1	**Kód 2**
0	0	0
1	0001	1000
2	1001	1001
3	0101	1010
4	1101	1011
5	0011	1100
6	1011	1101
7	0111	1110
8	01111	11110
9	11111	11111

Budeme pracovat s kódem 1 a zakódujeme např. zdrojový řetězec 016058. Výsledkem bude řetězec 00001101100011111110 . Délka tohoto řetězce je 19 znaků. Kdybychom kódovali BCD kódem, měla by zakódovaná zpráva 20 znaků, kódováním s nestejnou délkou značek jsme tedy u této zprávy „ušetřili“ jeden přenášený znak. Při předpokládaném nerov-

noměrném rozložení pravděpodobností písmen ve zprávách tedy bude vhodně zvolený kód s nestejně dlouhými kódovými značkami efektivnější, než blokový kód.

Do potíží se ovšem dostaneme při dekódování řetězce. Protože všechny značky nemají stejnou délku, nemůžeme řetězec 0000110110001111110 mechanicky „rozsekat“ na kódové značky odpovídající jednotlivým zdrojovým znakům a značky pak dekódovat samostatně. Pokusme se řetězec dekódovat intuitivně. Při dekódování se budeme snažit začátek nezpracované části řetězce interpretovat jako kódovou značku s tím, že „priorita značek“ je dána jejich pořadím v kódovací tabulce. Pokud se dostaneme do situace, kdy žádná taková značka neexistuje (červené otazníky), znamená to, že tento pokus k cíli nevedl, musíme tedy zkusit jiný.

```
                0 |0 |0 |0 |1  1  0  1 | 1  0  0  0  1   1  1  1  1   1  0
Pokus 1         0  0  0  0          4   ?????????

                0 |0 |0 |0  1  1  0  1  1  0  0  0  1   1  1  1  1   1  0
Pokus 2         0  0  0   ?????????

                0 |0 |0  0  1  1| 0 |  1   1  0  0  0  1   1  1  1  1   1  0
Pokus 3         0  0           5  0  ?????????

                0 |0  0  0  1 |1  0  1   1 | 0|0 |0 |1   1  1  1  1 | 1  0
Pokus 4         0           1           6   0 0  0             9   ???
                                        .
                                        .
                0 |0  0  0  1 |1  0  1   1 | 0 | 0  0 1  1 | 1  1  1  1 0
Pokus x         0           1           6   0          5             8
```

Je zřejmé, že dekódování řetězce, jehož řešení metodou „pokus – omyl“ jsme naznačili, by v případě algoritmizace vyžadovalo metodu založenou na backtrackingu, nebylo by tedy příliš efektivní.

Zakódujeme stejný zdrojový řetězec 016058 kódem 2. Výsledkem bude řetězec 010001101011000111 1 . Pokusíme se opět o dekódování:

```
                0 |1  0  0  0 |1  1  0   1 | 0 | 1  1  0  0 | 0  1  1  1 1
Pokus 1         0           1           6   0          5             8
```

Vidíme, že při dekódování stejným postupem, jaký jsme použili u kódu 1, jsme se nedostali do situace, kdy by začátek dosud nezpracované části řetězce nebylo možné interpretovat jako kódovou značku, na dekódování nám tedy stačil jediný pokus. Tato jednoznačnost vyplývá z toho, že (na rozdíl od kódu 1) u kódu 2 **žádná kódová značka není začátkem jiné kódové značky**. Kódy s touto důležitou vlastností se budeme více zabývat v kapitole 2.2.2.

(Přerušení motivačního příkladu)

Nyní již můžeme formálněji definovat některé základní pojmy z teorie kódování.

Nechť

$A = \{a_1, a_2, \ldots, a_r\}$ je *abeceda zdroje* (obsahuje *r* prvků)

$B = \{b_1, b_2, \ldots, b_s\}$ je (vstupní) *abeceda kanálu (kódová abeceda)* (obsahuje *s* prvků)

Bez újmy na obecnosti budeme předpokládat $r > s$ (pokud by tomu tak nebylo, kódování není třeba řešit, protože kanál je schopen přenášet přinejmenším tolik znaků, kolik prvků má abeceda zdroje; každý zdrojový znak tedy bude kódován jedním znakem abecedy kanálu).

Kódováním znaků rozumíme prosté (injektivní) zobrazení $K : A \to B^+$, kde B^+ představuje množinu všech neprázdných řetězců vytvořených z prvků abecedy kanálu B. Každý zdrojový znak je tedy jednoznačně zakódován do neprázdného řetězce znaků kódové abecedy.

Blokovým kódováním délky n rozumíme prosté (injektivní) zobrazení $K : A \to B^n$, kde n je stejná délka všech kódových značek a B^n je množina všech řetězců délky n z prvků abecedy B. Každý zdrojový znak je tedy jednoznačně zakódován do n−tice znaků kódové abecedy.

Kódováním řetězců (zpráv) rozumíme zobrazení $K^* : A^* \to B^*$, jež je jednoznačně určeno kódováním znaků K takto

$$K^*(a_1 a_2 \ldots a_l) = K(a_1) \cdot K(a_2) \cdot \ldots \ldots \cdot K(a_l) \quad , \quad K^*(e) = e$$

Kde A^* (B^*) představuje množinu všech řetězců vytvořených z prvků abecedy A (B) včetně prázdného řetězce a operace $\cdot$ představuje zřetězení znakových řetězců.

Injektivnost zobrazení K nestačí k tomu, aby byl kód jednoznačně dekódovatelný.

Podmínka jednoznačné dekódovatelnosti: Kódování řetězců K^* je prosté zobrazení.

Každý blokový kód je jednoznačně dekódovatelný.

Ilustrační příklad (nesplněná podmínka jednoznačné dekódovatelnosti):

Zdrojový znak	Kódová značka
A	0
B	1
C	01

Je zřejmé, že zakódováním řetězce AB vznikne kódový řetězec 01, ovšem stejný řetězec vznikne i zakódováním znaku C, nejsme tedy schopni řetězec 01 jednoznačně dekódovat. V tomto případě není splněna podmínka jednoznačné dekódovatelnosti, tabulka tedy nepředstavuje použitelný kód.

Konec ilustračního příkladu

Terminologické poznámky:

1. Obecný pojem *kód* bývá často používán ve dvou významech – jednou jako označení pro zobrazení K (kódování znaků) , podruhé jako označení pro množinu všech kódových značek, tedy jako obor hodnot zobrazení K .
2. Místo pojmu *kódová značka* se často používá pojem *kódové slovo.*

2.2.2. Prefixové kódy

Prefixový kód je kód, v němž žádná kódová značka není prefixem jiné kódové značky.

Každý prefixový kód je jednoznačně dekódovatelný. Prefixové kódy lze dekódovat „znak po znaku“ již během přenosu, není třeba čekat na dokončení přenosu celé zprávy. K dekódování stačí konečný automat s výstupní funkcí Mealyho typu.

(Návrat k motivačnímu příkladu)

Na základě výše uvedené kódovací tabulky kódu 2, jenž je prefixový, zkonstruujeme konečný automat Mealyho typu, který bude modelem dekodéru.

Množinou vstupních symbolů automatu bude kódová abeceda $\Sigma = \{0,1\}$, množinou výstupních symbolů bude abeceda zdroje rozšířená o symbol *u* (mezera = neutrální znak), tedy $O = \{0,1,2,3,4,5,6,7,8,9,u\}$.

Množina stavů automatu bude odpovídat prefixům (předponám) kódových značek s výjimkou kódových značek samotných.

Počátečním stavem bude stav S, který odpovídá předponě *e* (tj. prázdnému řetězci).

Tabulku přechodové a výstupní funkce zkonstruujeme takto: Pokud je zpracován takový vstupní znak, jenž „doplní“ předponu reprezentovanou aktuálním stavem na nějakou kódovou značku, provede automat přechod do počátečního stavu S s tím, že na výstupu bude vygenerován zdrojový znak odpovídající této značce. Pokud „doplněním“ aktuální předpony o vstupní znak nevznikne kódová značka, ale jiná předpona, automat provede přechod do stavu odpovídajícího této nové předponě a na výstupu bude generován neutrální symbol *u* (jeho význam – dekodér ještě o výstupním symbolu nemůže jednoznačně rozhodnout).

Výchozí kódovací tabulka a tabulka reprezentující výsledný automat:

Kódovací tabulka

Zdrojový znak	Kódová značka
0	0
1	1000
2	1001
3	1010
4	1011
5	1100
6	1101
7	1110
8	11110
9	11111

Přechodová a výstupní tabulka dekódujícího Mealyho automatu

Stav	Vstupní znak	
	0	1
→ S	S/0	A/u
A	B/u	C/u
B	D/u	E/u
C	F/u	G/u
D	S/1	S/2
E	S/3	S/4
F	S/5	S/6
G	S/7	H/u
H	S/8	S/9

Poznámka

Prefix reprezentovaný stavem
e
1
10
11
100
101
110
111
1111

(Konec motivačního příkladu)

Z podmínky injektivnosti zobrazení K vyplývá omezení vztahující se na délky kódových značek. Při zadaném počtu prvků abecedy zdroje a abecedy kódu není možné vytvořit „libovolně krátké“ kódové značky – viz následující příklad.

Ilustrační příklad: Navrhněte tříznakový prefixový kód, který umožní zakódovat šestiprvkovou zdrojovou abecedu tak, že dvě kódové značky budou mít délku 1 a zbylé čtyři značky budou mít délku 2. Tedy

$Z = \{A, B, C, D, E, F\}$ ..... abeceda zdroje
$X = \{0, 1, 2\}$ ..... kódová abeceda

Kódovací tabulku začneme vytvářet tak, že první dva zdrojové znaky A a B zakódujeme znaky 0 a 1. Kód má být prefixový, znaky 0 a 1 proto nemohou být prefixem žádné jiné kódové značky, všechny čtyři zbylé značky tedy musí začínat znakem 2 . Pomocí tří znaků kódové abecedy ovšem nedokážeme na už jen jedné „zadáním povolené“ pozici rozlišit zbývající čtyři zdrojová písmena. Prefixový kód vyhovující zadání tedy nejde vytvořit. Zadání se „nejvíce blíží“ prefixový kód *K* v posledním sloupci tabulky, poslední dvě kódové značky ovšem překračují požadovanou délku značky.

Zdrojový znak	Požadovaná délka kódové značky	Pokus o konstrukci dle zadání	Korektní prefixový kód *K*
A	1	0	0
B	1	1	1
C	2	20	20
D	2	21	21
E	2	22	220
F	2	2**?**	221

(Přerušení ilustračního příkladu)

Vztah mezi délkami kódových značek, počtem prvků zdrojové abecedy a počtem prvků kódové abecedy popisuje *Kraftova nerovnost*:

$$s^{-d_1} + s^{-d_2} + \cdots + s^{-d_r} \leq 1\,, \quad \text{kde} \quad \begin{array}{ll} s & \text{je počet prvků kódové abecedy} \\ d_i & \text{je délka } i \text{ −té kódové značky} \\ r & \text{je počet prvků zdrojové abecedy} \end{array}$$

Je-li nerovnost splněna, lze sestrojit prefixový kód s předpokládanými délkami kódových značek d_i. Pokud nerovnost splněna není, prefixové kódování s předpokládanými délkami značek neexistuje.

(Návrat k ilustračnímu příkladu)

Pro hodnoty ze zadání spočítáme levou stranu Kraftovy nerovnosti:

$$s^{-d_1} + s^{-d_2} + s^{-d_3} + s^{-d_4} + s^{-d_5} + s^{-d_6} = 3^{-1} + 3^{-1} + 3^{-2} + 3^{-2} + 3^{-2} + 3^{-2} =$$
$$= \frac{1}{3} + \frac{1}{3} + \frac{1}{9} + \frac{1}{9} + \frac{1}{9} + \frac{1}{9} = \frac{3+3+1+1+1+1}{9} = \frac{10}{9}$$

Vidíme, že Kraftova nerovnost splněna není, což odpovídá tomu, že jsme kód se zadanými délkami značek nebyli schopni vytvořit.

Ověříme, že Kraftova nerovnost platí pro prefixový kód *K*, který vznikl jako vedlejší produkt při pokusu o konstrukci kódu s (nevhodně) zadanými délkami značek:

$$s^{-d_1} + s^{-d_2} + s^{-d_3} + s^{-d_4} + s^{-d_5} + s^{-d_6} = 3^{-1} + 3^{-1} + 3^{-2} + 3^{-2} + 3^{-3} + 3^{-3} =$$
$$= \frac{1}{3} + \frac{1}{3} + \frac{1}{9} + \frac{1}{9} + \frac{1}{27} + \frac{1}{27} = \frac{9+9+3+3+1+1}{27} = \frac{26}{27}$$

Hodnota levé strany je menší než 1, Kraftova nerovnost tedy je splněna. Hodnota levé strany není rovna 1, což souvisí s tím, že náš kód nevyužil všechny kódové kombinace (není využita „potenciálně možná“ kódová značka 222). Kdyby měla zdrojová abeceda o jeden znak více (tedy G), byl by zakódován jako 222. V součtu na levé straně nerovnosti by pak přibyl ještě jeden člen $\frac{1}{27}$, součet levé strany by pak byl 1.

(Konec ilustračního příkladu)

POZOR ! Kraftova nerovnost je formulována jako implikace (**jestliže délky značek vyhovují** Kraftově nerovnosti, **pak existuje** prefixový kód s takovými délkami značek). V žádném případě nelze implikaci obrátit a vyvozovat, že kód splňující Kraftovu nerovnost je prefixový a/nebo jednoznačně dekódovatelný!!

Ukazuje se, že prefixové kódy jsou dostatečně obecnou a reprezentativní podtřídou jednoznačně dekódovatelných kódů.

Mc Millanova věta: Každé jednoznačně dekódovatelné kódování splňuje Kraftovu nerovnost.

Přímým důsledkem Mc Millanovy věty je, že ke každému jednoznačně dekódovatelnému kódu lze sestrojit prefixový kód, který bude mít stejné délky kódových značek (v tom smyslu, že každé značce výchozího jednoznačně dekódovatelného kódu bude jednoznačně přiřazena značka prefixového kódu se stejnou délkou). Vzhledem k tomu, že prefixové kódy mají nejjednodušší mechanismus dekódování, budeme je chápat jako reprezentanty (všech) jednoznačně dekódovatelných kódů a kódy, které nejsou prefixové, se proto zabývat nebudeme.

2.2.3. Huffmanova konstrukce prefixového kódu

Dosud jsme se ve svých úvahách o prefixových kódech zabývali pouze tím, zda lze kód sestrojit a jak kód dekódovat. Zatím jsme se nesnažili zformulovat žádná „kritéria kvality“, na jejichž základě bychom dokázali vybrat optimální kód, natož tím, jak takový kód zkonstruovat. Požadavek na efektivní využití sdělovacího kanálu nebo paměťového média vede k prosté myšlence, kterou si uvědomil už tvůrce Morseovy abecedy: zdrojovým znakům, které se ve zprávách vyskytují s větší pravděpodobností, by měly příslušet kratší kódové značky než písmenům, jejichž výskyt má menší pravděpodobnost.

Exaktním ukazatelem, který umožní kódy vzájemně porovnávat ve výše uvedeném smyslu, je *střední délka kódové značky*. Abychom ovšem byli schopni tuto (v podstatě statistickou) veličinu vyjádřit, musíme znát rozložení pravděpodobností znaků zdrojové abecedy, tedy

$$A = \{a_1, a_2, \ldots., a_r\}\ ,\ \ P(A) = \big(p(a_1), p(a_2), \ldots., p(a_r)\big)\ \ ,\qquad \sum_{i=1}^{r} p(a_i) = 1$$

Střední délka kódové značky kódu K je pak definována jako

$$\overline{d}(K) = \sum_{i=1}^{r} p(a_i) \cdot d\big(K(a_i)\big)$$

Objasnění symbolů v definici střední délky kódové značky:

$\overline{d}(K)$ .. střední délka kódové značky kódu K
a_i ... písmeno zdrojové abecedy
$p(a_i)$ ... pravděpodobnost výskytu písmena a_i zdrojové abecedy ve zprávách
$K(a_i)$ ... kódová značka příslušející písmenu a_i zdrojové abecedy
$d\big(K(a_i)\big)$. délka kódové značky příslušející písmenu a_i

Takto definovaná střední délka kódové značky nám pro dané rozložení pravděpodobností zdrojové abecedy umožňuje z množiny jednoznačně dekódovatelných kódů vybrat ten s minimální střední délkou. Takový kód bude nejefektivnější, protože jím kódované zprávy zaberou ze všech kódů nejmenší kapacitu sdělovacího kanálu nebo paměti. Tohoto principu využívají všechny *statistické metody bezeztrátové komprese*.

Otázku, jakým způsobem k danému rozložení pravděpodobností kód s minimální střední délkou zkonstruovat, vyřešil v roce 1952 David Albert Huffman.

Huffmanova konstrukce prefixového kódu s minimální střední délkou kódové značky

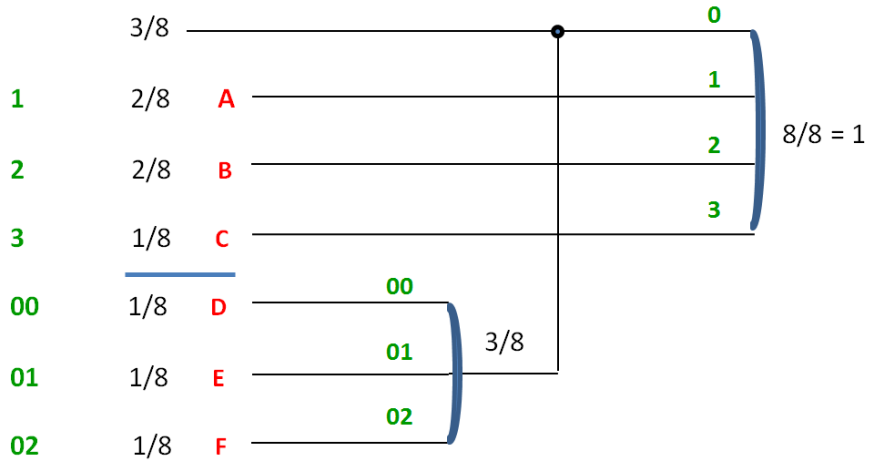
Vstupy algoritmu: zdrojová abeceda A o r prvcích a její pravděpodobnostní rozložení $P(A)$
kódová abeceda B o s prvcích

Výstup algoritmu: prefixové kódování $K : A \rightarrow B^+$ takové, že $\overline{d}(K)$ je minimální (tj. neexistuje jiné kódování $\widetilde{K}$, pro které by platilo $\overline{d}(\widetilde{K}) < \overline{d}(K)$)

Algoritmus:

1. Prvky zdrojové abecedy A seřadíme podle jejich pravděpodobností $p(a_i)$ do neklesající posloupnosti.
2. Takto seřazené prvky rozdělíme do skupin. Začínáme od prvků s nejvyšší pravděpodobností. Skupiny budou mít $s-1$ prvků. Výjimkou může být poslední skupina, která může mít od 2 do s prvků.
3. Sdružíme prvky v poslední skupině a nahradíme je *sdruženou skupinou*, kterou zařadíme podle její součtové pravděpodobnosti na správné místo do posloupnosti. Rozdělení provedené v bodě 2 v tuto chvíli zaniká.
4. Sdružíme posledních s prvků v posloupnosti a nahradíme je sdruženou skupinou, kterou zařadíme podle její součtové pravděpodobnosti na správné místo do posloupnosti.
5. Bod 4 opakujeme tak dlouho, dokud nezískáme jedinou sdruženou skupinu se součtem 1.
6. Zpětným chodem po větvích s –árního stromu vytvořeného v bodech 3 až 5 přiřadíme kódové značky listům stromu, tj. znakům zdrojové abecedy.

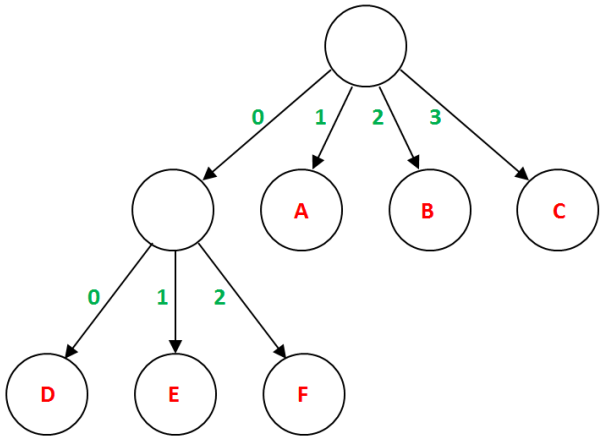
Ilustrační příklad: Do čtyřprvkové abecedy $X = \{0,1,2,3\}$ zakódujte zdrojovou abecedu $Z = \{A, B, C, D, E. F\}$ s rozložením $P(Z) = \left(\frac{2}{8}, \frac{2}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}\right)$.



Poznámky ke konstrukci kódu v ilustračním příkladu:

1. Dělením podle bodu 2 algoritmu vznikly dvě skupiny znaků – obě o třech prvcích.
2. Sdružením prvků v poslední (tj. druhé) skupině vznikl objekt – sdružená skupina s pravděpodobností 3/8, kterou jsme zařadili podle její součtové pravděpodobnosti na začátek posloupnosti. Posloupnost teď tvoří 4 objekty – skupina DEF, prvek A, prvek B, prvek C s pravděpodobnostmi (v uvedeném pořadí) 3/8, 2/8, 2/8, 1/8.
3. Sdružením posledních čtyř členů posloupnosti (DEF, A, B, C) získáme skupinu s celkovým součtem pravděpodobností 8/8 = 1.
4. Objekty posledního sloučení rozlišíme symboly kódové abecedy: DEF **0**, A **1**, B **2**, C **3.**
5. Skupina DEF se skládá z objektů, které musíme navzájem rozlišit. Všechny objekty skupiny přebírají jako prefix kód přiřazený skupině, navzájem se rozliší dalším symbolem, takže D bude zakódováno **00**, E **01**, F **02.**
6. Protože tím máme přiřazeny kódové značky všem zdrojovým symbolům, zpětný chod končí.

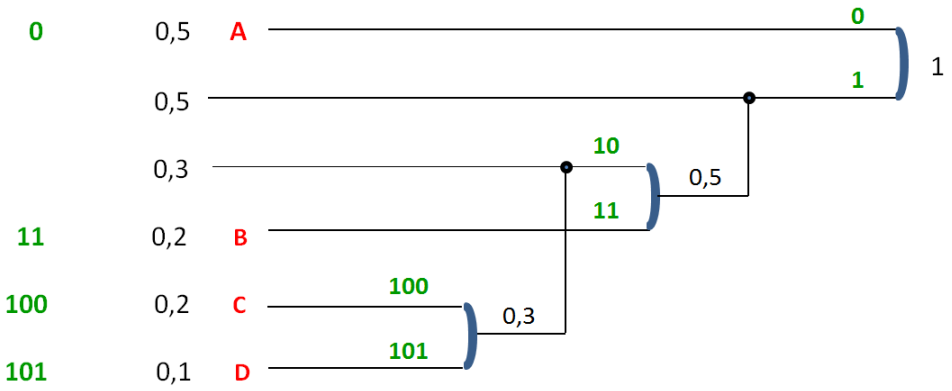
Pro větší názornost uvedeme ještě i běžnější způsob grafické reprezentace stromu, na němž bude lépe vidět přiřazování kódových písmen hranám stromu zpětným chodem. Písmena zdrojové abecedy se nachází v listech stromu, kódová značka příslušející konkrétnímu zdrojovému písmenu je pak ohodnocením cesty od kořene stromu ke konkrétnímu listu:



(Konec ilustračního příkladu)

Při kódování zpráv Huffmanovým kódem dochází k redukci redundance, redundance po zakódování je tedy menší než redundance ve zdrojové abecedě. Huffmanovo kódování tak provádí určitou kompresi dat. Redukci redundance ilustruje následující příklad.

Ilustrační příklad: Do dvouprvkové abecedy $X = \{0,1\}$ zakódujte zdrojovou abecedu $Z = \{A, B, C, D\}$ s rozložením $P(Z) = (0{,}5\ ,\ 0{,}2\ ,\ 0{,}2\ ,\ 0{,}1)$.



Poznámka: Při kódování do dvouprvkové abecedy nemá dělení do skupin (bod 2 výše popsaného algoritmu) význam. Začíná se rovnou sloučením posledních dvou prvků.

Nejprve spočítáme střední entropii a redundanci ve zdrojové abecedě:

$$H(Z) = -(0{,}5 \cdot \log_2 0{,}5 + 0{,}2 \cdot \log_2 0{,}2 + 0{,}2 \cdot \log_2 0{,}2 + 0{,}1 \cdot \log_2 0{,}1) = 1{,}76$$

$$\rho(Z) = 1 - \frac{1{,}76}{\log_2 4} = 1 - \frac{1{,}76}{2} = 0{,}12$$

Pro výpočet pravděpodobnostního rozložení kódové abecedy vyjádříme N_0 (střední počet znaků 0 ve značce), N_1(střední počet znaků 1 ve značce) a střední délku značky $\overline{d}$. Pravděpodobnosti písmen zdrojové abecedy, jejich kódování a počet nul a jedniček ve značkách jsou uvedeny v tabulce:

Zdrojový znak	P-st znaku	Kódová značka	Délka kódové značky	Počet znaků	
				0	1
A	0,5	0	1	1	0
B	0,2	11	2	0	2
C	0,2	100	3	2	1
D	0,1	101	3	1	2
		Střední hodnota:	**1,8**	**1,0**	**0,8**

$$\overline{d} = 0{,}5 \cdot 1 + 0{,}2 \cdot 2 + 0{,}2 \cdot 3 + 0{,}1 \cdot 3 = 1{,}8$$

$$N_0 = 0{,}5 \cdot 1 + 0{,}2 \cdot 0 + 0{,}2 \cdot 2 + 0{,}1 \cdot 1 = 1{,}0 \qquad p_0 = \frac{N_0}{\overline{d}} = \frac{1{,}0}{1{,}8} = \frac{5}{9}$$

$$N_1 = 0{,}5 \cdot 0 + 0{,}2 \cdot 2 + 0{,}2 \cdot 1 + 0{,}1 \cdot 2 = 0{,}8 \qquad p_1 = \frac{N_1}{\overline{d}} = \frac{0{,}8}{1{,}8} = \frac{4}{9}$$

Nyní již můžeme spočítat střední entropii a redundanci v kódové abecedě:

$$H(X) = -\left(\frac{5}{9}\cdot\log_2\frac{5}{9}+\frac{4}{9}\cdot\log_2\frac{4}{9}\right) = 0{,}99 \qquad \rho(X) = 1 - \frac{0{,}99}{\log_2 2} = 1 - \frac{0{,}99}{1} = 0{,}01$$

Redundance 12% ve zdrojových zprávách se tedy Huffmanovým kódováním zredukovala na pouhé 1%.

(Konec ilustračního příkladu)

Nejjednodušší metoda komprese založená na Huffmanově konstrukci probíhá ve dvou fázích. V první fázi (první průchod zdrojovým souborem) se vytvoří statistika četností zdrojových znaků a z ní pak „kódovací strom". Ve druhém průchodu zdrojovým souborem pak dochází ke kódování (kompresi) vstupních dat.

Poznámka: Mezi statistické metody bezeztrátové komprese patří také Shannonovo-Fanovo binární kódování. Od Huffmanova kódování se liší pouze konstrukcí (binárního) stromu. Množina znaků je rekurzivně dělena vždy na dvě podmnožiny tak, aby součet výskytů znaků v obou podmnožinách byl „přibližně stejný". Shannon - Fanův kód ale (na rozdíl od Huffmanova kódování) nemusí být optimální (tj. nemusí mít minimální střední délku kódové značky), proto mu není třeba věnovat více pozornosti.

2.3. Bezpečnostní kódy

2.3.1. Modelové důsledky šumu

Přenášená a uchovávaná data mohou být ovlivněna negativními vlivy vnějšího prostředí (v modelu z kapitoly 1.3 *šumem*). Tomu lze čelit vhodným kódováním, které bude do zpráv přidávat redundandní informace, jež (za určitých pravděpodobnostních předpokladů) umožní dekodéru výskyt přenosové chyby detekovat nebo dokonce tuto chybu lokalizovat a opravit.

Budeme předpokládat dva možné důsledky šumu:

- záměnu vyslaného znaku za jiný znak
- porušení synchronizace (tj. ztráta znaku nebo vytvoření „falešného znaku")

Záměnu vyslaného znaku za jiný znak můžeme matematicky popsat metodami z kapitoly 1.3.2 . Reálnou příčinou těchto chyb může být například vnější elektromagnetické rušení při přenosu dat na metalickém spoji, kosmické záření, ale třeba i tepelný šum v elektronických součástkách.

Příčinou porušení synchronizace může být například chyba komunikačního adaptéru nebo chyba některého jiného článku přenosové cesty. Chyby tohoto typu jsou jiného charakteru

než záměna vyslaného znaku – vznikají převážně nesprávnou funkcí komponent přenosového kanálu. Tyto stavy jsou v reálných přenosových systémech rychle diagnostikovány a opraveny. Šum typu „porušení synchronizace“ tedy budeme považovat za „mimořádnou událost“, kterou je třeba řešit jinými nástroji než kódováním.

V dalších úvahách tedy budeme předpokládat pouze šum typu „záměna vyslaného znaku“.

Budeme pracovat pouze s blokovými kódy, tedy s kódy, kde všechny kódové značky mají stejnou délku. Délku značky budeme označovat n. Budeme používat termín, *kód délky* n.

2.3.2. Hammingova vzdálenost a její význam pro zabezpečení

Motivační příklad: Binární kódování hexadecimálních číslic.

$Z = \{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\}$ ..... abeceda zdroje
$X = \{0,1\}$ ..... abeceda kanálu

Kódování každému zdrojovému znaku přiřadí řetězec tvořený znaky z abecedy kanálu. Všechny značky kódu budou mít stejnou délku. Příklady dvou kódování jsou v tabulce :

Zdrojový znak	Kódová značka	
	Kód 1	Kód 2
0	0000	00000
1	0001	00011
2	0010	00101
3	0011	00110
4	0100	01001
5	0101	01010
6	0110	01100
7	0111	01111
8	1000	10001
9	1001	10010
A	1010	10100
B	1011	10111
C	1100	11000
D	1101	11011
E	1110	11101
F	1111	11110

Kód 1 představuje „standardní“ reprezentaci šestnáctkových cifer čtveřicemi z nul a jedniček. Lze vytvořit 16 různých čtveřic, potřebujeme zakódovat 16 zdrojových znaků, všechny čtveřice jsou tedy využity jako kódové značky. Důsledkem je to, že při přenosu libovolné kódové značky a při libovolné chybě lze vždy „chybou ovlivněnou“ čtveřici bitů interpretovat jako kódovou značku, po dekódování tedy příjemce není schopen zjistit, že přijal jiné zdrojové písmeno, než bylo vysláno. Kód 1 tedy nemá žádnou zabezpečovací schopnost.

Kód 2 vznikl z kódu 1 tak, že každá čtyřbitová kódová značka byla rozšířena o *paritu*, tedy o pátý bit, jenž byl zvolen tak, aby celkový počet jedniček ve značce byl sudý. Na pěti bitech lze vytvořit 32 různých pětic, ovšem pouze 16 z nich je využito jako kódové značky. Množina všech pětic z nul a jedniček tedy byla rozdělena na dvě (v našem případě stejně početné) podmnožiny: množinu kódových značek a *množinu nekódových kombinací*. Důsledkem je to, že při přenosu libovolné kódové značky a při chybě v jednom prvku přenášené značky dekodér rozpozná, že počet jedniček ve značce není sudý, a detekuje přenosovou chybu. Kód 2 tedy má schopnost detekovat jednoduchou chybu, není ovšem schopen tuto chybu opravit. Např. nekódová kombinace 11111 mohla vzniknout jednoduchou chybou z kódových značek 01111, 10111, 11011, 11101, 11110, mohla vzniknout i vícenásobnou chybou z jiných značek, nelze tedy jednoznačně rozhodnout o tom, jaká kódová značka byla do kanálu vložena.

Snadno spočítáme redundance obou kódů: $\rho(K_1) = 0$, $\rho(K_2) = 1 - \frac{\log_2 16}{\log_2 32} = 1 - \frac{1}{5} = \frac{1}{5}$
Je zřejmé, že nenulová redundance kódu souvisí s tím, že pro kódové značky není využita celá množina všech n-tic ze znaků 0 a 1.

(Přerušení motivačního příkladu)

Předpokládejme blokový kód K délky n nad kódovou abecedou T, tedy

$K \subseteq T^n$, kde

$T^n = \{t_1 t_2 \ldots . . t_n \mid t_i \in T \quad \forall\, i = 1,1, \ldots, n\}$ je množina všech slov délky n nad T

Pak existuje dvoublokový rozklad množiny T^n na *kódové značky* K a *nekódové kombinace* $T^n - K$.

Kodér na straně zdroje dat do sdělovacího kanálu vkládá kódové značky z množiny K , dekodér na straně příjemce z kanálu přijímá obecně slova z T^n .

Přijetí nekódové kombinace, tedy slova z $T^n - K$ znamená, že při přenosu došlo k chybě, jinak řečeno - příjemce *detekuje chybu*.

Přijetí kódové kombinace, tedy slova z K znamená přesně to, že **buď** při přenosu nedošlo k chybě, **nebo** že došlo k takové chybě, kterou dekodér příjemce není schopen detekovat.

Při úvahách o bezpečnostních kódech budeme používat formulace typu „kód K detekuje t-násobné chyby“, respektive „kód K opravuje t-násobné chyby“. Proto je třeba jednoznačně popsat, co rozumíme pod pojmem „t-násobná chyba“.

t-násobnou chybou rozumíme (libovolnou) chybu, kde je počet chybně přenesených prvků **menší nebo roven** t .

Nyní již můžeme formulovat odpověď na otázku **Kdy kód detekuje *t*-násobnou chybu?**

Kód K detekuje t-násobnou chybu právě tehdy, jestliže je při vyslání **libovolné** kódové značky a **libovolné** t-násobné chybě přijata **vždy** nekódová kombinace.

Důsledkem výše uvedené interpretace pojmu t - násobná chyba je fakt, že např. z tvrzení „kód K detekuje čtyřnásobné chyby“ plyne, že detekuje i všechny chyby nižší násobnosti, tedy trojité, dvojité a jednoduché. Jinak řečeno – pokud kód nedetekuje např. dvojnásobné chyby, nemůže detekovat ani žádné chyby vyšší násobnosti.

(Návrat k motivačnímu příkladu)

V množině kódových značek kódu K_2 jsou například značky 01111 a 10111. Vyslaná značka **0**1111 se může jednoduchou chybou změnit na nekódovou kombinaci **1**1111, stejně tak se může 1**0**111 jednoduchou chybou změnit na stejnou nekódovou kombinaci 1**1**111. Je to možné proto, že se značky **01**111 a **10**111 liší právě ve dvou prvcích (prvním a druhém). Je zřejmé, že o tom, jaké chyby je kód schopen detekovat, rozhoduje počet prvků, v nichž se navzájem liší dvě „nejbližší“ značky. V množině kódových značek kódu K_2 bychom našli několik takových dvojic, jež se liší právě ve dvou prvcích, ale nenašli bychom tam žádnou dvojici značek, která se navzájem liší v jediném prvku (to nám zajišťuje parita).

(Konec motivačního příkladu)

Hammingovou vzdáleností slov u a v (stejné délky n) se rozumí počet pozic, v nichž se slova u a v liší. Formálně:

$$u = u_1 u_2 \ \ldots\ldots u_n$$
$$v = v_1 v_2 \ \ldots\ldots v_n$$
$$d(u,v) = \ \mathrm{card}\,\{i \mid \ u_i \neq v_i\ , i = 1,2,\ldots,n\}$$

Poznámka: Symbol card představuje mohutnost množiny, v našem případě mohutnost konečné množiny, tedy počet jejích prvků.

Minimální Hammingovou vzdáleností kódu K se rozumí nejmenší Hammingova vzdálenost mezi dvěma různými kódovými značkami kódu K. Formálně:

$$d_0(K) = \min_{u,v \in K,\ \ u \neq v} \ \ d(u,v)$$

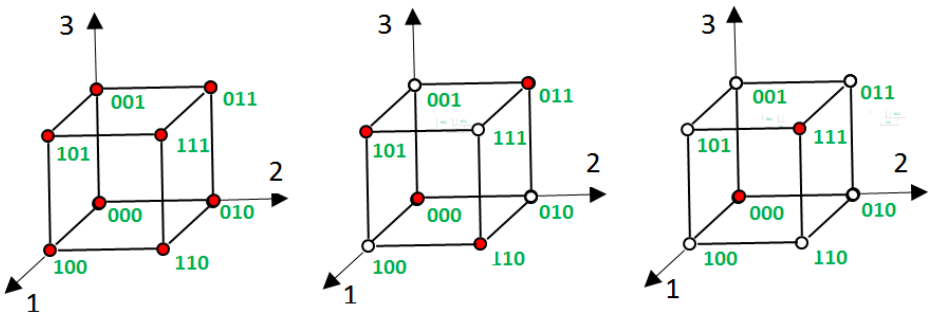
Je zřejmé, že o tom, jaké chyby je schopen kód detekovat, rozhoduje minimální Hammingova vzdálenost kódu. Ukážeme si to na následujícím názorném příkladu.

Ilustrační příklad: Binární kódy vybrané z $\{0,1\}^3$ („navlékání kódu na krychli“).

Budeme se zabývat třemi binárními kódy zadanými množinami kódových značek:

$$K_1 = \{000, 001, 010, 011, 100, 101, 110, 111\}$$
$$K_2 = \{000, 011, 101, 110\}$$
$$K_3 = \{000, 111\}$$

V následujícím obrázku máme všechny tři kódy graficky znázorněny. Červeně vybarvené vrcholy krychlí představují kódové značky, bílé vrcholy představují nekódové kombinace.



K_1 K_2 K_3

Z obrázků jsou zřejmé minimální Hammingovy vzdálenosti všech tří kódů:

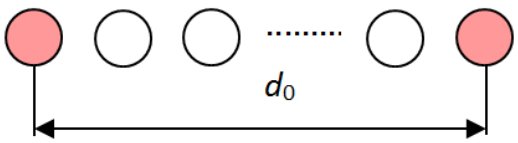
$$d_0(K_1) = 1 \qquad d_0(K_2) = 2 \qquad d_0(K_3) = 3$$

Z obrázků lze vypozorovat, že:

- Kód K_1 nedetekuje žádné chyby (všech osm kombinací je využito jako kódové značky)
- Kód K_2 detekuje jednoduché chyby (neexistují dvě kódové značky, které by se lišily jen v jednom prvku, tj. ležely na jedné hraně krychle; při vyslání libovolné kódové značky a jednoduché chybě v libovolné pozici vždy vznikne nekódová kombinace)
- Kód K_3 detekuje dvojité chyby (Hammingova vzdálenost mezi jedinými dvěma značkami je 3; trojitá chyba už by z kódové značky vytvořila jinou kódovou značku)

Kód K_3 je příkladem *opakovacího kódu*, v tomto případě s délkou 3.

Je zřejmé, že pro delší kódové značky a kódové abecedy s větším počtem prvků již nejsme schopni celou množinu kódových i nekódových kombinací srozumitelně znázornit, proto bývá zvykem v úvahách o zabezpečovacích vlastnostech kódů znázorňovat pouze „nejužší místo kódu“, tedy právě ty dvě značky, které určují minimální Hemmingovu vzdálenost a všechny nekódové kombinace ležící na nejkratší cestě mezi nimi:



(Přerušení ilustračního příkladu)

Zobecnění:
Blokový kód K s minimální Hammingovou vzdáleností d_0 detekuje všechny chyby s násobností $t < d_0$.

Důsledky:
Paritní kód detekuje jednoduché chyby.
Opakovací kód délky n detekuje všechny chyby s násobností $t < n$.

Za určitých předpokladů lze přenosové chyby nejen detekovat, ale i opravovat.

Budeme předpokládat symetrický binární sdělovací kanál s bitovou chybovostí (Bit Error Rate) p a statisticky nezávislé přenosové chyby. Výskyt přenosové chyby v i −tém přenášeném prvku n −prvkové značky tedy nezávisí na tom, zda se chyba vyskytne v jiných přenášených prvcích značky.

Chybovost p představuje pravděpodobnost, že při přenosu jednoho konkrétního prvku došlo k chybě. Na reálných přenosových kanálech chybovost závisí zejména na *odstupu signálu od šumu* (Signal to Noise Ratio). Za chybovost dostatečnou k běžnému datovému přenosu, je považována hodnota $p = 10^{-6}$. Kvalitní metalické spoje mají chybovost v řádu $p \approx 10^{-8}$, na optických vláknech je dosahováno chybovosti v řádech $p \approx 10^{-10}$ až $p \approx 10^{-12}$.

Označme symboly $p_0, p_1, p_2, \ldots, p_n$ pravděpodobnosti toho, že při přenosu značky délky n dojde k bezchybnému přenosu, k jednoduché chybě, dvojité chybě atd., až k chybě ve všech n prvcích.

Za výše uvedených předpokladů pak můžeme vyjádřit

pravděpodobnost bezchybného přenosu p_0: $p_0 = (1-p)^n$
pravděpodobnost jednoduché chyby **v jednom konkrétním (*i*-tém) bitu**: $p \cdot (1-p)^{n-1}$
pravděpodobnost chybného přenosu jednoho bitu ve značce p_1: $p_1 = n \cdot p \cdot (1-p)^{n-1}$
pravděpodobnost dvojité chyby **v konkrétní dvojici bitů (*i,j*)**: $p^2 \cdot (1-p)^{n-2}$
pravděpodobnost chybného přenosu dvou bitů ve značce p_2: $p_2 = \binom{n}{2} \cdot p^2 \cdot (1-p)^{n-2}$

pravděpodobnost chyby s násobností t : $p_t = \binom{n}{t} \cdot p^t \cdot (1-p)^{n-t}$

Následující tabulka ukazuje pravděpodobnosti $t-$ násobných chyb na osmiprvkových značkách při chybovostech reálných komunikačních kanálů zmíněných výše:

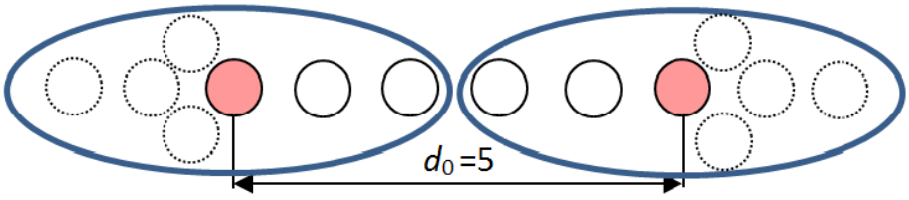
Násobnost chyby t	bitová chybovost p			
	1,0E-06	**1,0E-08**	**1,0E-10**	**1,0E-12**
0	0,99999200	0,99999992	1,00000000	1,00000000
1	0,00000800	0,00000001	1,0000E-10	1,0000E-12
2	2,8000E-11	1,0000E-16	1,0000E-20	1,0000E-24
3	5,6000E-17	1,0000E-24	1,0000E-30	1,0000E-36
4	7,0000E-23	1,0000E-32	1,0000E-40	1,0000E-48
5	5,6000E-29	1,0000E-40	1,0000E-50	1,0000E-60
6	2,8000E-35	1,0000E-48	1,0000E-60	1,0000E-72
7	8,0000E-42	1,0000E-56	1,0000E-70	1,0000E-84
8	1,0000E-48	1,0000E-64	1,0000E-80	1,0000E-96

Z tabulky je zřejmé, že pravděpodobnosti chyb menších násobností jsou o mnoho řádů větší, než pravděpodobnosti chyb vyšších násobností. Tabulka tak ilustruje oprávněnost opravování chyb na principu nalezení „nejbližší“ kódové značky ve smyslu Hammingovy vzdálenosti. Pokud lze „nejbližší“ kódovou značku určit jednoznačně, lze přijatou nekódovou kombinaci opravit na nalezenou kódovou značku. Pokud je „nejbližších“ značek několik, nelze přijatou nekódovou kombinaci jednoznačně opravit.

Můžeme tedy formulovat odpověď na otázku **Kdy kód opravuje t-násobnou chybu?**

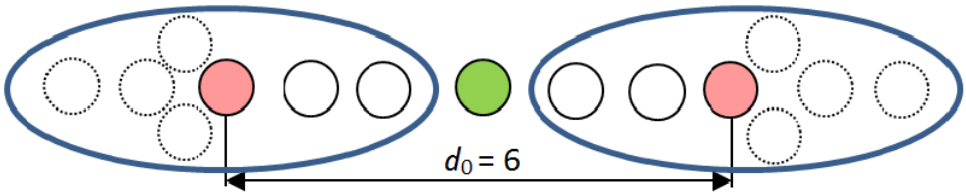
Kód K opravuje t-násobnou chybu právě tehdy, jestliže při vyslání **libovolné** kódové značky $v \in K$ a při **libovolné** t-násobné chybě má přijaté slovo $w \in T^n$ Hammingovu vzdálenost od vyslané kódové značky v menší, než od libovolné jiné kódové značky, platí tedy
$\forall x \in K: \; x \neq v \; \Rightarrow \; d(v,w) < d(x,w)$.

Graficky lze opravování na principu nalezení „nejbližší“ kódové značky znázornit tako:



Obrázek zobrazuje „nejužší místo“ kódu. Červeně je znázorněna dvojice kódových značek, která definuje minimální Hammingovu vzdálenost kódu, bílá kolečka ohraničená plnou čarou představují nekódové kombinace mezi těmito kódovými značkami, tedy „v nejužším místě“ kódu. Kolečka ohraničená přerušovanou čarou naznačují existenci dalších nekódových kombinací na „cestách k jiným kódovým značkám“. Modré ovály představují množiny všech n – tic, které příjemce opraví na stejnou kódovou značku. Tyto množiny jsou tvořeny vždy jednou kódovou značkou a všemi nekódovými kombinacemi, které jsou této kódové značce bližší než k jiným kódovým značkám. V případě kódu s minimální vzdáleností d_0= 5 jsou to všechny všechny kombinace, které mají od kódové značky vzdálenost menší nebo rovnu 2.

Ne vždy lze všem nekódovým kombinacím přiřadit jednoznačně kódové značky, jak ukazuje ilustrační obrázek pro kód se sudou minimální Hammingovou vzdáleností:

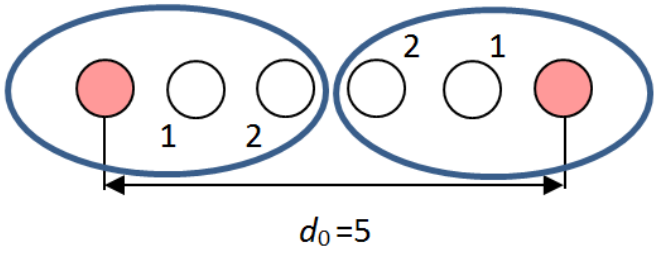


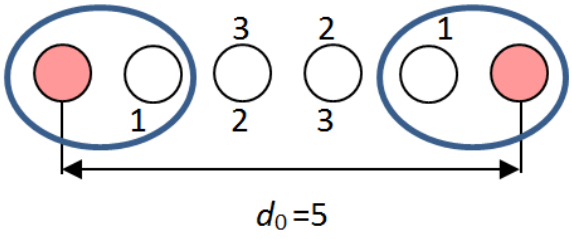
Zeleně zvýrazněná nekódová kombinace má stejnou Hammingovu vzdálenost od obou kódových značek, proto ji není možné opravit.

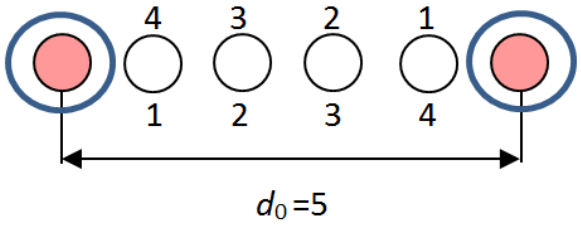
Zobecnění:
Blokový kód K s minimální Hammingovou vzdáleností d_0 opravuje všechny chyby s násobností $t < d_0/2$. Jinak řečeno – má-li kód opravovat $t-$ násobné chyby, musí mít minimální Hammingovskou vzdálenost $d_0 \geq 2 \cdot t + 1$.

Důsledek:
Opakovací kód **liché** délky n opravuje všechny chyby s násobností $t \leq (n-1)/2$, opakovací kód **sudé** délky n opravuje všechny chyby s násobností $t \leq (n-2)/2$.

Poznámka: Výše popsaný způsob opravování chyb je založen na tom, že nepředpokládáme výskyt chyb s vyšší násobností, než opravujeme. Znamená to, že pokud by při přenosu došlo k porušení většího počtu bitů než opravujeme, mohlo by dojít k „chybné opravě“ v tom smyslu, že příjemce opraví přijatou nekódovou kombinaci na jinou kódovou značku, než byla odeslána. Příjemce může (čistě teoreticky) zvolit i smíšenou strategii, kdy neopravuje všechny chyby, které mu minimální Hammingova vzdálenost umožňuje. Tím zvýší počet chyb, které je schopen detekovat. Příklad různých strategií opravy/detekce pro kód s minimální vzdáleností $d_0 = 5$ znázorňuje následující obrázek:







Nevybarvená kolečka uvnitř modrých oválů představují nekódové kombinace, které budeme opravovat, kolečka mimo tyto ovály představují nekódové kombinace, které detekují chybu, ale nebudou opravovány. Je zřejmé, že existuje závislost mezi násobností chyb, jež budeme ve smíšené strategii opravovat a které budeme pouze detekovat. Konkrétně pro $\mathrm{d}_0 = 5$ vidíme, že budeme-li dvojité chyby opravovat, nelze už žádné další chyby detekovat, tedy např. trojitou chybu může příjemce opravit špatně. V případě opravy pouze jednoduchých chyb získáváme možnost detekce trojitých chyb, nebudeme-li opravovat žádné chyby, můžeme detekovat chyby čtyřnásobné.

Nyní již můžeme přejít k matematické formulaci toho, co budeme považovat za opravování chyb (budeme je nazývat *dekódování*):

Dekódováním s opravou $t - \text{násobných chyb}$ budeme rozumět parciální funkci $\delta : T^n \rightarrow K$ takovou, že $\delta(w) = v \Leftrightarrow d(w,v) \leq t$.

Samozřejmá vlastnost fukce δ: $\delta(v) = v \quad \forall\, v \in K$ (dekódování nemění kódové značky)

Poznámka 1: *Parciální funkcí* $f : A \rightarrow B$ se rozumí funkce, u které, na rozdíl od (*totální*) funkce, nemusí být funkční hodnota definována pro všechny prvky množiny A.

Poznámka 2: Termín *dekódování* bývá zvykem používat i v jiném významu, a to jako „extrakce“ informační části ze zakódované značky, jak uvidíme později.

Je zřejmé, že dekódovací funkce δ jednoznačně definuje rozklad množiny všech n – tic T^n : $T^n = \{K_1, K_2, \ldots, K_{card\,K}, K_D\}$, kde

$card\,K$ je počet kódových značek
$K = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_{card\,K}\}$ je množina kódových značek
K_i je množina všech n–tic, které se opravují na kódovou značku $\boldsymbol{v}_i$
$(\forall\, i = 1,2,\ldots, card\,K)$
K_D je množina všech n – tic, které pouze detekují chybu a neopravují se

Prvkem každé třídy rozkladu kromě třídy K_D je právě jedna kódová značka a s ní jsou prvky této třídy i všechny nekódové kombinace, které se na tuto kódovou značku opravují.

(Návrat k ilustračnímu příkladu „navlékání kódu na krychli“):

V následující tabulce jsou uvedeny dekódovací funkce δ pro všechny tři dříve uvedené kódy:

Přijatá trojice *w*		**Dekódovací funkce δ**					
		Kód 1		**Kód 2**		**Kód 3**	
$\boldsymbol{w}_1$	000	$\boldsymbol{v}_1$	000	$\boldsymbol{v}_1$	000	$\boldsymbol{v}_1$	000
$\boldsymbol{w}_2$	001	$\boldsymbol{v}_2$	001		X		000
$\boldsymbol{w}_3$	010	$\boldsymbol{v}_3$	010		X		000
$\boldsymbol{w}_4$	011	$\boldsymbol{v}_4$	011	$\boldsymbol{v}_2$	011		111
$\boldsymbol{w}_5$	100	$\boldsymbol{v}_5$	100		X		000
$\boldsymbol{w}_6$	101	$\boldsymbol{v}_6$	101	$\boldsymbol{v}_3$	101		111
$\boldsymbol{w}_7$	110	$\boldsymbol{v}_7$	110	$\boldsymbol{v}_4$	110		111
$\boldsymbol{w}_8$	111	$\boldsymbol{v}_8$	111		X	$\boldsymbol{v}_2$	111

Dekódovací funkce pro kódy 1 a 3 jsou (totální) funkce, každé přijaté trojici je jednoznačně přiřazena kódová značka. U kódu 1, který pro kódové značky využívá všechny trojice, je dekódovací funkcí identita, u kódu 3 je funkční hodnota určena znakem, který má v přijaté trojici většinu. Dekódovací funkce kódu 2 je pouze parciální funkcí, některé přijaté trojice nemají přiřazenu hodnotu (kódovou značku), což vyjadřuje symbol X na místě funkční hodnoty. Podbarvené hodnoty představují kódové značky, jež při dekódování vznikly opravou přijaté nekódové kombinace.

Dekódovací funkce z tabulky jednoznačně definují rozklady množiny všech trojic:

Kód 1:
Každá značka je samostatnou třídou rozkladu, $K_i = \{\boldsymbol{w}_i\}\ \forall\ i = 1,2,\dots, card\ K,\ K_D = \emptyset$.

Kód 2:
$K_1 = \{000\}\ ,\ K_2 = \{011\}\ ,\ K_3 = \{101\},\ K_4 = \{110\}\ ,\ K_D = \{001, 010, 100, 111\}$

Kód 3:
$K_1 = \{000, 001, 010, 100\}\ ,\ K_2 = \{011, 101, 110, 111\}\ ,\ K_D = \emptyset\ .$

(Konec ilustračního příkladu „navlékání kódu na krychli“):

V předchozích odstavcích jsme formulovali podmínky, které musí být splněny, aby kód opravoval nebo detekoval $t-$ násobné chyby, zatím jsme ale neřešili, jak detekci/opravu realizovat. Obecně lze konstatovat, že porovnávání přijaté $n-$ tice se všemi kódovými značkami je neefektivní, navíc pro každou značku trvá různou dobu. Jednu z takových cest naznačuje následující algoritmus. Později se ale seznámíme s třídami kódů, u kterých lze detekci/opravu chyb realizovat mnohem efektivněji.

Algoritmus dekódování s opravou *t*-násobných chyb na principu nalezení „nejbližší“ kódové značky

Vstupy algoritmu: přijatá n – tice $\boldsymbol{w} \in T^n$, množina kódových značek K, násobnost opravovaných chyb t $(t < d_0(K)/2)$

Výstup algoritmu: kódová značka $\boldsymbol{v} \in K$ „nejbližší“ přijaté n – tici w (pokud existuje), taková, že $d(\boldsymbol{w}, \boldsymbol{v}) \leq t$ a je minimální (tj. neexistuje jiná kódová značka $\tilde{\boldsymbol{v}} \in K$, pro kterou by platilo $d(\boldsymbol{w}, \tilde{\boldsymbol{v}}) < d(\boldsymbol{w}, \boldsymbol{v})$)

Algoritmus:

1. Pro k od 0 do t provedeme kroky 2 a 3
2. Vytvoříme množinu $O_k \subseteq T^n$ jako množinu všech n – tic $\boldsymbol{u} \in T^n$ takových, že $d(\boldsymbol{u}, \boldsymbol{w}) = k$.
3. Pokud je prvkem množiny O_k kódová značka (tj. $O_k \cap K = \{\boldsymbol{v}\}$), je značka $\boldsymbol{v}$ výstupem algoritmu, algoritmus končí.
4. Hledaná kódová značka neexistuje, algoritmus končí.

2.3.3. Obecné vlastnosti lineárních kódů

Detekce nebo opravování chyb není principielně nic jiného, než rozhodnutí, zda n–tice $\boldsymbol{w} \in T^n$ je, či není prvkem nějaké množiny (tj. buď prvkem množiny kódových značek K, nebo prvkem nějaké třídy rozkladu K_i ve smyslu kapitoly 2.3.2). Jeden z přístupů k realizaci opravování chyb naznačil výše uvedený algoritmus, jiným přístupem může být třeba „prokládané“ prohledávání tabulek, v nichž budou uloženy (seřazeny podle pravděpodobnosti) jednotlivé prvky tříd rozkladu $T^n = \{K_1, K_2, \ldots., K_{|K|}, K_D\}$. Již dříve jsme ovšem konstatovali, že takové přístupy nejsou optimální vzhledem k relativní časové náročnosti a nestejné délce vyhledávacích operací.

Během studia matematiky jsme se ovšem již setkali s oblastmi, v nichž jsme schopni otázku typu **Je $\boldsymbol{x}$ prvkem množiny A?** rozhodnout „výpočetně“, tj. s konstantní délkou zpracování pro všechna x , pro které tato otázka má smysl.

Motivační příklad z analytické geometrie v prostoru:

Rovina ρ procházející počátkem soustavy souřadnic O je určena normálovým vektorem $\vec{n}$. Dále je dán bod A . Rozhodněte, zda bod A leží v rovině ρ .

Řešení: Vytvoříme vektor $\vec{a}$ jako průvodič bodu A , tedy $\vec{a} = A - O$. Pokud bod A leží v rovině ρ , je jeho průvodič $\vec{a}$ kolmý na normálový vektor $\vec{n}$, skalární součin $\vec{a} \cdot \vec{n}$ tedy bude nulový. Pokud bod A v rovině ρ neleží, svírá jeho průvodič s normálovým vektorem jiný úhel než pravý, skalární součin $\vec{a} \cdot \vec{n}$ je proto nenulový. Platí tedy

$$A \in \rho \iff a_x \cdot n_x + a_y \cdot n_y + a_z \cdot n_z = 0$$

Zobecnění: Na tuto geometrickou úlohu se můžeme podívat obecněji, čímž uvedený přístup k řešení rozšíříme na širší třídu aplikací: Množina všech (reálných) tříprvkových vektorů tvoří

lineární prostor $\mathbf{R^3}$. Rovina ρ procházející počátkem představuje lineární podprostor $\mathbf{L} \subseteq \mathbf{R^3}$. Dimenze podprostoru $\mathbf{L}$ je 2. Prvky podprostoru $\mathbf{L}$ jsou vektory (v naší geometrické interpretaci průvodiče všech bodů ležících v rovině ρ). K podprostoru $\mathbf{L}$ existuje ortogonální doplněk $\bar{\mathbf{L}}$ dimenze 1. Do $\bar{\mathbf{L}}$ patří všechny takové vektory, jež jsou ortogonální na (všechny) prvky podprostoru $\mathbf{L}$ (v naší geometrické interpretaci do $\bar{\mathbf{L}}$ patří všechny vektory kolmé na rovinu ρ). Normálový vektor $\vec{n}$ je pak bází ortogonálního doplňku $\bar{\mathbf{L}}$.

Z lineární algebry víme, že vektor $\boldsymbol{a}$ je prvkem lineárního podprostoru $\mathbf{L}$ právě tehdy, je-li ortogonální na všechny prvky ortogonálního doplňku $\bar{\mathbf{L}}$, což je právě tehdy, je-li ortogonální na všechny prvky báze prostoru $\bar{\mathbf{L}}$ (a přesně to v naší geometrické interpretaci vyjadřuje skalární součin $\vec{a} \cdot \vec{n}$).

Závěr: Pokud bychom dokázali konstruovat množiny kódových značek jako lineární prostory, mohli bychom rozhodovat o tom, zda je přijatá n–tice kódovou značkou, na základě „výpočetního kritéria“.

Konec motivačního příkladu

Lineární prostory jsou konstruovány nad číselnými tělesy. Prvním krokem ke kódům, které budou lineárními prostory, musí být „přeměna“ kódové abecedy T v těleso. Znamená to nadefinovat nad prvky abecedy T operace + (sečítání) a $\cdot$ (násobení), které budou mít **vlastnosti tělesa** (připomenutí látky z předmětu KMA/LA):

$\forall\, a, b \in T$: $\exists\, a + b \in T$ (ke každým dvěma prvkům existuje součet, je určen jednoznačně)
$\forall\, a, b \in T$: $\exists\, a \cdot b \in T$ (ke každým dvěma prvkům existuje součin, je určen jednoznačně)
$\forall\, a, b, c \in T$: $(a + b) + c = a + (b + c)$ (asociativnost operace sečítání)
$\forall\, a, b, c \in T$: $(a \cdot b) \cdot c = a \cdot (b \cdot c)$ (asociativnost operace násobení)
$\forall\, a, b \in T$: $(a + b) = (b + a)$ (komutativnost operace sečítání)
$\forall\, a, b \in T$: $(a \cdot b) = (b \cdot a)$ (komutativnost operace násobení)
$\forall\, a, b, c \in T$: $a \cdot (b + c) = a \cdot b + a \cdot c$ (distributivnost násobení vzhledem ke sčítání)
$\exists\, 0 \in T\ \forall\, a \in T$: $a + 0 = a$ (0 = neutrální prvek vzhledem k sečítání; je určen jednoznačně)
$\exists\, 1 \in T\ \forall\, a \in T$: $a \cdot 1 = a$ (1 = neutrální prvek vzhledem k násobení; určen jednoznačně)
$\forall\, a \in T\ \exists - a \in T$: $a + (-a) = 0$ (opačné prvky; jsou určeny jednoznačně)
$\forall\, a \in T,\ a \neq 0\ \exists\, a^{-1} \in T$: $a \cdot a^{-1} = 1$ (inverzní prvky; jsou určeny jednoznačně)

V případě binárních kódů vytvoříme těleso zavedením těchto operací:

+	0	1
0	0	1
1	1	0

·	0	1
0	0	0
1	0	1

Opačným prvkem k 0 je tedy 0, opačným prvkem k 1 je 1, inverzním prvkem k 1 je 1.

Množina $\{0,1\}$ s takto definovanými operacemi sečítání a násobení bývá označována jako *těleso* Z_2 .

Abychom mohli množinu kódových značek konstruovat jako lineární prostor, musíme nad n–ticemi z $L \subseteq T^n$ nadefinovat operaci $\oplus$ (sečítání prvků z T^n) a operaci $\otimes$ (násobení prvku z $L \subseteq T^n$ skalárem z tělesa T), které budou splňovat podmínky kladené na **lineární prostor** (další připomenutí):

$\forall\, \boldsymbol{a}, \boldsymbol{b} \in \mathbf{L}$: $\exists\, \boldsymbol{a} \oplus \boldsymbol{b} \in \mathbf{L}$ (ke každým dvěma prvkům v $\mathbf{L}$ existuje součet; a to jednoznačně)
$\forall\, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in \mathbf{L}$: $(\boldsymbol{a} \oplus \boldsymbol{b}) \oplus \boldsymbol{c} = \boldsymbol{a} \oplus (\boldsymbol{b} \oplus \boldsymbol{c})$ (asociativnost operace sečítání)
$\forall\, \boldsymbol{a}, \boldsymbol{b} \in \mathbf{L}$: $(\boldsymbol{a} \oplus \boldsymbol{b}) = (\boldsymbol{b} \oplus \boldsymbol{a})$ (komutativnost operace sečítání)
$\exists\, \mathbf{0} \in \mathbf{L}\ \forall\, \boldsymbol{a} \in \mathbf{L}$: $\boldsymbol{a} \oplus \mathbf{0} = \boldsymbol{a}$ ($\mathbf{0}$ = neutrální prvek vzhledem k sečítání; určen jednoznačně)
$\forall\, \boldsymbol{a} \in \mathbf{L}\ \exists - \boldsymbol{a} \in \mathbf{L}$: $\boldsymbol{a} \oplus (-\boldsymbol{a}) = \mathbf{0}$ (opačné prvky; jsou určeny jednoznačně)

$\forall\, \boldsymbol{a} \in \mathbf{L}\ \forall\, t \in T$: $\exists\, t \otimes \boldsymbol{a} \in \mathbf{L}$ (násobení vektoru skalárem, součin je určen jednoznačně)
$\forall\, \boldsymbol{a}, \boldsymbol{b} \in \mathbf{L}\ \forall\, s, t \in T$: $t \otimes (\boldsymbol{a} \oplus \boldsymbol{b}) = t \otimes \boldsymbol{a} \oplus t \otimes \boldsymbol{b}$
$\forall\, \boldsymbol{a} \in \mathbf{L}\ \forall\, s, t \in T$: $(s \cdot t) \otimes \boldsymbol{a} = s \otimes (t \otimes \boldsymbol{a})$
$\forall\, \boldsymbol{a} \in \mathbf{L}\ \forall\, s, t \in T$: $(s + t) \otimes \boldsymbol{a} = s \otimes \boldsymbol{a} \oplus t \otimes \boldsymbol{a}$
$\forall\, \boldsymbol{a} \in \mathbf{L}$: $1 \otimes \boldsymbol{a} = \boldsymbol{a}$, kde 1 je jednotkový prvek tělesa T
$\forall\, \boldsymbol{a} \in \mathbf{L}$: $0 \otimes \boldsymbol{a} = \mathbf{0}$, kde 0 je nulový prvek tělesa T

Je zřejmé, že v případě součtu dvou n–tic vznikne výsledná n–tice součty „bit po bitu“, čili $\boldsymbol{c} = \boldsymbol{a} \oplus \boldsymbol{b} \quad \Leftrightarrow\ \forall\, i = 1,2, \ldots, n$: $c_i = a_i + b_i$, kde $+$ je operace sečítání nad tělesem T.

Poznámka 1: V dalším textu již nahradíme označení operátorů $\oplus$ a $\otimes$ klasickým $+$ a $\cdot$, protože z kontextu bude vždy zřejmé, zda sečítáme značky nebo čísla (tj. vektory nebo skaláry).

Poznámka 2: U množiny kódových značek přejdeme od označení K k symbolu $\mathbf{K}$, protože kódové značky již nebudou tvořit pouhou množinu, ale lineární prostor. Analogicky místo T^n budeme používat $\mathbf{T^n}$

Po zavedení operace sečítání nad značkami můžeme začít popisovat kódy rovnicemi.

Ilustrační příklad: Paritní kód a opakovací kód.

Vrátíme se ke kódu 2 z kapitoly 2.3.2 (používané názvy - paritní kód nebo také kód celkové kontroly parity). Tento kód kóduje hexadecimální cifry do binárních pětic takovým způsobem, že první čtyři binární znaky odpovídají „standardní reprezentaci“ hexadecimální cifry váhovým kódem 8421, pátý (kontrolní) znak je pak zvolen tak, aby celkový počet jedniček ve značce byl sudý. Připomeňme několik kódových značek: 01111 (7) , 10100 (A) , 11101 (E).

Protože je ve značce sudý počet jedniček, lze je „posčítat po dvou“. V tělese Z_2 platí $1 + 1 = 0$, každá kódová značka $\boldsymbol{v} \in \mathbf{K}$ tedy splňuje podmínku $v_1 + v_2 + v_3 + v_4 + v_5 = 0$.

Dále uvažujme binární opakovací kód délky 5 . Kód obsahuje dvě kódové značky – 00000 a 11111. Protože jsou všechny znaky ve značce opakováním znaku v_1 , splňuje každá kódová značka $\boldsymbol{v} \in \mathbf{K}$ podmínky $v_1 = v_2$, $v_1 = v_3$, $v_1 = v_4$, $v_1 = v_5$. Tyto rovnice můžeme přičtením pravých stran rovnic k oběma stranám rovnic převést na homogenní rovnice

s nulovou pravou stranou: $v_1 + v_2 = 0$, $v_1 + v_3 = 0$, $v_1 + v_4 = 0$, $v_1 + v_5 = 0$. (protože v tělese Z_2 platí 0 + 0 = 0 a 1 + 1 = 0, platí také $v_i + v_i = 0$).

Jak paritní, tak i opakovací kód jsme popsali soustavami homogenních lineárních rovnic – paritní kód jednou rovnicí, opakovací kód čtyřmi rovnicemi. Tyto rovnice budeme nazývat *kontrolními rovnicemi*. Je zřejmé, že opakovací kód délky n by byl popsán $n - 1$ rovnicemi; paritní kód se značkou délky n by měl stále jen jednu kontrolní rovnici. Počet kontrolních rovnic tedy odpovídá počtu znaků, které jsme do značky ke znakům, jež „nesou informaci" (k *informačním znakům*) přidali pro dosažení zabezpečovacích vlastností (počet kontrolních rovnic tedy odpovídá počtu *kontrolních znaků*).

Kódové značky obou kódů jsou tedy řešením příslušných kontrolních rovnic (jedné rovnice v případě paritního kódu a soustavy rovnic v případě opakovacího kódu).

Připomenutí důležitého poznatku z lineární algebry: **Řešení soustavy homogenních lineárních rovnic o n proměnných tvoří lineární prostor, jenž je podprostorem lineárního prostoru $\mathbf{T}^n$.**

Množiny značek paritního kódu a opakovacího kódu tedy tvoří lineární prostory, patří mezi *lineární kódy*. Dimenze prostoru je dána počtem prvků, které ve značce můžeme libovolně volit (tedy těch, které „nesou informaci"). Dimenze paritního kódu délky 5 je tedy 4, dimenze opakovacího kódu bez ohledu na délku kódu je 1. n–tice z $\mathbf{T^n}$ je kódovou značkou právě tehdy, pokud vyhovuje soustavě kontrolních rovnic.

(Přerušení ilustračního příkladu)

Binární kód K je *lineárním kódem* $\mathbf{K}$, jestliže je podprostorem lineárního prostoru $\mathbf{Z_2}^n$. Dimenzi k lineárního kódu nazýváme *počtem informačních znaků*. Pro lineární kódy s k informačními znaky pak používáme název *lineární* (n, k) *kód*.

Důsledky:

Součet libovolných dvou kódových značek lineárního kódu je také kódová značka.
Skalární násobek libovolné kódové značky lineárního kódu je také kódová značka (u binárních kódů je tento důsledek nezajímavý).
Součástí každého lineárního kódu je nulová značka 00….0 .

V ilustračním příkladu s paritním a opakovacím kódem jsme použili termíny *informační znaky* a *kontrolní (zabezpečovací) znaky*. U těchto kódů je rozdělení znaků na informační a kontrolní zřejmé: paritní kód délky n má prvních $n - 1$ znaků informačních, poslední, tj. $n -$ tý znak je zabezpečovací; opakovací kód délky n má první znak informační, dalších $n - 1$ znaků je zabezpečovacích. Jsme tedy schopni informační a zabezpečovací znaky rozlišit.

POZOR! Obecně tomu tak být nemusí. Tvrzení „kód má k informačních znaků" totiž nemusí znamenat, že ve značce dokážeme rozlišit, které znaky jsou informační a které zabezpečovací. Pojem „k informačních znaků" vyjadřuje něco obecnějšího:

Blokový kód $\boldsymbol{K} \subseteq \boldsymbol{T^n}$ délky n má k *informačních znaků* (a $n-k$ kontrolních znaků) právě tehdy, jestliže existuje prosté zobrazení φ množiny všech slov délky k na množinu kódových značek $\boldsymbol{K}$, tedy φ: $\boldsymbol{T^k} \rightarrow \boldsymbol{K}$.

Používaná terminologie a značení:

φ: $\boldsymbol{T^k} \rightarrow \boldsymbol{K}$ nazýváme *kódování informačních znaků*
$u \in \boldsymbol{T^k}$ nazýváme *informační část*
$v \in \boldsymbol{K}$, $v = \varphi(u)$ je *kódová značka*
(n,k) kód je kód s k informačními a $n-k$ kontrolními znaky

Lze dokázat *souvislost mezi minimální Hammingovou vzdáleností kódu* $d_0(K)$ *a počtem kontrolních znaků* $(n-k)$ ve značkách: $d_0(K) \leq n-k+1$.

Poznámka: Pojmy informační znaky, kontrolní znaky, (n,k) kód jsou obecné, týkají se i kódů, které nejsou lineární.

Paritní kód délky n je $(n,n-1)$ kód, opakovací kód délky n je $(n,1)$ kód. Později (např. v kapitolách 2.3.6 a 2.3.7) uvidíme, že existují kódy, u nichž některé informační znaky nejsou ve značce obsaženy „explicitně“, ale vyskytují se tam pouze ve formě lineárních kombinací s jinými informačními znaky.

Terminologická poznámka k dekódování: V kapitole 2.3.2 jsme zavedli pojem dekódovací funkce jako zobrazení $\delta : T^n \rightarrow K$ definované tak, že umožňuje opravy $t-$ násobných chyb. Termíny *dekódování*, respektive *dekódovací funkce* bývají používány i pro funkci φ^{-1}: $K \rightarrow T^k$, která z kódové značky „extrahuje“ informační část. Zpracování přijaté $n-$ tice tedy lze (v ideálním případě, kdy neexistují „neopravitelné“ $n-$ tice) popsat takto:

$$u' = \varphi^{-1}\big(\delta(w)\big) \text{ , kde}$$

$w \in T^n$ je přijatá $n-$ tice (ovlivněná šumem)
$\delta(w) \in K$ je přijatá $n-$ tice po opravě chyby (tedy kódová značka)
$u' \in T^k$ je informační část extrahovaná z přijaté $n-$ tice po opravě chyby

Připomenutí dalších důležitých poznatků o lineárních prostorech:

Každý lineární prostor je jednoznačně určen svojí bází.
Každý prvek lineárního prostoru je v dané bázi jednoznačně určen svými souřadnicemi.

V případě lineárních kódů dimenze k bude bázi tvořit k lineárně nezávislých značek a každá kódová značka bude jednoznačně určena svými souřadnicemi, tedy $k-$ prvkovým vektorem, který představuje informační znaky, jež jsou ve značce zakódovány. Kódování informační části φ má pak u lineárních kódů tvar lineární kombinace bázových prvků, kde jako koeficienty kombinace použijeme informační znaky: $\boldsymbol{v} = \varphi(u) = u_1 \cdot \boldsymbol{b_1} + u_2 \cdot \boldsymbol{b_2} + \cdots + u_k \cdot \boldsymbol{b_k}$.

Při kódování informační části i při kontrole přijaté $n-$tice v lineárních kódech budeme pracovat s vektory a maticemi. Používané značení vektorů a jejich rozměry:

$\boldsymbol{u}$ typu $k/1$ (sloupeček o k prvcích) je informační část (vektor informačních znaků)

$\boldsymbol{v}$ typu $n/1$ (sloupeček o n prvcích) je kódová značka (tj. zakódovaná informační část)
$\boldsymbol{e}$ typu $n/1$ (sloupeček o n prvcích) je chybový vektor
$\boldsymbol{w}$ typu $n/1$ (sloupeček o n prvcích) je přijatá n −tice ($\boldsymbol{w} = \boldsymbol{v} + \boldsymbol{e}$)
$\boldsymbol{s}$ typu $(n-k)/1$ (sloupeček o $n-k$ prvcích) je *syndrom* (výsledek kontroly přijaté n −tice)

Vektory budeme obvykle zapisovat ve formě transponovaných řádků, např. $\boldsymbol{u} = [001110]^{\boldsymbol{T}}$ nebo $\boldsymbol{v} = [v_{\boldsymbol{1}} v_{\boldsymbol{2}} \ldots\ldots v_{\boldsymbol{n}}]^{\boldsymbol{T}}$.

Každá kódová značka lineárního (n, k) kódu $\mathbf{K}$ musí být řešením soustavy $n-k$ kontrolních rovnic. Matici této soustavy ve tvaru s nulovou pravou stranou nazýváme *kontrolní matice* a označujeme ji $\boldsymbol{H}$. Kontrolní matice $\boldsymbol{H}$ je typu $(n-k)/n$ a její řádky jsou bázovými prvky ortogonálního doplňku k lineárnímu kódu $\mathbf{K}$.

Vlastnosti kontrolní matice:

- Řádky kontrolní matice jsou lineárně nezávislé
- Slovo $\boldsymbol{w}$ je kódovou značkou právě tehdy, když jeho součin s kontrolní maticí $\boldsymbol{H} \cdot \boldsymbol{w}$ je nulový

Výsledkem kontroly přijaté značky kontrolní maticí je syndrom $\boldsymbol{s} = \boldsymbol{H} \cdot \boldsymbol{w}$, který je nulový právě tehy, když $\boldsymbol{w} \in \mathbf{K}$. Formálně $\boldsymbol{w} \in \mathbf{K} \Leftrightarrow \boldsymbol{H} \cdot \boldsymbol{w} = \mathbf{0}$, kde symbol **0** představuje „vektorovou nulu“, tj. jeden sloupec o $(n-k)$ řádcích.

Uspořádáme-li prvky báze lineárního kódu do matice tak, že každý bázový prvek bude řádkem této matice, vytvoříme tak *generující matici* $\boldsymbol{G}$ typu k/n :

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{b_1} \\ \vdots \\ \boldsymbol{b_k} \end{bmatrix} = \begin{bmatrix} b_{11} & \ldots & b_{1n} \\ \vdots & \ddots & \vdots \\ b_{k1} & \vdots & b_{kn} \end{bmatrix}$$

Potom lze kódování informační části $\boldsymbol{v} = u_1 \cdot \boldsymbol{b_1} + u_2 \cdot \boldsymbol{b_2} + \cdots + u_k \cdot \boldsymbol{b_k}$ vyjádřit ve formě maticového násobení $\boldsymbol{v} = \boldsymbol{G}^{\boldsymbol{T}} \cdot \boldsymbol{u}$.
Vlastnosti generující matice:

- Každý řádek generující matice je kódovou značkou
- Řádky generující matice jsou lineárně nezávislé
- Každé kódové slovo je lineární kombinací řádků generující matice a je jednoznačně určeno informačními znaky

Vztah mezi kontrolní maticí $\boldsymbol{H}$ a generující maticí $\boldsymbol{G}$

Matice jsou maticemi bázových prvků dvou lineárních prostorů, které jsou navzájem ortogonální ($\boldsymbol{G}$ – báze kódu**,** $\boldsymbol{H}$ – báze jeho ortogonálního doplňku). Každý řádek matice $\boldsymbol{H}$ je tedy ortogonální na každý řádek matice $\boldsymbol{G}$ **.** Vyjádřeme-li toto maticově, platí $\boldsymbol{H} \cdot \boldsymbol{G}^{\boldsymbol{T}} = \mathbf{0}$, kde symbol $\mathbf{0}$ představuje nulovou matici typu $(n-k)/k$.

Položme si nyní otázku: Jak k zadané generující matici $\boldsymbol{G}$ nalézt kontrolní matici $\boldsymbol{H}$?

Budeme hledat obecný řádek matice $\boldsymbol{H}$ ve tvaru $\boldsymbol{h} = [h_1 h_2 \dots \dots . h_n]$. Skalární součin tohoto řádku s libovolným z k řádků matice $\boldsymbol{G}$ musí být nulový, získáme tak k homogenních rovnic pro n neznámých. Řešením této soustavy je lineární prostor dimenze $(n-k)$, jinak řečeno „$(n-k)$-parametrické řešení“. Vhodnou (tj. lineárně nezávislou) volbou $(n-k)$ parametrů získáme prvky báze, které pak umístíme do řádků matice $\boldsymbol{H}$ **.** Z uvedeného je zřejmé, že kontrolní matice existuje vždy, není ale určena jednoznačně, tj. k jedné generující matici může existovat více kontrolních matic.

(Návrat k ilustračnímu příkladu Binární paritní kód a binární opakovací kód):

Tělesem $\mathbf{T}$ **,** nad nímž je lineární kód konstruován, je těleso Z_2 . Zkonstruujeme bázi paritního kódu délky 4. Přirozenou cestou je vytvořit ji zakódováním kanonické báze lineárního prostoru $\mathbf{T}^k$, tedy prostoru všech informačních částí. Do kanonické báze prostoru $\mathbf{T}^k$ patří všechny k −tice, které obsahují právě jednu jedničku. Kódová značka paritního kódu musí obsahovat sudý počet jedniček, každá značka báze paritního kódu tedy bude mít právě jednu jedničku v informačních prvcích a jedničku v kontrolním prvku:

$\boldsymbol{b_1} = [10001]$, $\boldsymbol{b_2} = [01001]$, $\boldsymbol{b_3} = [00101]$, $\boldsymbol{b_4} = [00011]$.

Ukázka zakódování informační části $\boldsymbol{u} = [1010]$ prostým vytvořením lineární kombincie bázových prvků:

$$\boldsymbol{v} = u_1 \cdot \boldsymbol{b_1} + u_2 \cdot \boldsymbol{b_2} + u_3 \cdot \boldsymbol{b_3} + u_4 \cdot \boldsymbol{b_4} =$$
$$= 1 \cdot [10001] + 0 \cdot [01001] + 1 \cdot [00101] + 0 \cdot [00011] = [10100]$$

Uspořádání bázových prvků do generující matice:

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{b_1} \\ \boldsymbol{b_2} \\ \boldsymbol{b_3} \\ \boldsymbol{b_4} \end{bmatrix} = \begin{bmatrix} b_{11}\, b_{12}\, b_{13}\, b_{14}\, b_{15} \\ b_{21}\, b_{22}\, b_{23}\, b_{24}\, b_{25} \\ b_{31}\, b_{32}\, b_{33}\, b_{34}\, b_{35} \\ b_{41}\, b_{42}\, b_{43}\, b_{44}\, b_{45} \end{bmatrix} = \begin{bmatrix} 10001 \\ 01001 \\ 00101 \\ 00011 \end{bmatrix}$$

Ukázka zakódování informační části $\boldsymbol{u} = [1010]$ maticovým násobením $\boldsymbol{v} = \boldsymbol{G^T} \cdot \boldsymbol{u}$:

$$\boldsymbol{v} = \boldsymbol{G^T} \cdot \boldsymbol{u} = \begin{bmatrix} 1000 \\ 0100 \\ 0010 \\ 0001 \\ 1111 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

Praktický postup:

		Vybrané řádky
$\begin{bmatrix} 10001 \\ 01001 \\ 00101 \\ 00011 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$	10001 00101
$\boldsymbol{G}$	$\boldsymbol{u}$	$\overline{[10100]}\ \boldsymbol{v}$

Výpočet $\boldsymbol{v} = \boldsymbol{G}^{\boldsymbol{T}} \cdot \boldsymbol{u}$ se nejsnáze provede tak, že se za generující matici $\boldsymbol{G}$ zapíše informační vektor $\boldsymbol{u}$ jako sloupeček. Pak se sečtou ty řádky matice $\boldsymbol{G}$, u kterých je v informačním vektoru znak 1 (viz levá strana řádku s výpočtem $\boldsymbol{v} = \boldsymbol{G}^{\boldsymbol{T}} \cdot \boldsymbol{u}$).

Kontrolní matice $\boldsymbol{H}$ je maticí soustavy homogenních kontrolních rovnic. Paritní kód délky 5 má jedinou kontrolní rovnici $v_1 + v_2 + v_3 + v_4 + v_5 = 0$, kontrolní matice proto bude $\boldsymbol{H} = [11111]$.

Dále ukážeme kontrolu přijaté značky v případě kódové značky $\boldsymbol{v} = [10100]^T$ a jednoduché chyby ve třetím přenášeném znaku, tj. výpočet syndromu $\boldsymbol{s} = \boldsymbol{H} \cdot \boldsymbol{w}$:

$\boldsymbol{e} = [00100]^{\boldsymbol{T}} \quad \boldsymbol{w} = \boldsymbol{v} + \boldsymbol{e} = [10100]^{\boldsymbol{T}} + [00100]^{\boldsymbol{T}} = [10000]^{\boldsymbol{T}}$

$\boldsymbol{s} = \boldsymbol{H} \cdot \boldsymbol{w} = [11111] \cdot [10000]^{\boldsymbol{T}} = [1]$

Praktický postup:

	$\boldsymbol{w}$	$[10000]^{\boldsymbol{T}}$
$\boldsymbol{H}$	$[11111]$	
Vybrané sloupce	$[1 \quad]$	$\mid [1]$ $\boldsymbol{s}$

Výpočet $\boldsymbol{s} = \boldsymbol{H} \cdot \boldsymbol{w}$ lze prakticky provést podobně jako výpočet $\boldsymbol{v} = \boldsymbol{G}^{\boldsymbol{T}} \cdot \boldsymbol{u}$. Nad kontrolní matici $\boldsymbol{H}$ se zapíše přijatý vektor $\boldsymbol{w}$ jako řádek. Pak se sečtou ty sloupce matice $\boldsymbol{H}$, u kterých je v přijatém vektoru znak 1 (viz levá strana řádku s výpočtem $\boldsymbol{s} = \boldsymbol{H} \cdot \boldsymbol{w}$). Syndrom $\boldsymbol{s}$ není roven nulovému vektoru, z čehož příjemce pozná, že při přenosu značky došlo k chybě.

Stejným způsobem vytvoříme bázi opakovacího kódu délky 5. Kód má jeden informační znak ($k = 1$). Kanonickou bázi prostoru $\mathbf{T}^k$ tedy tvoří jediný prvek, a to $[1]$. Kódová značka opakovacího kódu opakuje první znak ve všech následujících, jediným prvkem báze opakovacího kódu tedy bude značka $\boldsymbol{b_1} = [11111]^T$

Do množiny kódových značek pak patří pouze značky $0 \cdot \boldsymbol{b_1} = 0 \cdot [11111]^T = [00000]^T$ a $1 \cdot \boldsymbol{b_1} = 1 \cdot [11111]^T = [11111]^T$.

Připomenutí kontrolních rovnic : $v_1 + v_2 = 0$, $v_1 + v_3 = 0$, $v_1 + v_4 = 0$, $v_1 + v_5 = 0$
Maticí této soustavy je tedy kontrolní matice $\boldsymbol{H}$.

$$\boldsymbol{H} = \begin{bmatrix} 11000 \\ 10100 \\ 10010 \\ 10001 \end{bmatrix}$$

Ukážeme kontrolu přijaté značky v případě přenášené kódové značky $\boldsymbol{v} = [11111]^T$ a dvojité chybě ve třetím a čtvrtém přenášeném znaku:

$\boldsymbol{e} = [00110]^{\boldsymbol{T}} \quad \boldsymbol{w} = \boldsymbol{v} + \boldsymbol{e} = [11111]^{\boldsymbol{T}} + [00110]^{\boldsymbol{T}} = [11001]^{\boldsymbol{T}}$

$$\boldsymbol{s} = \boldsymbol{H} \cdot \boldsymbol{w} = \begin{bmatrix} 11000 \\ 10100 \\ 10010 \\ 10001 \end{bmatrix} \cdot [11001]^{\boldsymbol{T}} = [0110]^{\boldsymbol{T}} \qquad \begin{matrix} [11001] \\ \begin{bmatrix} 11000 \\ 10100 \\ 10010 \\ 10001 \end{bmatrix} \end{matrix} \quad \boldsymbol{s} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Syndrom $\boldsymbol{s}$ není roven nulovému vektoru, při přenosu značky tedy došlo k chybě.

(Konec ilustračního příkladu)

V dalším textu a příkladech již nebudeme barevně odlišovat kontrolní prvky značek, protože (jak jsem už konstatovali) informační a kontrolní znaky nemusí být „explicitně“ odděleny, v některých kódech mohou některé prvky mít současně obě role. Barevné označení bude dále používáno pouze ke zvýraznění popisovaných souvislostí.

Ilustrační příklad – podrobnější pohled na generující matici:

Binární lineární kód je zadán generující maticí

$$\boldsymbol{G_1} = \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \boldsymbol{b}_3 \\ \boldsymbol{b}_4 \end{bmatrix} = \begin{bmatrix} 1000011 \\ 0100101 \\ 0010110 \\ 0001111 \end{bmatrix}$$

Řádky matice $\boldsymbol{G_1}$ jsou na první pohled lineárně nezávislé, matice je tedy korektní maticí bázových prvků lineárního prostoru dimenze 4. Lineární (7,4) kód $\mathbf{K}$ generovaný maticí $\boldsymbol{G_1}$ je lineárním obalem řádků matice $\boldsymbol{G_1}$:

$$\mathbf{K} = \{\boldsymbol{v} | \boldsymbol{v} = u_1 \cdot \boldsymbol{b}_1 + u_2 \cdot \boldsymbol{b}_2 + u_3 \cdot \boldsymbol{b}_3 + u_4 \cdot \boldsymbol{b}_4, \quad \forall u_1, u_2, u_3, u_4 \in \mathbf{Z_2}\}$$

Z definice sčítání vektorů (v našem případě $v_i = u_1 \cdot b_{i1} + u_2 \cdot b_{i2} + u_3 \cdot b_{i3} + u_4 \cdot b_{i4}$) je zřejmé, že j –tý sloupec generující matice definuje, sečtením kterých informačních prvků vznikne j –tý prvek kódové značky $\boldsymbol{v}$. Jinak řečeno – podle sloupců generující matice lze napsat „vytvořující rovnice“ pro jednotlivé prvky kódové značky $\boldsymbol{v}$. V našem případě

$$v_1 = u_1 \quad , \quad v_2 = \boldsymbol{u_2} \quad , \quad v_3 = u_3 \quad , \quad v_4 = u_4$$
$$v_5 = \boldsymbol{u_2} + \boldsymbol{u_3} + \boldsymbol{u_4} \quad , \quad v_6 = \boldsymbol{u_1} + \boldsymbol{u_3} + \boldsymbol{u_4} \quad , \quad v_7 = \boldsymbol{u_1} + \boldsymbol{u_2} + \boldsymbol{u_4}$$

Pokud je na pravé straně „vytvořující rovnice“ prvku v_j jediný prvek u_i, je zřejmé, že je v prvku v_j „čistě“ přenášen informační prvek u_i. Pokud je na pravé straně součet více informačních prvků, je přenášena lineární kombinace informačních prvků.

Uvažujme jiný lineární (7,4) kód generovaný maticí $\boldsymbol{G_2}$:

$$\boldsymbol{G_2} = \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \boldsymbol{b}_3 \\ \boldsymbol{b}_4 \end{bmatrix} = \begin{bmatrix} 1101000 \\ 0110100 \\ 0011010 \\ 0001101 \end{bmatrix}$$

I u této matice jsou řádky zjevně lineárně nezávislé, matice je tedy korektní generující maticí lineárního kódu. Podle sloupců matice $\boldsymbol{G_2}$ napíšeme „vytvořující rovnice“ vektoru $\boldsymbol{v}$:

$$v_1 = u_1 \quad , \quad v_2 = \boldsymbol{u_1} + \boldsymbol{u_2} \quad , \quad v_3 = u_2 + u_3 \quad , \quad v_4 = \boldsymbol{u_1} + \boldsymbol{u_3} + \boldsymbol{u_4}$$
$$v_5 = \boldsymbol{u_2} + \boldsymbol{u_4} \quad , \quad v_6 = u_3 \quad , \quad v_7 = \boldsymbol{u_4}$$

Z rovnic je zřejmé, že v prvcích v_1, v_6, v_7 jsou přenášeny informační znaky „přímo“, je ale vidět i to, že informační znak u_2 není ve značce obsažen „v čisté podobě“, pouze je přenášen ve třech lineárních kombinacích. Přesto ovšem říkáme, že tento kód má čtyři informační znaky, protože je lze jednoznačně zakódovat (existuje zobrazení $\mathbf{T}^4 \to \mathbf{K}$). Např. informační části $\boldsymbol{u} = [1110]$ bude odpovídat značka :

$$\boldsymbol{v} = \cdot\, \boldsymbol{G_2^T} \cdot \boldsymbol{u} = \begin{bmatrix}1101000\\0110100\\0011010\\0001101\end{bmatrix}\begin{bmatrix}\mathbf{1}\\\mathbf{1}\\\mathbf{1}\\0\end{bmatrix} \quad \begin{matrix}1101000\\0110100\\0011010\end{matrix}$$

$$\boldsymbol{v} = \overline{[\mathbf{1000110}]^T}$$

Tento kód je příkladem cyklického kódu (viz 2.3.7), dekódováním značek se v tuto chvíli zabývat nebudeme.

(Konec ilustračního příkladu)

Velký praktický význam mají *systematické kódy*, což jsou kódy, kde jsou informační znaky umístěny na začátku značky při zachování jejich pořadí, platí tedy $v_1 = u_1$, $v_2 = u_2$, ……., $v_k = u_k$. Generující matice systematického kódu má tedy tvar

$$\boldsymbol{G} = \begin{bmatrix}10 \ldots . 0 & b_{11}b_{12} \ldots . . b_{1(n-k)}\\01 \ldots . 0 & b_{21}b_{22} \ldots . . b_{2(n-k)}\\ & \ldots \ldots \ldots .\\ & \ldots \ldots \ldots\\00 \ldots . 1 & b_{k1}b_{k2} \ldots . . b_{k(n-k)}\end{bmatrix} = [\boldsymbol{I_k} \mid \boldsymbol{B}] \quad ,$$

kde $\boldsymbol{I_k}$ je jednotková matice řádu k a matice $\boldsymbol{B}$ typu $k/(n-k)$ představuje blok, který generuje kontrolní prvky tak, že pro $i > k$ platí $v_i = u_1 \cdot b_{1i} + u_2 \cdot b_{2i} + \cdots + u_k \cdot b_{ki}$.

Výhodou systematických kódů je to, že výpočet kontrolních prvků může probíhat paralelně s vysíláním informační části, a snadná realizace zobrazení φ^{-1} (extrakce informační části = odříznutí kontrolních prvků). Lze dokázat, že k systematickému kódu s generující maticí ve tvaru $\boldsymbol{G} = [\boldsymbol{I_k} \mid \boldsymbol{B}]$ je jednou z kontrolních matic matice ve tvaru $\boldsymbol{H} = [-\boldsymbol{B^T} \mid \boldsymbol{I_{n-k}}]$. Tento tvar je u systematických kódů považován za standardní tvar kontrolní matice.

Poznámka: Pro binární kódy nad tělesem Z_2 je znaménko $-$ u matice $\boldsymbol{B^T}$ nadbytečné (každý prvek tělesa Z_2 je opačný sám k sobě) nicméně tento tvar kontrolní matice je obecný pro lineární kódy nad jakýmkoli konečným tělesem, a pro ta už tvrzení v závorce neplatí.

V textu kapitoly 2.3.2 jsme definovali minimální Hammingovu vzdálenost kódu jako $d_0(K) = \min_{u,v \in K,\ u \neq v} d(u,v)$, tedy jako vzdálenost dvou „nejbližších“ kódových značek. Pokud bychom měli z množiny kódových značek zjistit minimální Hammingovu vzdálenost, stačil by na to jednoduchý algoritmus, který by dvojitým cyklem procházel všechny navzájem různé dvojice značek a jejich vzdálenost by porovnával s dosud nalezenou minimální vzdáleností.

Pokud bychom věděli, že kód je lineární, mohl by díky linearitě být algoritmus ještě jednodušší:

Předpokládejme, že u a v jsou právě ty dvě kódové značky, které určují minimální vzdálenost $d_0(K)$. Protože je kód lineární, jsou kodovými značkami i „opačná" značka $-u$ a „součtová značka" $v + (-u) = v - u$. Platí

$$d_0 = d(u,v) = d(u-u, v-u) = d(0, v-u) = \|v-u\| \ , \quad \text{kde}$$

$\|v-u\|$ představuje počet nenulových prvků ve značce $v-u$, tj. *Hammingovu váhu značky* $v-u$

Ilustrace platnosti vztahu $d(u,v) = d(u+x, v+x)$:

$u = [110101]^T$ $\quad x = [101110]^T$ $\quad u+x = [011011]^T$
$v = [010110]^T$ $\quad v+x = [111000]^T$
Rozdíly: ! ! ! ! ! !
$d(u,v) = 3$ $\quad d(u+x, v+x) = 3$

Vidíme, že přičteme-li ke dvěma značkám stejnou n – tici, liší se „součtové značky" stále ve stejných pozicích, Hammnigova vzdálenost tedy zůstává stejná.

Závěr: Minimální Hammingova vzdálenost lineárního kódu je rovna minimální váze nenulové značky.

K nalezení minimální Hammingovy vzdálenosti z množiny kódových značek by tedy stačil jednoduchý cyklus, který by prošel všechny nenulové kódové značky a jejich váhu by porovnával s dosud nalezenou minimální vahou.

Opravování chyb pomocí syndromu

Při kontrole přijaté n–tice nás zatím zajímalo pouze to, zda je syndrom $\boldsymbol{s} = \boldsymbol{H} \cdot \boldsymbol{w}$ nenulový, což by znamenalo, že při přenosu došlo k chybě. V následujících úvahách budeme zkoumat, zdy by nebylo možné využít hodnoty syndromu k lokalizaci (a tedy i k následné opravě) chyby. Přijatá n–tice vznikne z vysílané kódové značky přičtením chybového vektoru (ten představuje abstraktní popis konkrétního projevu šumu). V důsledku linearity kódu pak platí:

$$\boldsymbol{s} = \boldsymbol{H} \cdot \boldsymbol{w} = \boldsymbol{H} \cdot (\boldsymbol{v} + \boldsymbol{e}) = \boldsymbol{H} \cdot \boldsymbol{v} + \boldsymbol{H} \cdot \boldsymbol{e} = \boldsymbol{0} + \boldsymbol{H} \cdot \boldsymbol{e} = \boldsymbol{H} \cdot \boldsymbol{e}$$

Součin $\boldsymbol{H} \cdot \boldsymbol{v}$ je nulový, protože $\boldsymbol{v}$ je kódovou značkou $(\boldsymbol{v} \in \mathbf{K} \Leftrightarrow \boldsymbol{H} \cdot \boldsymbol{v} = \boldsymbol{0})$.

Závěry: U lineárních kódů syndrom závisí pouze na chybovém vektoru a nezávisí na přenášené kódové značce.

Lineární kód není schopen odhalit chyby s chybovým vektorem ve tvaru kódové značky.

První závěr nabízí lákavou myšlenku - podle syndromu určit chybový vektor. Budeme násobení chybového vektoru $\boldsymbol{e}$ maticí $\boldsymbol{H}$ chápat jako zobrazení ω takové, že $\omega(\boldsymbol{e}) = \boldsymbol{H} \cdot \boldsymbol{e}$. Definičním oborem zobrazení ω je pak množina všech chybových vektorů o n prvcích, oborem

hodnot množina všech syndromů o $n-k$ prvcích, tedy $\omega : \mathbf{T}^n \rightarrow \mathbf{T}^{n-k}$. Chceme-li ze syndromu „zpětně“ určit chybový vektor, potřebujeme k tomu inverzní zobrazení ω^{-1} . Uvědomíme-li si ovšem, že definiční obor má v případě binárního kódu 2^n prvků, obor hodnot 2^{n-k} prvků, vidíme, že má definiční obor více prvků než obor hodnot, zobrazení tedy není injektivní a inverzní zobrazení ω^{-1} neexistuje.

Ilustrační příklad :

Binární lineární kód je zadán kontrolní maticí

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

Spočítáme syndromy pro různé chybové vektory – nejprve pro všechny jednoduché chyby:

$$\begin{array}{ll} \boldsymbol{e} = [1000000]^{\boldsymbol{T}} & \boldsymbol{s} = [001]^{\boldsymbol{T}} \\ \boldsymbol{e} = [0100000]^{\boldsymbol{T}} & \boldsymbol{s} = [010]^{\boldsymbol{T}} \\ \boldsymbol{e} = [0010000]^{\boldsymbol{T}} & \boldsymbol{s} = [011]^{\boldsymbol{T}} \\ \boldsymbol{e} = [0001000]^{\boldsymbol{T}} & \boldsymbol{s} = [100]^{\boldsymbol{T}} \\ \boldsymbol{e} = [0000100]^{\boldsymbol{T}} & \boldsymbol{s} = [101]^{\boldsymbol{T}} \\ \boldsymbol{e} = [0000010]^{\boldsymbol{T}} & \boldsymbol{s} = [110]^{\boldsymbol{T}} \\ \boldsymbol{e} = [0000001]^{\boldsymbol{T}} & \boldsymbol{s} = [111]^{\boldsymbol{T}} \end{array}$$

Budeme pokračovat syndromy dvojitých chyb:

$$\begin{array}{ll} \boldsymbol{e} = [1100000]^{\boldsymbol{T}} & \boldsymbol{s} = [011]^{\boldsymbol{T}} \\ \boldsymbol{e} = [1010000]^{\boldsymbol{T}} & \boldsymbol{s} = [010]^{\boldsymbol{T}} \\ \boldsymbol{\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots} & \\ \boldsymbol{\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots} & \end{array}$$

Vidíme, že syndromy, které jsou generovány jednoduchými chybami, jsou jedinečné a tvoří množinu všech nenulových syndromů (**POZOR!** Není to obecná vlastnost lineárních kódů, je to dáno konkrétní kontrolní maticí ze zadání). Dvojité chyby (ani žádné chyby vyšší násobnosti) tedy už nemohou vygenerovat žádný „nový“ syndrom.

Pokud bychom se v předpokladech omezili jen na jednoduché chyby s tím, že chyby s vyšší násobností nemohu nastat, bylo by možné ze syndromu jednoznačně určit chybový vektor.

(Přerušení Ilustračního příkladu)

Na základě ilustračního příkladu můžeme konstatovat, že zobrazení ω jednoznačně definuje rozklad množiny všech chybových vektorů na $n-k$ tříd. Každému syndromu odpovídá jedna třída, do níž patří všechny chybové vektory, které tento syndrom generují. Z každé třídy vybereme chybový vektor $\widehat{\boldsymbol{e}_j}$ s nejmenší Hammingovou vahou (budeme jej nazývat *reprezentant chybové třídy*). Vzhledem k pravděpodobnostním předpokladům zdůvodněným v kapitole 2.3.2 je reprezentant chybové třídy nejpravděpodobnějším chybovým vektorem.

Za těchto předpokladů lze zobrazení ω^{-1} jednoznačně definovat tak, že syndromu přiřadí reprezentanta příslušné chybové třídy:

Zobrazení ω^{-1}:

Syndrom:	$\boldsymbol{s_1}$	$\boldsymbol{s_2}$	$\boldsymbol{s_3}$	……	$\boldsymbol{s_{n-k}}$
Reprezentant:	$\widehat{\boldsymbol{e_1}}$	$\widehat{\boldsymbol{e_2}}$	$\widehat{\boldsymbol{e_3}}$	……	$\widehat{\boldsymbol{e_{n-k}}}$

Opravu chyby pak provedeme takto:

$$\boldsymbol{s} = \boldsymbol{H} \cdot \boldsymbol{w} = \boldsymbol{s_j} \quad , \quad \hat{\boldsymbol{v}} = \boldsymbol{w} - \omega^{-1}(\boldsymbol{s_j}) = \boldsymbol{w} - \hat{\boldsymbol{e}}_{\boldsymbol{j}} \quad , \text{ kde}$$

$\hat{\boldsymbol{v}}$ je opravená kódová značka (z praktického pohledu nejpravděpodobněji odesílaná kódová značka)

Je zřejmé, že třídou odpovídající nulovému syndromu je množina kódových značek $\mathbf{K}$ a jejím reprezentantem je nulová značka.

(Návrat k Ilustračnímu příkladu)

Je zřejmé, že reprezentanty chybových tříd jsou nulový vektor (pro třídu odpovídající nulovému chybovému vektoru) a všechny chybové vektory s Hammingovou vahou 1.

(Konec Ilustračního příkladu)

Při opravování chyb na základě syndromu musí mít množina všech možných syndromů přinejmenším stejně prvků, jako má množina všech přípustných chybových vektorů.

Uvažujeme-li binární lineární kód, existuje $2^{\mathrm{n-k}}$ různých syndromů.

Chceme-li opravovat jednoduché chyby, potřebujeme k jejich lokalizaci jeden (nulový) syndrom pro nulový chybový vektor (tj. bezchybný přenos) a n dalších různých syndromů pro rozlišení chybových vektorů s vahou 1. Musí tedy platit nerovnost $2^{\mathrm{n-k}} \geq 1 + n$.

Chceme-li opravovat dvojité chyby, potřebujeme jeden syndrom pro bezchybný přenos, n syndromů pro rozlišení chybových vektorů s vahou 1 a $\binom{n}{2}$ dalších různých syndromů pro rozlišení chybových vektorů s vahou 2. Musí tedy platit $2^{\mathrm{n-k}} \geq 1 + n + \binom{n}{2}$.

Analogicky můžeme vztah pro počet syndromů (a tedy i počet kontrolních znaků) potřebných pro opravu obecných t –násobných chyb: $2^{\mathrm{n-k}} \geq 1 + n + \binom{n}{2} + \cdots + \binom{n}{t}$.

Výše uvedené vztahy dokladují, že počet kontrolních prvků je determinován počtem informačních prvků a násobností opravovaných chyb.

2.3.4. Hammingovy kódy

Hammingovy kódy jsou kódy, které opravují jednoduché chyby a přitom mají minimální redundanci.

Předpokládejme, že při přenosu kódové značky $\boldsymbol{v}$ došlo k jednoduché chybě reprezentované chybovým vektorem $\boldsymbol{e_i} = [0 \dots 010 \dots 0]^{\boldsymbol{T}}$ s jedničkou v i-tém znaku. Přijatou n-tici $\boldsymbol{w}$ pak lze vyjádřit jako $\boldsymbol{w} = \boldsymbol{v} + \boldsymbol{e_i}$ a platí

$$\boldsymbol{s} = \boldsymbol{H} \cdot \boldsymbol{w} = \boldsymbol{H} \cdot (\boldsymbol{v} + \boldsymbol{e_i}) = \boldsymbol{H} \cdot \boldsymbol{v} + \boldsymbol{H} \cdot \boldsymbol{e_i} = \boldsymbol{0} + \boldsymbol{H} \cdot \boldsymbol{e_i} = \boldsymbol{H}_{*,\boldsymbol{i}} \quad , \text{ kde}$$

$\boldsymbol{H}_{*,\boldsymbol{i}}$ představuje i-tý sloupec kontrolní matice $\boldsymbol{H}$.

V případě jednoduché chyby v i-tém prvku má tedy syndrom hodnotu i-tého sloupce kontrolní matice (násobíme-li matici vektorem s jediným nenulovým prvkem, jenž má hodnotu 1, je výsledkem „vybraná“ řada matice).

Důsledek: Vlastnosti kontrolní matice kódu pro opravu jednoduchých chyb

- nesmí obsahovat nulový cloupec $\boldsymbol{H}_{*,\boldsymbol{i}} \neq \boldsymbol{0} \qquad \forall\, i = 1,2, \dots, n$
- nesmí obsahovat stejné sloupce $\boldsymbol{H}_{*,\boldsymbol{i}} \neq \boldsymbol{H}_{*,\boldsymbol{j}} \quad \forall\, i, j = 1,2, \dots, n, \;\; i \neq j$

Je zřejmé, že v případě nedodržení první podmínky by příjemce nerozlišil chybu v i-tém prvku od bezchybného přenosu, v případě nedodržení druhé podmínky by příjemce nedokázal rozlišit chybu v i-tém prvku od chyby v j-tém prvku.

Hammingův kód je binární kód, jehož kontrolní matice obsahuje ve sloupcích (právě jen) **všechna** nenulová slova dané délky a žádné z nich se neopakuje.

Pro zkrácení zápisu budeme pro počet kontrolních prvků používat symbol r ($r = n - k$). Při daném počtu kontrolních prvků r má tedy kontrolní matice Hammingova kódu $2^{\mathrm{r}} - 1$ (různých) sloupců (2^{r} je počet všech binárních slov délky r, musíme ale odpočítat nulové slovo, které být sloupcem kontrolní matice nemůže). Počet sloupců kontrolní matice je stejný jako počet znaků v kódové značce, tedy $n = 2^{\mathrm{r}} - 1$. Z celkového počtu n znaků ve značce jich ovšem je r kontrolních, informačních znaků tedy bude $k = n - r = 2^{\mathrm{r}} - 1 - r$ (častěji bývá tento vztah uváděn ve tvaru $2^{\mathrm{r}} = k + r + 1$).

Na základě výše uvedených vztahů můžeme začít počítat parametry (n, k) Hammingových kódů (je vidět, že s rostoucím r roste *informační poměr* k/n, tedy efektivita kódu):

r	2^{r}	n	k	(n,k)	k/n
3	8	7	4	(7,4)	0,57143
4	16	15	11	(15,11)	0,73333
5	32	31	26	(31.26)	0,83871
6	64	63	57	(63,57)	0,90476
.	.	.	.	.	.
.	.	.	.	.	atd.

Ilustrační příklad (Hammingův kód $(7,4)$)

S kontrolní maticí Hammingova kódu jsme se již setkali v posledním ilustračním příkladu v kapitole 2.3.3 :

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

Na pořadí sloupců v kontrolní matici nezáleží. Výše uvedená kontrolní matice je maticí nesystematického kódu, který má tu vlastnost, že v případě jednoduché chyby je syndrom binárním vyjádřením čísla sloupce, v němž chyby vznikla. Častěji se s Hammingovým kódem $(7,4)$ potkáváme jako s kódem systematickým s maticemi ve tvaru

$$\boldsymbol{G} = \begin{bmatrix} 1000\,011 \\ 0100\,101 \\ 0010\,110 \\ 0001\,111 \end{bmatrix} \qquad \boldsymbol{H} = \begin{bmatrix} 0111\,100 \\ 1011\,010 \\ 1101\,001 \end{bmatrix}$$

respektive v podobě *cyklického Hammingova kódu* $(7,4)$ (o cyklických kódech pojednává kapitola 2.3.7):

$$\boldsymbol{G} = \begin{bmatrix} 1000\,101 \\ 0100\,111 \\ 0010\,110 \\ 0001\,011 \end{bmatrix} \qquad \boldsymbol{H} = \begin{bmatrix} 1110\,100 \\ 0111\,010 \\ 1101\,001 \end{bmatrix}$$

(Konec ilustračního příkladu)

Vytvoření generujících a kontrolních matic systematických Hammingových kódů je obecně velmi jednoduché. Pro zadaný počet kontrolních prvků r z rovnosti $2^{\mathrm{r}} = k + r + 1$ spočítáme počet informačních znaků k . Generující matice tedy bude mít k řádků a $k + r$ sloupců. V „levé části“ generující matice bude blok I_k jednotkové matice řádu k . „Pravou část“ matice (tedy v ilustračním příkladu podžlucenou „submatici“ B z vyjádření obecného tvaru generující matice systematického kódu) vytvoříme tak, že do jejích řádků umístíme všechny $r-$ bitové řetězce, obsahující **alespoň dvě jedničky** (tedy binární rozvoje čísel, jež nejsou mocninami dvou). Do k řádků umístíme postupně všechny takové kombinace (žadná nebude nepoužita, žádná nebude použita dvakrát). Proč právě řetězce s alespoň dvěma jedničkami? Protože v kontrolní matici se submatice B objeví transponovaná, její řádky se stanou sloupci systematické kontrolní matice a v té jsou už sloupce s právě jednou jedničkou obsaženy v „pravém bloku“, tj. v jednotkové matici I_{n-k} řádu $n-k$. Na pořadí řádků matice B nezáleží, nicméně bývá dobrým zvykem uvádět rozvoje vhodných čísel v rostoucím pořadí, tj. po řádcích X,X,3, X,5,6,7, X,9,10,11,12,13,14,15, X,17,…….), jak je vidět v první generující matici systematického $(7,4)$ kódu. Kontrolní matici pak vytvoříme standardním postupem jako $\boldsymbol{H} = [-\boldsymbol{B^T} | \, \boldsymbol{I_{n-k}}]$.

Minimální vzdálenost Hammingova kódu je rovna minimální váze nenulové kódové značky, je tedy 3. Proč? Všechny značky, které jsou v řádcích generující matice G, mají váhu přinej-

menším 3 (jedna jednička v informační části a minimálně dvě jedničky v kontrolní části). To samo o sobě k tvrzení o minimální váze **všech** značek kódu nestačí, musíme ještě prozkoumat i všechny lineární kombinace řádků matice G. Součet dvou řádků matice G má v informační části dvě jedničky a v zabezpečovací části minimálně jednu další jedničku, protože všechny řádky submatice B jsou různé. Sečteme-li tři nebo více řádků matice G, máme „potřebné“ tři jedničky už v informační části. Závěr: v Hammingových kódech neexistují kódové značky, které by měly jednu nebo dvě jedničky, minimální vzdálenost Hammingova kódu je tedy 3.

Perfektní kódy

Hammingovy kódy mají důležitou vlastnost: všechna slova váhy $t \leq 1$ jsou reprezentanty chybových tříd, tj. tříd rozkladu podle zobrazení ω, kde $\omega(\boldsymbol{e}) = \boldsymbol{H} \cdot \boldsymbol{e}$. Jinak řečeno – každému syndromu přísluší jedinečný chybový vektor váhy $t \leq 1$ a naopak každému chybovému vektoru váhy váhy $t \leq 1$ přísluší jedinečný syndrom. Důsledkem této vlastnosti je fakt, že mají Hammingovy kódy nejmenší redundanci ze všech kódů pro opravy jednoduchých chyb. Takovým kódům říkáme *perfektní kódy.* Obecně:

Lineární kód je *perfektní pro opravy t −násobných chyb* , jestliže všechna chybová slova váhy $\leq t$ jsou reprezentanty chybových tříd.

Kromě Hammingových kódů patří mezi perfektní kódy ještě opakovací kód s lichou délkou značky $n = 2 \cdot t + 1$ (opravuje t −násobné chyby) a Golayův kód (opravuje trojité chyby). Jiné perfektní binární kódy neexistují.

Jak postupovat, pokud máme vytvořit kód pro opravu jednoduchých chyb pro obecný počet informačních prvků k ? Počet kontrolních znaků r kódu pro opravu jednoduchých chyb (bez pořadavku na minimální redundanci) musí vyhovovat nerovnosti $2^{\mathrm{r}} \geq k + r + 1$. Najdeme tedy nejmenší číslo r , které je řešením nerovnice. Generující matici vytvoříme stejným postupem jako u Hammingových kódů s tím rozdílem, že (v případě, že $2^{\mathrm{r}} \neq k + r + 1$) nevyčerpáme všechny $r -$ bitové řetězce s alespoň dvěma jedničkami (řádků v submatici B je méně než řetězců s touto vlastností).

Ilustrační příklad:

Navrhněte systematický kód pro kódování šestibitových informačních částí, který bude opravovat jednoduché chyby.

Nejprve zjistíme potřebný počet kontrolních znaků:

$2^{\mathrm{r}} \geq k + r + 1 \quad \Rightarrow \; 2^{\mathrm{r}} \geq 6 + r + 1 \Rightarrow \; 2^{\mathrm{r}} \geq r + 7 \quad \Rightarrow r = 4$ (nejmenší vyhovující r)

Výše popsaným postupem vytvoříme generující matici $\boldsymbol{G}$, kontrolní jako $\boldsymbol{H} = [-\boldsymbol{B^T} \mid \boldsymbol{I_{n-k}}]$.

$$\boldsymbol{G} = \begin{bmatrix} 100000\ 0011 \\ 010000\ 0101 \\ 001000\ 0110 \\ 000100\ 0111 \\ 000010\ 1001 \\ 000001\ 1010 \end{bmatrix} \qquad \boldsymbol{H} = \begin{bmatrix} 000011\ 1000 \\ 011100\ 0100 \\ 101101\ 0010 \\ 110110\ 0001 \end{bmatrix}$$

Ke stejnému výsledku bychom došli *zkrácením* Hammingova kódu $(15,11)$ (žádný z Hammingových kódů s menším r k zabezpečení 6 informačních znaků nestačí). *Zkrácením* budeme rozumět to, že z generující a kontrolní matice kódu $(15,11)$ vypustíme ty řady (tj. řádky a sloupce v matici $G_{(15,11)}$, respektive sloupce v matici $H_{(15,11)}$), které odpovídají nevyužitým informačním prvkům. Z matice $G_{(15,11)}$ tedy vypustíme řádky 7 až 11 a sloupce 7 až 11, v matici $H_{(15,11)}$ pak vypustíme sloupce 7 až 11. V následujících maticích Hammingova kódu $(15,11)$ jsou řady, které budou vypuštěny, podbarveny. Nepodbarvený „zbytek" matic pak tvoří výše uvedené matice G a H , ke kterým jsme jednodušší cestou došli výše.

$$\boldsymbol{G} = \begin{bmatrix} 100000\mathbf{00000}\ 0011 \\ 010000\mathbf{00000}\ 0101 \\ 001000\mathbf{00000}\ 0110 \\ 000100\mathbf{00000}\ 0111 \\ 000010\mathbf{00000}\ 1001 \\ 000001\mathbf{00000}\ 1010 \\ \mathbf{00000010000\ 1011} \\ \mathbf{00000001000\ 1100} \\ \mathbf{00000000100\ 1101} \\ \mathbf{00000000010\ 1110} \\ \mathbf{00000000001\ 1111} \end{bmatrix} \qquad \boldsymbol{H} = \begin{bmatrix} 000011\mathbf{11111}\ 1000 \\ 011100\mathbf{01111}\ 0100 \\ 101101\mathbf{10011}\ 0010 \\ 110110\mathbf{10101}\ 0001 \end{bmatrix}$$

(Konec ilustračního příkladu)

Rozšíření kódu

Lineární kód s **lichou** minimální vzdáleností d_0 lze *rozšířit* tak, že ke každé kódové značce přidáme další kontrolní prvek – celkovou kontrolu parity, tj. doplnění značky tak, aby měla sudý počet jedniček. Minimální Hammingova vzdálenost rozšířeného kódu pak bude $d_0 + 1$. **POZOR! Rozšíření kódu se sudou minimální vzdáleností tuto vzdálenost nezvětší!**

Ilustrační příklad (rozšířený Hammnigův kód $(8,4)$:

$$\boldsymbol{G} = \begin{bmatrix} 1000\ \mathbf{011}\ \mathbf{1} \\ 0100\ \mathbf{101}\ \mathbf{1} \\ 0010\ \mathbf{110}\ \mathbf{1} \\ 0001\ \mathbf{111}\ \mathbf{0} \end{bmatrix} \qquad \boldsymbol{H_1} = \begin{bmatrix} \mathbf{0111}\ 1000 \\ \mathbf{1011}\ 0100 \\ \mathbf{1101}\ 0010 \\ \mathbf{1110}\ 0001 \end{bmatrix} \qquad \boldsymbol{H_2} = \begin{bmatrix} 0111\ 100\ \mathbf{0} \\ 1011\ 010\ \mathbf{0} \\ 1101\ 001\ \mathbf{0} \\ \mathbf{1111\ 111\ 1} \end{bmatrix}$$

Generující matici kódu jsme vytvořili tak, že jsme každou bázovou značku (tj. každý řádek matice $\boldsymbol{G}$) rozšířili o paritní znak (modře podbarvené kontrolní prvky). Žlutě podbarvené prv-

ky představují blok $\boldsymbol{B}$, který generuje kontrolní prvky „standardního“ Hammingova kódu $(7,4)$.

Kontrolní matici $\boldsymbol{H_1}$ vytvoříme standardně jako $\boldsymbol{H_1} = \left[-\widetilde{\boldsymbol{B}}^{\boldsymbol{T}} \middle| \boldsymbol{I_{n-k}}\right]$. Žlutě a modře podbarvené prvky v kontrolní matici ilustrují transpozici bloku $\widetilde{\boldsymbol{B}}$ (u kódu $(8,4)$ je už ovšem tento blok tvořen nejenom žlutým blokem $\boldsymbol{B}$ „výchozího“ kódu $(7,4)$, ale i modrým sloupcem paritních prvků).

Jiná kontrolní matice $\boldsymbol{H_2}$ byla vytvořena na základě toho, že jsme rozšiřující kontrolní znak definovali jako celkovou kontrolu parity. K původním třem kontrolním rovnicím jsme doplnili čtvrtou kontrolní rovnici $v_1 + v_2 + v_3 + v_4 + v_5 + v_6 + v_7 + v_8 = 0$. První tři kontrolní rovnice se nezměnily (nově přidaný paritní prvek nezabezpečují; tomu odpovídají šedivě podbarvené nuly), zeleně podbarvené jedničky vyjadřují přidanou rovnici celkové kontroly parity.

(Konec ilustračního příkladu)

2.3.5. Golayovy kódy

Golayův kód $\boldsymbol{G_{23}}$ je systematický binární kód s délkou značky 23 znaků. Je to perfektní kód pro opravu trojitých chyb, má tedy minimální redundanci. Jeho generující matice je typu 12/23 a má tvar

$$\boldsymbol{G_{23}} = \begin{bmatrix} 100000000000 & 11011100010 \\ 010000000000 & 01101110001 \\ 001000000000 & 10110111000 \\ 000100000000 & 01011011100 \\ 000010000000 & 00101101110 \\ 000001000000 & 00010110111 \\ 000000100000 & 10001011011 \\ 000000010000 & 11000101101 \\ 000000001000 & 11100010110 \\ 000000000100 & 01110001011 \\ 000000000010 & 10111000101 \\ 000000000001 & 11111111111 \end{bmatrix} = \left[\boldsymbol{I_{12}} \mid \frac{\boldsymbol{B}}{11 \ldots .1}\right]$$

kde žlutě podbarvená čvercová submatice $\boldsymbol{B}$ řádu 11 je tvořena cyklickými posuvy prvního řádku, tedy slova 11011100010 . Pro příznivce „opravdové matematiky“: Jedničky v prvním řádku v posouvaném slově jsou v pozicích *i*, jež jsou kvadráty prvků v *tělese* Z_{11} (pozice jsou indexovány zleva jako $0\ 1\ 2\ 3 \ldots \ldots \ldots 10$)

Golayův kód $\boldsymbol{G_{24}}$ vznikne rozšířením kódu $\boldsymbol{G_{23}}$ o celkovou kontrolu parity (modře podbarvený sloupec). Jeho generující matice je tedy typu 12/24 a má tvar

$$G_{24} = \begin{array}{c} \begin{matrix} a_0 & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_8 & a_9 & a_{10} & a_{11} \end{matrix} \\ \left[\begin{matrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{matrix}\right. \end{array} \begin{array}{c} \begin{matrix} b_0 & b_1 & b_2 & b_3 & b_4 & b_5 & b_6 & b_7 & b_8 & b_9 & b_{10} & b_{11} \end{matrix} \\ \left[\begin{matrix} 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{matrix}\right. \end{array} \begin{matrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \\ c_8 \\ c_9 \\ c_{10} \\ c_{11} \end{matrix}$$

Označení sloupců a řádků matice jsme zavedli proto, abychom mohli popsat důležité vlastnosti kódu $\boldsymbol{G_{24}}$, které později využijeme při dekódování. Symbolem $\widetilde{\boldsymbol{B}}$ označíme „zabezpečovací část“ matice $\boldsymbol{G_{24}}$ (tj. všechny prvky podbarvené žlutě, zeleně a modře). Symboly $\boldsymbol{b_i}$ (respektive $\boldsymbol{c_i}$) označují sloupce (respektive řádky) submatice $\widetilde{\boldsymbol{B}}$.

Generující matice $\boldsymbol{G_{24}}$ má několik „pozoruhodných“ vlastností, které lze principielně jednoduše (nicméně pracně) ověřit:

- v každém řádku žlutě podbarvené čvercové submatice $\boldsymbol{B}$ je šest jedniček
- v každém řádku matice $\boldsymbol{G_{24}}$ je tedy nejméně osm jedniček (u řádků 0 až 10 přibude jednička v informační části a parita; řádek 11 má dokonce 12 jedniček)
- každé dva řádky submatice $\boldsymbol{B}$ mají právě tři společné jedničky
- každá lineární kombinace řádků matice $\boldsymbol{G_{24}}$ má nejméně osm jedniček
- váha každého kódové značky kódu $\boldsymbol{G_{24}}$ je násobkem 4
- pro každé dvě značky $u, v \in K$ platí $\sum_{i=1}^{24} u_i \cdot v_i = 0$, kód je tedy podprostorem svého ortogonálního doplňku; protože je dimenze kódu $\boldsymbol{G_{24}}$ 12, má jeho ortogonální doplněk stejnou dimenzi $(24 - 12 = 12)$, tudíž kód $\boldsymbol{G_{24}}$ je „doplňkem sebe sama“ (*samoduální kód*), tj. jako kontrolní matici lze použít matici generující - $\boldsymbol{H_{24}} = \boldsymbol{G_{24}}$.

Důsledky:

- kód $\boldsymbol{G_{24}}$ má minimální Hammingovu vzdálenost 8
- je-li kódovou značkou slovo

  $[\, w_0\, w_1\, w_2\, w_3 w_4 w_5 w_6 w_7 w_8 w_9 w_{10} w_{11}\, w'_0\, w'_1\, w'_2\, w'_3 w'_4\, w'_5\, w'_6 w'_7\, w'_8\, w'_9\, w'_{10} w'_{11}\,]^T$

  pak, je kódovou značkou i slovo

  $[\, w'_0\, w'_{10}\, w'_9\, w'_8 w'_7\, w'_6\, w'_5 w'_4\, w'_3\, w'_2\, w'_1 w'_{11}\, w_0 w_{10}\, w_9\, w_8 w_7\, w_6\, w_5 w_4\, w_3\, w_2\, w_1 w_{11}\,]^T$

  Jinak řečeno – provedeme-li v matici $\boldsymbol{G_{24}}$ permutace sloupců

  $\boldsymbol{a_0} \leftrightarrow \boldsymbol{b_0}$, $\boldsymbol{a_{11}} \leftrightarrow \boldsymbol{b_{11}}$ a $\boldsymbol{a_i} \leftrightarrow \boldsymbol{b_{11-i}}$ $\forall\, i = 1,2, \ldots, 10$,

  zůstávají vlastnosti kódu zachovány. Tuto vlastnost využívá dekódovací algoritmus.

Vlstnosti Golayova kódu G_{23}

- kód $\boldsymbol{G_{23}}$ má minimální Hammingovu vzdálenost 7, opravuje tedy trojnásobné chyby (každé slovo váhy $t \leq 3$ generuje jiný syndrom)
- počet různých syndromů v kódu $\boldsymbol{G_{23}}$ je $2^r = 2^{11}$
- počet různých slov váhy $t \leq 3$ je
  $1 + 23 + \binom{23}{2} + \binom{23}{3} = 1 + 23 + \frac{23!}{21!\cdot 2!} + \frac{23!}{20!\cdot 3!} = 1 + 23 + \frac{23\cdot 22}{2\cdot 1} + \frac{23\cdot 22\cdot 21}{3\cdot 2\cdot 1} =$
  $= 1 + 23 + 23 \cdot 11 + 23 \cdot 11 \cdot 7 = 1 + 23 + 253 + 1771 = 2048 = 2^{11}$
- existuje vzájemně jednoznačné zobrazení mezi syndromy a (chybovými) slovy váhy $t \leq 3$; slova váhy $t \leq 3$ jsou tedy reprezentanty chybových tříd, kód $\boldsymbol{G_{23}}$ je tedy perfektní pro opravu trojnásobných chyb

Dekódování Golayova kódu G_{24}

Dekódování Golayova kódu $\boldsymbol{G_{24}}$ probíhá ve třech krocích:

První krok: Provedeme výpočet syndromu $\boldsymbol{s} = \boldsymbol{H} \cdot \boldsymbol{w}$ (syndrom $\boldsymbol{s}$ je sloupec o 12 prvcích).

Druhý krok: Provedeme výpočet pomocného slova $\boldsymbol{t} = \widetilde{\boldsymbol{B}} \cdot \boldsymbol{s}$, kde $\boldsymbol{t} = [t_0 t_1 \ldots . t_{11}]^{\boldsymbol{T}}$ je sloupec o 12 prvcích a $\widetilde{\boldsymbol{B}}$ je „zabezpečovací část“ matice $\boldsymbol{G_{24}}$.

Třetí krok: Provedeme výpočet reprezentanta chybové třídy $\hat{\boldsymbol{e}}$ (sloupce o 24 prvcích) tímto způsobem:

- prozkoumáme váhy těchto 26 slov: $\boldsymbol{s}, \boldsymbol{t}$, $\boldsymbol{s} + \boldsymbol{b_i}$ a $\boldsymbol{t} + \boldsymbol{c_i^T}$ $\forall\, i = 0,1, \ldots ,11$, kde $\boldsymbol{b_i}$ je i −tý sloupec submatice $\widetilde{\boldsymbol{B}}$ a $\boldsymbol{c_i}$ je i −tý řádek submatice $\widetilde{\boldsymbol{B}}$
- je-li váha syndromu $\|\boldsymbol{s}\| \leq 3$, vytvoříme chybový vektor $\hat{\boldsymbol{e}}$ jako
  $\hat{\boldsymbol{e}} = \begin{bmatrix} \boldsymbol{s} \\ \boldsymbol{0} \end{bmatrix} = [s_0 s_1 s_2 s_3 s_4 s_5 s_6 s_7 s_8 s_9 s_{10} s_{11}\ 000000000000]^T$
- je-li váha pomocného slova $\|\boldsymbol{t}\| \leq 3$, vytvoříme chybový vektor $\hat{\boldsymbol{e}}$ jako
  $\hat{\boldsymbol{e}} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{t} \end{bmatrix} = [000000000000\ t_0 t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_{10} t_{11}]^T$
- je-li pro nějaký index i váha slova $\|\boldsymbol{s} + \boldsymbol{b_i}\| \leq 2$, vytvoříme chybový vektor $\hat{\boldsymbol{e}}$ jako
  $\hat{\boldsymbol{e}} = \begin{bmatrix} \boldsymbol{s} + \boldsymbol{b_i} \\ \boldsymbol{d_i} \end{bmatrix}$, kde $\boldsymbol{d_i}$ je sloupec o 11 prvcích, který má jedničku v i −té pozici a všechny ostatní znaky jsou nulové
- je-li pro nějaký index i váha slova $\left\|\boldsymbol{t} + \boldsymbol{c_i^T}\right\| \leq 2$, vytvoříme chybový vektor $\hat{\boldsymbol{e}}$ jako
  $\hat{\boldsymbol{e}} = \begin{bmatrix} \boldsymbol{d_i} \\ \boldsymbol{t} + \boldsymbol{c_i^T} \end{bmatrix}$, kde $\boldsymbol{d_i}$ má stejný význam jako výše.

Byl-li splněn předpoklad, že došlo k trojité chybě (tj. váha chybového slova $\|\boldsymbol{e}\| \leq 3$), pak je splněna právě jedna z výše fomulovaných podmínek a slovo $\hat{\boldsymbol{v}} = \boldsymbol{w} - \hat{\boldsymbol{e}}$ je rovno vyslanému slovu $\boldsymbol{v}$.

Použití Golayova kódu

Kód G_{24} byl úspěšně použit ke kódování při přenosu stovek barevných obrázků Jupitera a Saturnu, které na začátku osmdesátých let pořizovaly vesmírné sondy Voyager 1 a 2. Dodnes je tento kód součástí některých komunikačních standardů U.S. Army a komunikačních postupů v civilném letectví.

2.3.6. Reedovy - Mullerovy kódy

Reedovy – Mullerovy představují třídu nesystematických kódů, umožňující opravy předem zadaného počtu chyb. Současně umožňují relativně jednoduché dekódování.

Vytvoření generující matice

Pro libovolná čísla $m \epsilon N$ a $r \epsilon N_0$, kdy $0 \leq r \leq m$, lze zkonstruovat matici o $n = 2^m$ sloupcích jako blokovou matici

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

kde $\boldsymbol{G_0} = [1\ 1\ \ldots 1]$ je matice typu $1/n$,

$\boldsymbol{G_1} = \begin{bmatrix} 0\ 0\ \ldots 0\ 1 \\ \vdots \\ 0\ 0\ \ldots 1\ 1 \\ 0\ 1\ \ldots 1\ 1 \end{bmatrix}$ je matice typu m/n , obsahující popořadě všechny sloupce z 0 a 1

a

$\boldsymbol{G_l}$, kde $2 \leq l \leq r$ jsou matice, jejichž obsahují všechny možné součiny l řádků matice $\boldsymbol{G_1}$ (rozumí se „součiny po složkách“).

Ilustrační příklad: Vytvoření matice $\boldsymbol{G}$ pro $m = 3$ a $r = 3$:

$\boldsymbol{G_0} = [1\ 1\ 1\ 1\ 1\ 1\ 1\ 1]$,

$\boldsymbol{G_1} = \begin{bmatrix} 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1 \\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1 \\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1 \end{bmatrix}$

$\boldsymbol{G_2} = \begin{bmatrix} 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1 \\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1 \\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 1 \end{bmatrix}$ $\begin{matrix} 1.\text{ř.}\,\boldsymbol{G_1} \cdot 2.\text{ř.}\,\boldsymbol{G_1} \\ 2.\text{ř.}\,\boldsymbol{G_1} \cdot 3.\text{ř.}\,\boldsymbol{G_1} \\ 1.\text{ř.}\,\boldsymbol{G_1} \cdot 3.\text{ř.}\,\boldsymbol{G_1} \end{matrix}$ (součiny řádků jsou „bit po bitu“).

$\boldsymbol{G_3} = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 1]$ $1.\text{ř.}\,\boldsymbol{G_1} \cdot 2.\text{ř.}\,\boldsymbol{G_1} \cdot 3.\text{ř.}\,\boldsymbol{G_1}$

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{G_0} \\ \boldsymbol{G_1} \\ \boldsymbol{G_2} \\ \boldsymbol{G_3} \end{bmatrix} = \left[\begin{array}{cccccccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

Přerušení ilustračního příkladu.

Z příkladu lze vypozorovat následující:

- v každé matici $\boldsymbol{G_l}$ mají všechny řádky stejný počet jedniček, a to 2^{m-l}
- řádky matice $\boldsymbol{G}$ jsou lineárně nezávislé (není vidět na první pohled, ale snadno se ověří převedením na stupňovitý tvar)

Reedovým – Mullerovým kódem (dále jen R-M kódem) s pamatry m, r ($m \in N$, $r \in N_0$, $0 \leq r \leq m$) pak rozumíme binární kód určený generující maticí

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{G_0} \\ \boldsymbol{G_1} \\ \vdots \\ \boldsymbol{G_r} \end{bmatrix}$$, jejíž konstrukce byla popsána výše.

Počet informačních znaků kódu je $k = 1 + m + \binom{m}{2} + \cdots + \binom{m}{r} = \sum_{l=0}^{r} \binom{m}{l}$.

Obecné značení R-M kódů je $K_{r,m}$.

Je zřejmé, že $K_{0,m}$ je opakovací kód, $K_{m-2,m}$ je rozšířený Hammingův kód, $K_{m-1,m}$ je paritní kód, $K_{m,m}$ je množina všech n −tic, tedy Z_2^n, kde $n = 2^m$.

Minimální vzdálenost R-M kódu $K_{r,m}$ je $d_0 = 2^{m-r}$, opravuje tedy $t -$ násobné chyby, kde $t \leq 2^{m-r-1} - 1$.

Návrat k ilustračnímu příkladu:

Vybereme-li z matice $\boldsymbol{G}$ submatici $[\boldsymbol{G_0}]$, vytvoříme tak generující matici R-M kódu $K_{0,3}$:

$$\boldsymbol{G} = [\boldsymbol{G_0}] = [1\ 1\ 1\ 1\ 1\ 1\ 1\ 1] \quad .$$

Vidíme, že je to matice opakovacího kódu délky $n = 8$, opravuje tedy chyby násobnosti $t \leq (n-2)/2$, tedy $t = 3$.

Vybereme-li z matice $\boldsymbol{G}$ submatice $\begin{bmatrix} \boldsymbol{G_0} \\ \boldsymbol{G_1} \end{bmatrix}$, vytvoříme tak generující matici R-M kódu $K_{1,3}$:

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{G_0} \\ \boldsymbol{G_1} \end{bmatrix} = \begin{bmatrix} 1\,1\,1\,1\,1\,1\,1\,1 \\ 0\,0\,0\,0\,1\,1\,1\,1 \\ 0\,0\,1\,1\,0\,0\,1\,1 \\ 0\,1\,0\,1\,0\,1\,0\,1 \end{bmatrix} .$$

Tento kód je ekvivalentní s rozšířeným Hammingovým kódem $(8,4)$, opravuje tedy jednoduché chyby.

Dekódování Reedových - Mullerových kódů

Dekódování so nejdříve ukážeme na příkladu kódu $K_{1,3}$. Nejprve podle sloupců matice $\boldsymbol{G}$ sestavíme „vytvořující rovnice“ pro jednotlivé prvky kódové značky $\boldsymbol{v}$. Protože se kódová značka vytvoří jako $\boldsymbol{v} = \boldsymbol{G^T} \cdot \boldsymbol{u}$, platí

$$v_1 = u_1 \quad , \quad v_2 = u_1 + u_4 \quad , \quad v_3 = u_1 + u_3 \quad , \quad v_4 = u_1 + u_3 + u_4$$
$$v_5 = u_1 + u_2 \, , \quad v_6 = u_1 + u_2 + u_4 \quad , \quad v_7 = u_1 + u_2 + u_3 , \quad v_8 = u_1 + u_2 + u_3 + u_4$$

Specifickým seskupením a následným sečtením výše uvedených rovnic dostaneme pro každý z prvků u_2, u_3, u_4 čtyři rovnice (rovnice číslujeme podle indexu u prvku v_i na levé straně rovnice:

Pro prvek u_4 sečteme rovnice takto:
(1 + 2): $u_4 = v_1 + v_2$ (3 + 4): $u_4 = v_3 + v_4$ (5 + 6): $u_4 = v_5 + v_6$ (7 + 8): $u_4 = v_7 + v_8$
Pro prvek u_3:
(1 + 3): $u_3 = v_1 + v_3$ (2 + 4): $u_3 = v_2 + v_4$ (5 + 7): $u_3 = v_5 + v_7$ (6 + 8): $u_3 = v_6 + v_8$
Pro prvek u_2:
(1 + 5): $u_2 = v_1 + v_5$ (2 + 6): $u_2 = v_2 + v_6$ (3 + 7): $u_2 = v_3 + v_6$ (4 + 8): $u_2 = v_4 + v_8$

Na tyto rovnice se ovšem nebudeme dívat jako na soustavu rovnic, u níž hledáme řešení ve smyslu lineární algebry, jejich řešením je totiž libovolná trojice u_2, u_3, u_4 .

Předpokládejme, že příjemce přijme slovo $\boldsymbol{w} = \boldsymbol{v} + \boldsymbol{e}$, kde $\boldsymbol{e}$ je chybový vektor obsahující právě jednu jedničku, a to v i −té pozici (čili $w_i = v_i + 1$, $w_j = v_j \;\; \forall\, j \neq i$). Kdybychom výše uvedené „sady rovnic“ formálně přepsali tak, že bychom místo prvků kódové značky v_i použili prvky přijatého slova w_i, neměla by tato „přeurčená“ soustava rovnic řešení (respektive měla by řešení pouze tehdy, pokud bychom z ní odstranili rovnice, v nichž se na pravé straně vyskytuje prvek w_i zatížený chybou).

„Řešení“ těchto rovnic budeme hledat na **majoritním principu.** Rovnice mají důležitou vlastnost: v „sadě rovnic“ pro každý z prvků u_2, u_3, u_4 se na pravých stranách rovnic každé w_i vyskytuje právě jednou. Při výskytu jednoduché chyby v i −té pozici vyjde tedy právě ta (jedna) rovnice, v níž se w_i vyskytuje na pravé straně, jinak než ostatní rovnice sady. Protože ale pro každý prvek u_i máme čtyři rovnice a „špatný výsledek“ dá v případě jednoduché chyby vždy právě jedna rovnice, rozhodne o „správném výsledku“ většina, která (v případě jednoduché chyby u kódu $K_{1,3}$) bude vždycky správně.

Konkrétně:

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{G_0} \\ \boldsymbol{G_1} \end{bmatrix} = \begin{bmatrix} 1\,1\,1\,1\,1\,1\,1\,1 \\ 0\,0\,0\,0\,1\,1\,1\,1 \\ 0\,0\,1\,1\,0\,0\,1\,1 \\ 0\,1\,0\,1\,0\,1\,0\,1 \end{bmatrix} \quad \boldsymbol{u} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\alpha_0} \\ \boldsymbol{\alpha_1} \end{bmatrix} \quad \boldsymbol{v} = \boldsymbol{G^T} \cdot \boldsymbol{u} = [1\,1\,0\,0\,0\,0\,1\,1]^{\boldsymbol{T}}$$

$$\boldsymbol{\alpha_0} = [u_1] = [1] \qquad \boldsymbol{\alpha_1} = [u_2\, u_3\, u_4]^{\boldsymbol{T}} = [1\,1\,0]^{\boldsymbol{T}}$$

$$\boldsymbol{e} = [0\,0\,1\,0\,0\,0\,0\,0]^{\boldsymbol{T}} \qquad \boldsymbol{w} = \boldsymbol{v} + \boldsymbol{e} = [1\,1\,1\,0\,0\,0\,1\,1]^{\boldsymbol{T}}$$

tedy $w_1 = w_2 = w_3 = w_7 = w_8 = 1$, ostatní w_i jsou nulové.

Informační část budeme při dekódování vnímat jako rozdělenou do segmentů, jak je naznačeno zeleným písmem. Počty prvků jednotlivých segmentů odpovídají počtu řádek odpovídajících submatic generující matice $\boldsymbol{G}$. Potom můžeme kódování informační části rozepsat „po segmentech“ jako

$$\boldsymbol{v} = \boldsymbol{G^T} \cdot \boldsymbol{u} = \begin{bmatrix} \boldsymbol{G_0} \\ \boldsymbol{G_1} \end{bmatrix}^{\boldsymbol{T}} \cdot \begin{bmatrix} \boldsymbol{\alpha_0} \\ \boldsymbol{\alpha_1} \end{bmatrix} = \boldsymbol{G_0^T} \cdot \boldsymbol{\alpha_0} + \boldsymbol{G_1^T} \cdot \boldsymbol{\alpha_1}$$

Typy matic: 8/4 4/1 8/1 1/1 8/3 3/1

Výpočet u_i z rovnic: **Majorita:**

Pro u_4:	$u_4 = w_1 + w_2$	$u_4 = w_3 + w_4$	$u_4 = w_5 + w_6$	$u_4 = w_7 + w_8$	
	0	1	0	0	**0**
Pro u_3:	$u_3 = w_1 + w_3$	$u_3 = w_2 + w_4$	$u_3 = w_5 + w_7$	$u_3 = w_6 + w_8$	
	0	1	1	1	**1**
Pro u_2:	$u_2 = w_1 + w_5$	$u_2 = w_2 + w_6$	$u_2 = w_3 + w_7$	$u_2 = w_4 + w_8$	
	1	1	0	1	**1**

Dále budeme pokračovat výpočtem prvku u_1. Nejdříve ovšem musíme přijaté slovo $\boldsymbol{w}$ „očistit od vlivu“ zakódovaného segmentu $\boldsymbol{\alpha_1}$, tj. již opravené části informačního slova $u_2u_3u_4$. Spočítáme tedy slovo $\boldsymbol{w'}$ jako $\boldsymbol{w'} := \boldsymbol{w} - \boldsymbol{G_1^T} \cdot \boldsymbol{\alpha_1} = \boldsymbol{w} - \boldsymbol{G_1^T} \cdot [110]^{\boldsymbol{T}} = [11100011]^{\boldsymbol{T}} - [00111100]^{\boldsymbol{T}} = [11011111]^{\boldsymbol{T}}$.

Prvním řádkem matice $\boldsymbol{G}$ je submatice $\boldsymbol{G_0}$, která (analogicky předešlému) představuje osm rovnic pro prvek u_1. Výpočet majority:

$$u_1 = w_1' = 1 \qquad u_1 = w_2' = 1 \qquad u_1 = w_3' = 0 \qquad u_1 = w_4' = 1$$
$$u_1 = w_5' = 1 \qquad u_1 = w_6' = 1 \qquad u_1 = w_7' = 1 \qquad u_1 = w_8' = 1$$

Majoritním principem určíme u_1 jako $u_1 = 1$.

Konec ilustračního příkladu.

Obecný *Reedův dekódovací algoritmus* je zobecněním výše uvedeného postupu v tom smyslu, že se pro libovolný R-M kód $K_{r,m}$ tento postup provede rekurzivně od segmentu s nejvyšším indexem, tedy od segmentu $\boldsymbol{\alpha_r}$. K popisu algoritmu tedy stačí popsat postup pro obecný segment $\boldsymbol{\alpha_l}$:

Pro každý informační prvek u_i segmentu $\boldsymbol{\alpha_l}$ vytvoříme sadu 2^{m-l} rovnic, které vyjádří prvek u_i jako součet některých symbolů přijatého slova $\boldsymbol{w}$ (respektive modifikovaného slova $\boldsymbol{w'}$). Počet sčítanců na levé straně každé rovnice je 2^l. Celkem se tedy na všech levých stranách jedné sady rovnic vyskytne $2^{m-l} \cdot 2^l = 2^m = n$ symbolů přijatého slova $\boldsymbol{w}$. **Každý symbol $\boldsymbol{w_i}$ se na levých stranách jedné sady rovnic vyskytne právě jednou.** Takovou sadu rovnic lze (právě jedním způsobem) vytvořit vždy, pro malé hodnoty l intuitivně, pro větší l může pomoci algoritmizace. „Řešení“, tj. hodoty informačních prvků u_i segmentu $\boldsymbol{\alpha_l}$ určíme na základě majority.

Dále budeme pokračovat výpočtem informačních prvků v dalším segmentu $\boldsymbol{\alpha_{l-1}}$. Předtím ale modifikujeme slovo $\boldsymbol{w'}$ jako $\boldsymbol{w'} := \boldsymbol{w'} - \boldsymbol{G_l^T} \cdot \boldsymbol{\alpha_l}$ (po výpočtu v „prvním“ segmentu $\boldsymbol{\alpha_r}$ ovšem slovo $\boldsymbol{w'}$ vytvoříme nově jako $\boldsymbol{w'} := \boldsymbol{w} - \boldsymbol{G_r^T} \cdot \boldsymbol{\alpha_r}$).

Výpočet končí řešením segmentu $\boldsymbol{\alpha_0}$ tedy určením hodnoty prvku u_1.

Nejmenší počet rovnic mají sady v r –tém segmentu, a to 2^{m-r} rovnic pro každý informační prvek. Předpokládejme, že v přijatém slově $\boldsymbol{w}$ došlo k t –násobné chybě a to takové, že se v jedné konkrétní sadě v r –tém segmentu neobjeví rovnice, která by obsahovala více než jeden chybou ovlivněný prvek přijatého slova. Jinak řečeno – chybami je ovlivněno právě t – rovnic sady. To je největší možný počet ovlivněných rovnic při t –násobné chybě (v případě, že se v jedné rovnici vyskytne sudý počet chybových prvků, jejich vliv se vzájemně „vykompenzuje“). Nadpoloviční většina ze (sudého) čísla 2^{m-r} je $(2^{m-r} - 2)/2$, tedy $2^{m-r-1} - 1$. Násobnost chyb t, které je R-M kód $K_{r,m}$ schopen opravit, je tedy $t \leq 2^{m-r-1} - 1$.

Pro znázornění souvislosti mezi chybami a výsledky rovnic jsou v následujících dvou ilustračních příkladech všechny hodnoty související s chybami v přijatém slově zapisovány červeně.

Ilustrační příklad: R-M kód $K_{2,4}$ (opravuje jednoduché chyby).

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{G_0} \\ \boldsymbol{G_1} \\ \boldsymbol{G_2} \end{bmatrix} = \begin{bmatrix} 1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1 \\ 0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1\,1\,1\,1\,1 \\ 0\,0\,0\,0\,1\,1\,1\,1\,0\,0\,0\,0\,1\,1\,1\,1 \\ 0\,0\,1\,1\,0\,0\,1\,1\,0\,0\,1\,1\,0\,0\,1\,1 \\ 0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1 \\ 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1 \\ 0\,0\,0\,0\,0\,0\,1\,1\,0\,0\,0\,0\,0\,0\,1\,1 \\ 0\,0\,0\,1\,0\,0\,0\,1\,0\,0\,0\,1\,0\,0\,0\,1 \\ 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,0\,1\,1 \\ 0\,0\,0\,0\,0\,1\,0\,1\,0\,0\,0\,0\,0\,1\,0\,1 \\ 0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,0\,1\,0\,1 \end{bmatrix} \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 1 \cdot 2 \\ 2 \cdot 3 \\ 3 \cdot 4 \\ 1 \cdot 3 \\ 2 \cdot 4 \\ 1 \cdot 4 \end{matrix} \qquad \boldsymbol{u} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\alpha_0} \\ \boldsymbol{\alpha_1} \\ \boldsymbol{\alpha_2} \end{bmatrix}$$

$$\boldsymbol{v} = \boldsymbol{G^T} \cdot \boldsymbol{u} = [1\,1\,1\,1\,0\,0\,1\,1\,0\,1\,1\,0\,0\,1\,0\,1]^T$$

$$\boldsymbol{e} = [0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0]^T \qquad \boldsymbol{w} = \boldsymbol{v} + \boldsymbol{e} = [1\,1\,0\,1\,0\,0\,1\,1\,0\,1\,1\,0\,0\,1\,0\,1]^T$$

tedy $w_1 = w_2 = w_4 = w_7 = w_8 = w_{10} = w_{11} = w_{14} = w_{16} = 1$, ostatní w_i jsou nulové.

Rovnice pro informační prvky ze segmentu $\boldsymbol{\alpha_2}$:

$$
\begin{aligned}
u_{11}: \quad & u_{11} = w_1 + w_2 + w_9 + w_{10} && = 1+1+0+1 = 1\\
& u_{11} = w_3 + w_4 + w_{11} + w_{12} && = 0+1+1+0 = 0\\
& u_{11} = w_5 + w_6 + w_{13} + w_{14} && = 0+0+0+1 = 1\\
& u_{11} = w_7 + w_8 + w_{15} + w_{16} && = 1+1+0+1 = 1\\
u_{10}: \quad & u_{10} = w_1 + w_2 + w_5 + w_6 && = 1+1+0+0 = 0\\
& u_{10} = w_3 + w_4 + w_7 + w_8 && = 0+1+1+1 = 1\\
& u_{10} = w_9 + w_{10} + w_{13} + w_{14} && = 0+1+0+1 = 0\\
& u_{10} = w_{11} + w_{12} + w_{15} + w_{16} && = 1+0+0+1 = 0\\
u_9: \quad & u_9 = w_1 + w_3 + w_9 + w_{11} && = 1+0+0+1 = 0\\
& u_9 = w_2 + w_4 + w_{10} + w_{12} && = 1+1+1+0 = 1\\
& u_9 = w_5 + w_7 + w_{13} + w_{15} && = 0+1+0+0 = 1\\
& u_9 = w_6 + w_8 + w_{14} + w_{16} && = 0+1+1+1 = 1\\
u_8: \quad & u_8 = w_1 + w_2 + w_3 + w_4 && = 1+1+0+1 = 1\\
& u_8 = w_5 + w_6 + w_7 + w_8 && = 0+0+1+1 = 0\\
& u_8 = w_9 + w_{10} + w_{11} + w_{12} && = 0+1+1+0 = 0\\
& u_8 = w_{13} + w_{14} + w_{15} + w_{16} && = 0+1+0+1 = 0\\
u_7: \quad & u_7 = w_1 + w_3 + w_5 + w_7 && = 1+0+0+1 = 0\\
& u_7 = w_2 + w_4 + w_6 + w_8 && = 1+1+0+1 = 1\\
& u_7 = w_9 + w_{11} + w_{13} + w_{15} && = 0+1+0+0 = 1\\
& u_7 = w_{10} + w_{12} + w_{14} + w_{16} && = 1+0+1+1 = 1\\
u_6: \quad & u_6 = w_1 + w_5 + w_9 + w_{13} && = 1+0+0+0 = 1\\
& u_6 = w_2 + w_6 + w_{10} + w_{14} && = 1+0+1+1 = 1\\
& u_6 = w_3 + w_7 + w_{11} + w_{15} && = 0+1+1+0 = 0\\
& u_6 = w_4 + w_8 + w_{12} + w_{16} && = 1+1+0+1 = 1
\end{aligned}
$$

Prvky segmentu $\boldsymbol{\alpha_2}$ tedy určíme jako $\boldsymbol{\alpha_2} = [u_6\, u_7\, u_8\, u_9\, u_{10}\, u_{11}]^{\boldsymbol{T}} = [1\ 1\ 0\ 1\ 0\ 1]^{\boldsymbol{T}}$.

Vytvoříme pomocné slovo $\boldsymbol{w'}$ jako $\boldsymbol{w'} := \boldsymbol{w} - \boldsymbol{G_2^T} \cdot \boldsymbol{\alpha_2} =$

$$= [1\ 1\ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ 0\ 1]^{\boldsymbol{T}} - \begin{bmatrix} 0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1\\ 0&0&0&0&0&0&1&1&0&0&0&0&0&0&1&1\\ 0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1\\ 0&0&0&0&0&0&0&0&0&0&1&1&0&0&1&1\\ 0&0&0&0&0&1&0&1&0&0&0&0&0&1&0&1\\ 0&0&0&0&0&0&0&0&0&1&0&1&0&1&0&1 \end{bmatrix}^{\boldsymbol{T}} \cdot \begin{bmatrix} 1\\1\\0\\1\\0\\1 \end{bmatrix} =$$

$$= [1\ 1\ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ 0\ 1]^{\boldsymbol{T}} - [0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 0]^{\boldsymbol{T}} =$$
$$= [1\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1]^{\boldsymbol{T}} \quad ,$$

tedy $w_1' = w_2' = w_4' = w_{13}' = w_{14}' = w_{15} = w_{16}' = 1$, ostatní w_i' jsou nulové.

Rovnice pro informační prvky ze segmentu $\boldsymbol{\alpha_1}$:

u_5 : $u_5 = w_1' + w_2' = 1 + 1 = 0$ $\quad u_5 = w_9' + w_{10}' = 0 + 0 = 0$
$u_5 = w_3' + w_4' = 0 + 1 = 1$ $\quad u_5 = w_{11}' + w_{12}' = 0 + 0 = 0$
$u_5 = w_5' + w_6' = 0 + 0 = 0$ $\quad u_5 = w_{13}' + w_{14}' = 1 + 1 = 0$
$u_5 = w_7' + w_8' = 0 + 0 = 0$ $\quad u_5 = w_{15}' + w_{16}' = 1 + 1 = 0$

u_4 : $u_4 = w_1' + w_3' = 1 + 0 = 1$ $\quad u_4 = w_9' + w_{11}' = 0 + 0 = 0$
$u_4 = w_2' + w_4' = 1 + 1 = 0$ $\quad u_4 = w_{10}' + w_{12}' = 0 + 0 = 0$
$u_4 = w_5' + w_7' = 0 + 0 = 0$ $\quad u_4 = w_{13}' + w_{15}' = 1 + 1 = 0$
$u_4 = w_6' + w_8' = 0 + 0 = 0$ $\quad u_4 = w_{14}' + w_{16}' = 1 + 1 = 0$

u_3 : $u_3 = w_1' + w_5' = 1 + 0 = 1$ $\quad u_3 = w_9' + w_{13}' = 0 + 1 = 1$
$u_3 = w_2' + w_6' = 1 + 0 = 1$ $\quad u_3 = w_{10}' + w_{14}' = 0 + 1 = 1$
$u_3 = w_3' + w_7' = 0 + 0 = 0$ $\quad u_3 = w_{11}' + w_{15}' = 0 + 1 = 1$
$u_3 = w_4' + w_8' = 1 + 0 = 1$ $\quad u_3 = w_{12}' + w_{16}' = 0 + 1 = 1$

u_2 : $u_2 = w_1' + w_9' = 1 + 0 = 1$ $\quad u_2 = w_5' + w_{13}' = 0 + 1 = 1$
$u_2 = w_2' + w_{10}' = 1 + 0 = 1$ $\quad u_2 = w_6' + w_{14}' = 0 + 1 = 1$
$u_2 = w_3' + w_{11}' = 0 + 0 = 0$ $\quad u_2 = w_7' + w_{15}' = 0 + 1 = 1$
$u_2 = w_4' + w_{12}' = 1 + 0 = 1$ $\quad u_2 = w_8' + w_{16}' = 0 + 1 = 1$

Prvky segmentu $\boldsymbol{\alpha_1}$ tedy určíme jako $\boldsymbol{\alpha_1} = [u_2\ u_3\ u_4\ u_5]^{\boldsymbol{T}} = [1\ 1\ 0\ 0]^{\boldsymbol{T}}$.

Modifikujeme pomocné slovo $\boldsymbol{w'}$ jako $\boldsymbol{w'} := \boldsymbol{w'} - \boldsymbol{G_1^T} \cdot \boldsymbol{\alpha_1} =$

$$= [1\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1]^{\boldsymbol{T}} - \begin{bmatrix} 0&0&0&0&0&0&0&0&1&1&1&1&1&1&1 \\ 0&0&0&0&1&1&1&1&0&0&0&0&1&1&1 \\ 0&0&1&1&0&0&1&1&0&0&1&1&0&0&1 \\ 0&1&0&1&0&1&0&1&0&1&0&1&0&1&1 \end{bmatrix}^{\boldsymbol{T}} \cdot \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} =$$

$$= [1\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1]^{\boldsymbol{T}} - [0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0]^{\boldsymbol{T}} =$$

$$= [1\ 1\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1]^{\boldsymbol{T}} \quad ,$$

tedy $w_3' = 0$, ostatní w_i' mají hodnotu $w_i' = 1$.

Rovnice pro informační prvek u_1 ze segmentu $\boldsymbol{\alpha_0}$:

$u_1 = w_1' = 1$ $\quad u_1 = w_5' = 1$ $\quad u_1 = w_9' = 0$ $\quad u_1 = w_{13}' = 1$
$u_1 = w_2' = 1$ $\quad u_1 = w_6' = 1$ $\quad u_1 = w_{10}' = 0$ $\quad u_1 = w_{14}' = 1$
$u_1 = w_3' = 0$ $\quad u_1 = w_7' = 1$ $\quad u_1 = w_{11}' = 0$ $\quad u_1 = w_{15}' = 1$
$u_1 = w_4' = 1$ $\quad u_1 = w_8' = 1$ $\quad u_1 = w_{12}' = 0$ $\quad u_1 = w_{16}' = 1$

Majoritním principem určíme u_1 jako $u_1 = 1$.

Byl-li splněn předpoklad, že došlo pouze k jednoduché chybě, bylo zakódováno a odesláno informační slovo $\boldsymbol{u} = [\boldsymbol{\alpha_0}\boldsymbol{\alpha_1}\boldsymbol{\alpha_2}]^{\boldsymbol{T}} = [1\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 1]^{\boldsymbol{T}}$.

Konec ilustračního příkladu R-M kód $K_{2,4}$.

Ilustrační příklad: R-M kód $K_{1,4}$ (opravuje trojnásobné chyby).

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{G_0} \\ \boldsymbol{G_1} \end{bmatrix} = \left[\begin{array}{cccccccccccccccc} 1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1 \\ \hline 0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1 \\ 0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1 \\ 0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1 \\ 0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1 \end{array}\right] \qquad \boldsymbol{u} = \begin{bmatrix} 0 \\ \hline 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\alpha_0} \\ \hline \boldsymbol{\alpha_1} \end{bmatrix}$$

$$\boldsymbol{v} = \boldsymbol{G}^{\boldsymbol{T}} \cdot \boldsymbol{u} = [0\,1\,0\,1\,1\,0\,1\,0\,1\,0\,1\,0\,0\,1\,0\,1\,]^{\boldsymbol{T}}$$

$$\boldsymbol{e} = [0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1]^{\boldsymbol{T}} \qquad \boldsymbol{w} = \boldsymbol{v} + \boldsymbol{e} = [0\,1\,1\,1\,1\,0\,1\,0\,1\,0\,1\,0\,0\,1\,1\,0\,]^{\boldsymbol{T}}$$

tedy $w_2 = w_3 = w_4 = w_5 = w_7 = w_9 = w_{11} = w_{14} = w_{15} = 1$, ostatní w_i jsou nulové.

Rovnice pro informační prvky ze segmentu $\boldsymbol{\alpha_1}$:

u_5 : $u_5 = w_1 + w_2 = 0 + 1 = 1$ $\quad u_5 = w_9 + w_{10} = 1 + 0 = 1$
$u_5 = w_3 + w_4 = 1 + 1 = 0$ $\quad u_5 = w_{11} + w_{12} = 1 + 0 = 1$
$u_5 = w_5 + w_6 = 1 + 0 = 1$ $\quad u_5 = w_{13} + w_{14} = 0 + 1 = 1$
$u_5 = w_7 + w_8 = 1 + 0 = 1$ $\quad u_5 = w_{15} + w_{16} = 1 + 0 = 1$
(důsledky obou chyb se „vykompenzovaly“)

u_4 : $u_4 = w_1 + w_3 = 0 + 1 = 1$ $\quad u_4 = w_9 + w_{11} = 1 + 1 = 0$
$u_4 = w_2 + w_4 = 1 + 1 = 0$ $\quad u_4 = w_{10} + w_{12} = 0 + 0 = 0$
$u_4 = w_5 + w_7 = 1 + 1 = 0$ $\quad u_4 = w_{13} + w_{15} = 0 + 1 = 1$
$u_4 = w_6 + w_8 = 0 + 0 = 0$ $\quad u_4 = w_{14} + w_{16} = 1 + 0 = 1$

u_3 : $u_3 = w_1 + w_5 = 0 + 1 = 1$ $\quad u_3 = w_9 + w_{13} = 1 + 0 = 1$
$u_3 = w_2 + w_6 = 1 + 0 = 1$ $\quad u_3 = w_{10} + w_{14} = 0 + 1 = 1$
$u_3 = w_3 + w_7 = 1 + 1 = 0$ $\quad u_3 = w_{11} + w_{15} = 1 + 1 = 0$
$u_3 = w_4 + w_8 = 1 + 0 = 1$ $\quad u_3 = w_{12} + w_{16} = 0 + 0 = 0$

u_2 : $u_2 = w_1 + w_9 = 0 + 1 = 1$ $\quad u_2 = w_5 + w_{13} = 1 + 0 = 1$
$u_2 = w_2 + w_{10} = 1 + 0 = 1$ $\quad u_2 = w_6 + w_{14} = 0 + 1 = 1$
$u_2 = w_3 + w_{11} = 1 + 1 = 0$ $\quad u_2 = w_7 + w_{15} = 1 + 1 = 0$
$u_2 = w_4 + w_{12} = 1 + 0 = 1$ $\quad u_2 = w_8 + w_{16} = 0 + 0 = 0$

Prvky segmentu $\boldsymbol{\alpha_1}$ tedy určíme jako $\boldsymbol{\alpha_1} = [u_2\,u_3\,u_4\,u_5]^{\boldsymbol{T}} = [1\,1\,0\,1]^{\boldsymbol{T}}$.

Modifikujeme pomocné slovo $\boldsymbol{w'}$ jako $\boldsymbol{w'} := \boldsymbol{w'} - \boldsymbol{G_1^T} \cdot \boldsymbol{\alpha_1} =$

$$= [0\,1\,1\,1\,1\,0\,1\,0\,1\,0\,1\,0\,0\,1\,1\,0]^{T} - \begin{bmatrix} 0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1 \\ 0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1 \\ 0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1 \\ 0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1 \end{bmatrix}^{T} \cdot \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} =$$

$$= [0\,1\,1\,1\,1\,0\,1\,0\,1\,0\,1\,0\,0\,1\,1\,0]^{T} - [0\,1\,0\,1\,1\,0\,1\,0\,1\,0\,1\,0\,0\,1\,0\,1]^{T} =$$

$$= [0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1]^{T} \quad ,$$

tedy $w_3' = w_{15}' = w_{16}' = 1$, ostatní w_i' mají hodnotu $w_i' = 0$.

Rovnice pro informační prvek u_1 ze segmentu $\boldsymbol{\alpha_0}$:

$u_1 = w_1' = 0$	$u_1 = w_5' = 0$	$u_1 = w_9' = 0$	$u_1 = w_{13}' = 0$
$u_1 = w_2' = 0$	$u_1 = w_6' = 0$	$u_1 = w_{10}' = 0$	$u_1 = w_{14}' = 0$
$u_1 = w_3' = 1$	$u_1 = w_7' = 0$	$u_1 = w_{11}' = 0$	$u_1 = w_{15}' = 1$
$u_1 = w_4' = 0$	$u_1 = w_8' = 0$	$u_1 = w_{12}' = 0$	$u_1 = w_{16}' = 1$

Majoritním principem určíme u_1 jako $u_1 = 0$.

Byl-li splněn předpoklad, že došlo pouze k trojnásobné chybě, bylo zakódováno a odesláno informační slovo $\boldsymbol{u} = [\boldsymbol{\alpha_0}\boldsymbol{\alpha_1}]^{T} = [0\,1\,1\,0\,1]^{T}$.

Konec ilustračního příkladu R-M kód $K_{1,4}$.

2.3.7. Cyklické kódy

Cyklické kódy jsou podmnožinou lineárních kódů. Oproti obecným vlastostem lineárních kódů navíc ještě vykazují *cykličnost* , tedy to, že s každou kódovou značkou patří do množiny kódových značek i její libovný cyklický posuv. Formálně:

$$\boldsymbol{v} = [v_0\ v_1 \dots \dots v_{n-1}]^{T} \in \boldsymbol{K} \Rightarrow [v_{n-1}v_0\ v_1 \dots \dots v_{n-2}]^{T} \in \boldsymbol{K}$$

(indukcí z výše popsaného cyklického posuvu o jednu pozici doprava snadno dovodíme uzavřenost množiny kódových značek vůči **libovolnému** cyklickému posuvu).

POZOR! V indexování prvků kódových značek jsme přešli od indexování „od 1 do n“ k indexování „od 0 do $n-1$“. Abychom totiž umožnili „matematické vyjádření“ cykličnosti, nebudeme značky cyklických kódů popisovat pomocí vektorů, ale budeme je reprezentovat pomocí polynomů (= mnohočlenů) nad prvočíselnými tělesem Z_p (indexy koeficientů mnohočlenu budou odpovídat mocnině v exponentu příslušného členu).

Binární cyklické kódy tedy budeme reprezentovat polynomy nad tělesem Z_2 .

POZNÁMKA: Matematický popis cyklických kódů je založen na faktorových okruzích polynomů. Cílem této kapitoly nebude detailní matematický výklad s přesnými definicemi všech

pojmů, ale spíše jen objasnění základních principů s využitím ilustračních příkladů a zejména pak ukázky postupů kódování a kontroly přijaté značky.

U čtenáře je předpokládána znalost pojmů polynom, stupeň polynomu a operací sečítání, násobení a dělení mnohočlenů nad prvočíselnými tělesy (zejména nad tělesem Z_2).

n −tice tedy budeme reprezentovat polynomy:

$$\boldsymbol{v} = [v_0\, v_1 \ldots v_i \ldots v_{n-1}\,]^{\boldsymbol{T}} \quad \approx \quad v(x) = v_0 + v_1 \cdot x + \cdots + v_i \cdot x^i + \cdots + v_{n-1} \cdot x^{n-1}$$

Například

$$\boldsymbol{v} = [00111010]^{\boldsymbol{T}} \quad \approx \quad v(x) = x^2 + x^3 + x^4 + x^6$$

Vynásobíme-li polynom $v(x)$ činitelem x , dostaneme polynom

$$\bar{v}(x) = v(x) \cdot x = x^3 + x^4 + x^5 + x^7 ,$$

jemuž odpovídá n −tice $\bar{\boldsymbol{v}} = [00011101]^{\boldsymbol{T}}$, která je cyklickým posuvem n −tice $\boldsymbol{v}$ o jednu pozici doprava.

Vynásobíme-li činitelem x polynom $\bar{v}(x)$, dostaneme polynom

$$\bar{\bar{v}}(x) = \bar{v}(x) \cdot x = x^4 + x^5 + x^6 + x^8 ,$$

jehož nejvyšší mocnina „přetekla“ mimo rozsah našich n −tic, v tomto případě tedy násobení činitelem x cyklickému posunu neodpovídá.

Hledáme-li operaci, odpovídající cyklickému posuvu o jednu pozici doprava, musíme nalézt mechanizmus, který do operace násobení polynomů zavede identitu $x^{\mathrm{n}} = x^0 = 1$, musíme tedy „u proměnné x počítat s exponenty v režimu modulo n“.

Uplatníme-li identitu $x^8 = 1$ na $\bar{\bar{v}}(x)$, dostaneme polynom $1 + x^4 + x^5 + x^6$, jemuž odpovídá n −tice $\bar{\bar{v}} = [10001110]^{\mathrm{T}}$, která je cyklickým posuvem n −tice $\bar{v}$ o jednu pozici doprava.

Má-li platit $x^{\mathrm{n}} = 1$, musí platit $x^{\mathrm{n}} - 1 = 0$. Analogicky k operacím v prvočíselných tělesech Z_{p} , kde je výsledek operace sečítání definován jako $(a + b) \bmod p$ (což lze vyjádřit identitou $p = 0$), dospějeme k operaci $u(x) * v(x) = [u(x) \cdot v(x)] \bmod (x^{\mathrm{n}} - 1)$, kde součin $u(x) \cdot v(x)$ je „standardní“ součin polynomů nad tělesem Z_p .

Výše zavedená operace $*$ je operací násobení v *okruhu polynomů modulo* $x^{\mathrm{n}} - 1$. Tento okruh je tvořen všemi polynomy stupně menšího než n . Operace sečítání polynomů v tomto okruhu je „standardní“ sečítání polynomů, operace násobení je definována výše jako zbytek po dělení „standardního součinu“ „modulovým polynomem“ $x^{\mathrm{n}} - 1$ (tedy vztahem $x^{\mathrm{n}} = 1$)

Ilustrační příklad: Okruh polynomů modulo $(x^2 - 1)$ nad tělesem Z_2 (značení - $Z_2 / x^2 - 1$).

Prvky tohoto okruhu jsou všechny polynomy stupně menšího než n = 2, jsou to tedy mnohočleny $0,\ 1,\ x,\ x + 1$. Operace nad nimi nadefinujeme takto:

+	0	1	x	x + 1
0	0	1	x	x + 1
1	1	0	x + 1	X
x	X	x + 1	0	1
x + 1	x + 1	x	1	0

*	0	1	x	x + 1
0	0	0	0	0
1	0	1	x	x + 1
x	0	x	1	x + 1
x + 1	0	x + 1	x + 1	0

Při konstrukci tabulky operace sečítání **+** je nutné si uvědomit, že v tělese Z_2 platí $1 + 1 = 0$, tedy také $x + x = 0$ (a v případě $n > 2$ i $x^2 + x^2 = 0$, $x^3 + x^3 = 0$, atd.).

Při konstrukci tabulky operace násobení ***** budeme využívat identity $x^2 = 1$. Například součin $x \cdot (x + 1)$ spočítáme tak, že nejprve mnohočleny vynásobíme „standardně“ , tedy $x \cdot (x + 1) = x^2 + x$. Pak položíme $x^2 = 1$ a vyjádříme $x * (x + 1) = 1 + x = x + 1$.

(Konec Ilustračního příkladu)

Ilustrační příklad: Paritní kód délky $n = 4$.

V předchozích kapitolách jsme již poznali paritní kód, který je nepochybně také kódem cyklickým (sudost/lichost počtu jedniček ve značce se cyklickými posuvy nemění). Pro konkrétní délku $n = 4$ tvoří množinu kódových značek tyto čtveřice: $\{0000, 1100, 1010, 1001, 0110, 0101, 0011, 1111\}$. V následující tabulce vyjádříme kódové značky pomocí polynomů a ukážeme, že každý „značkový polynom“ lze vyjádřit jako násobek jistého „speciálního“ polynomu $g(x)$:

v	*v(x)*	*v(x) = u(x) * g(x)*	*u(x)*
0000	0	$0 \cdot (1 + x)$	000
1100	1 + x	$1 \cdot (1 + x)$	100
1010	$1 + x^2$	$(1 + x) \cdot (1 + x)$	110
1001	$1 + x^3$	$(1 + x + x^2) \cdot (1 + x)$	111
0110	$x + x^2$	$x \cdot (1 + x)$	010
0101	$x + x^3$	$(x + x^2) \cdot (1 + x)$	011
0011	$x^2 + x^3$	$x^2 \cdot (1 + x)$	001
1111	$1 + x + x^2 + x^3$	$(1 + x^2) \cdot (1 + x)$	101

Oním „speciálním“ polynomem $g(x)$ je v případě parity mnohočlen $g(x) = 1 + x$.

Přerušení ilustračního příkladu

V každém cyklickém kódu splňujícím podmínku $1 < k < n$ existuje polynom $g(x)$ *(generující mnohočlen)*. Stupeň generujícího mnohočlenu je roven $n - k$ (tedy počtu zabezpečovacích prvků ve značce). Všechny polynomy reprezentující kódové značky jsou násobkem generují-

cího mnohočlenu v okruhu polynomů $\mathrm{Z}_p/\ x^n - 1$. Generující mnohočlen je nenulový polynom nejnižšího stupně ze všech „značkových“ polynomů. U binárních cyklických kódů (tedy kódů nad tělesem Z_2) je generující mnohočlen určen jednoznačně. U kódů nad obecným prvočíselným tělesem Z_p má vlastnosti generujícího mnohočlenu více mnohočlenů, ale všechny se liší jen multiplikativní konstantou (je-li $g(x)$ generujícím mnohočlenem, může být generujícím mnohočlenem i jeho libovolný násobek $k \cdot g(x)$ pro všechna nenulová $k \in \mathrm{Z}_p$).

Vlastnosti generujícího mnohočlenu $g(x)$:

- množina polynomů reprezentujících kódové značky cyklického kódu je tvořena všemi násobky generujícího mnohočlenu $g(x)$ v okruhu polynomů $\mathrm{Z}_p/\ x^n - 1$. Formálně $K = \{q(x) * g(x) \mid q(x) \in \mathrm{Z}_p/\ x^n - 1\}$
- polynomy $g(x),\ x \cdot g(x),\ x^2 \cdot g(x), \ldots\ldots, x^{k-1} \cdot g(x)$ tvoří bázi kódu K
- generující mnohočlen $g(x)$ je *nerozložitelný* nebo $(x + 1)$ násobkem nerozložitelného polynomu (viz Poznámka)
- generující mnohočlen $g(x)$ je *primitivní*, což představuje dvě vlastnosti:
  - $g(x)$ je dělitelem polynomu $x^n - 1$ (tj. dělí jej beze zbytku)
  - neexistuje $m < n$ takové, že $g(x)$ dělí beze zbytku $x^m - 1$

Z posledních vlastností generujícího mnohočlenu $g(x)$ je zřejmé, že ne každý mnohočlen nad tělesem Z_p může být generujícím mnohočlenem. Ověření primitivnosti polynomu je algoritmicky zvládnuté (i když výpočetně složité), jsou publikovány tabulky s polynomy, které mají vlastnosti generujících mnohočlenů.

Poznámka: Ukazuje se, že požadavky na nerozložitelnost a primitivnost generujícího mnohočlenu nejsou nepřekročitelné, existují i mnohočleny, které je nesplňují, například mnohočlen $\mathrm{x}^{32} + \mathrm{x}^{26} + \mathrm{x}^{23} + \mathrm{x}^{22} + \mathrm{x}^{16} + \mathrm{x}^{12} + \mathrm{x}^{11} + \mathrm{x}^{10} + \mathrm{x}^8 + \mathrm{x}^7 + \mathrm{x}^5 + \mathrm{x}^4 + \mathrm{x}^2 + \mathrm{x} + 1$ specifikovaný standardem IEEE 802.3 není primitivní [KOOPMAN].

Generující matice cyklického kódu

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{b}_0 \\ \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \vdots \\ \vdots \\ \boldsymbol{b}_{k-1} \end{bmatrix} = \begin{bmatrix} g(x) \\ x \cdot g(x) \\ x^2 \cdot g(x) \\ \vdots \\ \vdots \\ x^{k-1} \cdot g(x) \end{bmatrix} = \begin{bmatrix} g_0 & g_1 & g_2 & \cdots & g_{n-k} & 0 & \cdots & 0 & 0 & 0 \\ 0 & g_0 & g_1 & g_2 & \cdots & g_{n-k} & 0 & \cdots & 0 & 0 \\ 0 & 0 & g_0 & g_1 & g_2 & \cdots & g_{n-k} & 0 & \cdots & 0 \\ & & & \ddots & \ddots & \ddots & \ddots & \ddots & & \\ 0 & 0 & \cdots & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & \cdots & 0 & g_0 & g_1 & g_2 & \cdots & g_{n-k} \end{bmatrix}$$

reprezentace bázových prvků polynomy

reprezentace bázových prvků vektory

Kódování informační části

U lineárních kódů (tedy i u kódů cyklických) je kódová značka vytvořena jako lineární kombinace prvků báze, kde se za koeficienty této kombinace vezmou jednotlivé prvky informační části, tedy

$$\boldsymbol{v} = \varphi(u) = u_0 \cdot \boldsymbol{b}_0 + u_1 \cdot \boldsymbol{b}_1 + \cdots + u_{k-1} \cdot \boldsymbol{b}_{k-1}\,, \text{ tedy}$$

$$v(x) = u_0 \cdot g(x) + u_1 \cdot x\, g(x) + \cdots + u_{k-1} \cdot x^{k-1}\, g(x) =$$
$$= (u_0 + u_1 \cdot x + \cdots + u_{k-1} \cdot x^{k-1}) \cdot g(x) = u(x) \cdot g(x)$$

Z výše uvedeného odvození je zřejmé, že při kódování informační části může u cyklických kódů úlohu generující matice převzít generující mnohočlen. Informační části (tj. vektoru o k prvcích) přiřadíme *informační mnohočlen* tak, že

$$\boldsymbol{u} = [u_0\, u_1 \ldots u_i \ldots u_{k-1}\,]^{\boldsymbol{T}} \quad \approx \quad u(x) = u_0 + u_1 \cdot x + \cdots + u_i \cdot x^i + \cdots + u_{k-1} \cdot x^{k-1}$$

Informační mnohočlen $u(x)$ pak zakódujeme vynásobením generujícím mnohočlenem $g(x)$ **:**

$$v(x) = u(x) \cdot g(x)$$

Návrat k ilustračnímu příkladu Paritní kód délky $n = 4$.

Na základě generujícího polynomu $g(x) = 1 + x$ vytvoříme generující matici $\boldsymbol{G}$. Je zřejmé, že bude mít tři řádky a čtyři sloupce ($n = 4$, stupeň $g(x) = n - k = 1$, tedy $k = 3$).

$$\boldsymbol{G} = \begin{bmatrix} 1 + x \\ x \cdot (1 + x) \\ x^2 \cdot (1 + x) \end{bmatrix} = \begin{matrix} x^0 x^1 x^2 x^3 \\ \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \end{matrix}$$

Podle sloupců generující matice napíšeme „vytvořující rovnice“ pro jednotlivé prvky kódové značky $\boldsymbol{v}$. V našem případě

$$v_1 = u_1 \quad , \quad v_2 = u_1 + u_2 \quad , \quad v_3 = u_2 + u_3 \quad , \quad v_4 = u_3$$

Z rovnic je zřejmé, že kód není systematický, navíc informační prvek u_2 není ve značce obsažen „explicitně“, ale pouze ve formě součtů s jinými informačními prvky.

Zakódujeme informační část $\boldsymbol{u} = [u_0\, u_1\, u_2\,]^{\boldsymbol{T}} = [1\,0\,1\,]^{\boldsymbol{T}}$ **.** Pak

$$u(x) = u_0 + u_1 + u_2 \cdot x^2 = 1 + x^2$$

$$v(x) = u(x) \cdot g(x) = (1 + x^2) \cdot (1 + x) = 1 + x^2 + x + x^3 = 1 + x + x^2 + x^3$$

Značkou odpovídající polynomu $v(x)$ je tedy $\boldsymbol{v} = [v_0\, v_1\, v_2\, v_3\,]^{\boldsymbol{T}} = [1\,1\,1\,1\,]^{\boldsymbol{T}}$

Přerušení ilustračního příkladu

Kontrola přijaté značky

Již jsme poznali, že u cyklických kódů přebírá úlohu generující matice generující mnohočlen. Podobně úlohu kontrolní matice převezme při kontrole přijaté značky kontrolní mnohočlen.

Kontrolní mnohočlen $h(x)$ je polynom stupně k. Je jednoznačně určen mnohočlenem generujícím jako $h(x) = (x^n - 1) : g(x)$ (toto dělení polynomů vyjde beze zbytku, což je jedna z již zmiňovaných vlastností generujícího mnohočlenu).

Každý značkový polynom $v(x)\ \epsilon\ K$ pak vyhovuje podmínce $h(x) * v(x) = 0$, kde $*$ je operace násobení polynomů v okruhu polynomů $\mathrm{Z}_p / x^n - 1$. Kód K tvoří všechny polynomy s touto vlastností, formálně tedy

$$K = \{v(x) \mid v(x) \in \mathrm{Z}_p / x^n - 1 \ \wedge \ h(x) * v(x) = 0\}$$

Kontrola přijaté značky se tedy v (nesystematickém) cyklickém kódu provádí násobením kontrolním polynomem s využitím identity $x^n = 1$. Pokud je výsledkem takového násobení 0, kontrolovaný mnohočlen představuje kódovou značku.

Kontrolní matice cyklického kódu

Kontrolní matici $\boldsymbol{H}$ lze sestavit z koeficientů mnohočlenu $h(x)$ takto:

$$\boldsymbol{H} = \begin{bmatrix} 0 & 0 & 0 & 0 & \cdots & 0 & h_k & \cdots & h_2 & h_1 & h_0 \\ 0 & 0 & 0 & \cdots & 0 & h_k & h_{k-1} & \cdots & h_1 & h_0 & 0 \\ 0 & 0 & \cdots & 0 & h_k & h_{k-1} & \cdots & h_1 & h_0 & 0 & 0 \\ & & & \iddots & \iddots & \iddots & \iddots & \iddots & & & \\ & & \iddots & \iddots & \iddots & \iddots & \iddots & & & & \\ 0 & h_k & h_{k-1} & \cdots & h_1 & h_0 & 0 & 0 & \cdots & 0 & 0 \\ h_k & h_{k-1} & \cdots & h_1 & h_0 & 0 & 0 & \cdots & 0 & 0 & 0 \end{bmatrix}$$

V té podobě, v jaké byly cyklické kódy zatím popsány, jsou prakticky nepoužitelné, protože je nelze jednoduše dekódovat (tj. z přijaté n-tice „extrahovat“ informační část).

Systematické cyklické kódy

Systematické cyklické kódy jsou konstruovány tak, aby v kódové značce byla „zabezpečovací část přilepena za část informační“. To ovšem vyžaduje jinou konstrukci informačního mnohočlenu:

(přiřazení mocnin členů polynomu informačním prvkům)

$$\begin{matrix} k-1 & k-2 & & 1 & 0 \end{matrix}$$

$$\boldsymbol{u} = [u_{k-1}\ u_{k-2} \ldots \ldots u_1\ u_0\ ]^{\boldsymbol{T}} \quad \approx \quad u(x) = u_{k-1} \cdot x^{k-1} + u_{k-2} \cdot x^{k-2} + \cdots + u_1 \cdot x + u_0$$

Při konstrukci informačního mnohočlenu jsme zaměnili přiřazení informačních prvků mocninám proměnné x. Dosud (tedy u nesystematických cyklických kódů) jsme informační prvek v „levé krajní pozici“ interpretovali jako absolutní člen polynomu, zatímco informační prvek v „pravé krajní pozici“ jsme interpretovali jako koeficient u nejvyšší mocniny polynomu.

U systematických cyklických kódů jsme toto přiřazení otočili - informační prvek v „levé krajní pozici“ představuje koeficient u nejvyšší mocniny polynomu, informační prvek v „pravé krajní pozici“ představuje absolutní člen.

Takto vytvořený informační mnohočlen vynásobíme mnohočlenem x^{n-k} :

$$u(x) \cdot x^{n-k} = u_{k-1} \cdot x^{n-1} + u_{k-2} \cdot x^{n-2} + \cdots + u_1 \cdot x^{n-k+1} + u_0 \cdot x^{n-k}$$

Přiřadíme-li tomuto polynomu $n-$ tici, vidíme, že efektem násobení mnohočlenem x^{n-k} je „přilepení“ $n-k$ nul zprava k informační části:

(přiřazení mocnin členů informačním prvkům)

$$\begin{matrix} n-1 & n-2 & & n-k & | & n-k-1 & & 0 \end{matrix}$$

$$u(x) \cdot x^{n-k} \approx [u_{k-1}\ u_{k-2} \ldots \ldots u_1\ u_0 \mid 0\ 0\ 0 \ldots \ldots 0\ ]^T$$

Nyní vydělíme polynom $u(x) \cdot x^{n-k}$ generujícím mnohočlenem $g(x)$. Výsledkem tohoto dělení budou dva mnohočleny – podíl $q(x)$ a zbytek $r(x)$. Stupeň zbytku $r(x)$ je menší než stupeň dělitele (tj. generujícího mnohočlenu $g(x)$), je tedy menší než $n-k$. Podíl $q(x)$ i zbytek $r(x)$ jsou určeny jednoznačně a platí identita

$$u(x) \cdot x^{n-k} = q(x) \cdot g(x) + r(x)$$

Od obou stran této rovnosti odečteme mnohočlen $r(x)$ (tj. přičteme opačný prvek $-r(x)$):

$$u(x) \cdot x^{n-k} - r(x) = q(x) \cdot g(x)$$

Levá strana této rovnosti, tedy mnohočlen $v(x) = u(x) \cdot x^{n-k} - r(x)$ je násobkem generujícího mnohočlenu, **je tedy mnohočlenem reprezentujícím značku cyklického kódu**.

Z grafického znázornění uspořádání obou sčítanců je dále zřejmé, že efektem sečtení členů $u(x) \cdot x^{n-k}$ a $-r(x)$ je požadované „slepení“ obou částí:

u_{k-1}	u_{k-2}	…	…	u_1	u_0	0	…	…	0	0
+						$-r_{n-k-1}$	…	…	$-r_1$	$-r_0$
u_{k-1}	u_{k-2}	…	…	u_1	u_0	$-r_{n-k-1}$	…	…	$-r_1$	$-r_0$

Tímto postupem vytvořená značka je kódovou značkou systematického cyklického kódu.

Už je tedy zřejmé, proč bylo na začátku nutné vynásobit informační polynom právě činitelem x^{n-k} a proč bylo třeba „otočit“ změnu přiřazení mocnin proměnné x informačním prvkům.

Rekapitulace postupu pro kódování v systematickém binárním cyklickém kódu

1. Informační části přiřadíme polynom $u(x)$ (tak, že absolutní člen = **pravý** krajní prvek)
2. Vytvoříme součin $u(x) \cdot x^{n-k}$
3. Vydělíme $u(x) \cdot x^{n-k} : g(x)$. Výsledek: podíl $q(x)$ (k ničemu není) a zbytek $r(x)$.
4. Součet $v(x) = u(x) \cdot x^{n-k} + r(x)$ představuje kódovou značku (tj. zakódovanou informační část).

Kontrola přijaté značky v systematickém binárním cyklickém kódu

Přijatá značka se kontroluje dělením generujícím mnohočlenem. Protože platí, že kódový polynom $v(x) = u(x) \cdot x^{n-k} + r(x)$ je násobkem generujícího mnohočlenu, musí být zbytek po dělení kódového polynomu generujícím mnohočlenem $g(x)$ nulový. Přesněji

$$w(x) \in K \Leftrightarrow w(x) : g(x) = q(x) \text{ a } r(x) = 0$$

Dělení mnohočlenů nad prvočíselnými tělesy prakticky

Nejprve si ukážeme dělení polynomů nad tělesem Z_2 (zde platí 1 + 1 = 0 a -1 = 1, tedy odčítání = sečítání).

$$(x^7 + x^6 + x^5 + x^3 + x^2 + 1) : (x^3 + x^2 + 1) = x^4 + x^2 + 1 \quad = \text{podíl } q(x)$$
$$\underline{-(x^7 + x^6 + x^4)}$$
$$x^5 + x^4 + x^3 + x^2 + 1$$
$$\underline{-(x^5 + x^4 + x^2)}$$
$$x^3 \quad + 1$$
$$\underline{-(x^3 + x^2 + 1)}$$
$$x^2 \quad = \text{ zbytek } r(x)$$

Stejný výpočet můžeme provést přímo nad řetězci vytvořenými z koeficientů polynomů:

11101101 : 1101 = 10101 reprezentuje podíl $q(x) = x^4 + x^2 + 1$

+1101

00111101

0111101

+1101

001001

01001

+1101

0100 reprezentuje zbytek $r(x) = x^2$

Červené cifry představují „posouvaný rámeček“, který vyznačuje stejný počet cifer, jako má dělitel. Je-li první znak v „rámečku“ 1 , připíšeme k podílu 1 , k (postupně modifikovanému) dělenci přičteme dělitele a „posuneme rámeček“. Je-li první znak „v rámečku“ 0 , připíšeme k podílu 0 , a „posuneme rámeček“. Výpočet končí, když má „modifikovaný dělenec“ méně platných cifer než dělitel.

Dělení polynomů nad obecným tělesem Z_p demonstrujeme nad tělesem Z_5 definovaném těmito operacemi:

Sečítání

+	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

Násobení

·	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

Opačné prvky

0	1	2	3	4
0	4	3	2	1

Inverzní prvky

1	2	3	4
1	3	2	4

$$(4x^5 + 3x^4 + 2x^3 + 3x + 2) : (3x^4 + 2x^3 + 4x^2 + 3) = 3x + 4 \quad = \text{ podíl } q(x)$$
$$\underline{-(4x^5 + x^4 + 2x^3 + 4x)}$$
$$2x^4 + 4x + 2$$
$$\underline{-(2x^4 + 3x^3 + x^2 + 2)}$$
$$2x^3 + 4x^2 + 4x \quad = \text{ zbytek } r(x)$$

Koeficienty 3 a 4 u členů podílu vzniknou dělením členů $4x^5 : 3x^4$, respektive $2x^4 : 3x^4$, tedy

$$4x^5 : 3x^4 = (4 : 3)\, x = (4 \cdot 3^{-1})\, x = (4 \cdot 2)\, x = 3x \text{ , respektive}$$
$$2x^4 : 3x^4 = (2 : 3) = (2 \cdot 3^{-1}) = (2 \cdot 2) = 4$$

Odečtení mnohočlenu při modifikaci dělence provedeme tak, že přičteme mnohočlen s opačnými koeficienty.

Stejně jako u dělení mnohočlenů nad tělesem Z_2 můžeme výpočet provést analogicky přímo nad koeficienty mnohočlenu:

$$432032 : 32403 = 34 \quad \text{reprezentuje podíl } q(x) = 3x + 4$$
$$\underline{-41204}$$
$$020042$$
$$\underline{-23102}$$
$$\mathbf{02440} \quad \text{reprezentuje zbytek } r(x) = 2x^3 + 4x^2 + 4x$$

Ilustrační příklad: Cyklické kódy s generujícím mnohočlenem $g(x) = x^3 + x + 1$.

V nesystematickém cyklickém kódu s generujícím mnohočlenem $g(x)$ zakódujeme informační část $\boldsymbol{u} = [1\ 0\ 1\ 1]^{\boldsymbol{T}}$ a to jak pomocí generující matice, tak i pomocí generujícího mnohočlenu.

Nejdříve z koeficientů mnohočlenů $g(x)$ vytvoříme generující matici $\boldsymbol{G}$ (při nesystematickém kódování odpovídá levý krajní prvek značky absolutnímu členu polynomu a pravý krajní prvek koeficientu u nejvyšší mocniny) a spočítáme kódovou značku jako $\boldsymbol{v} = \boldsymbol{G}^{\boldsymbol{T}} \cdot \boldsymbol{u}$:

$$\boldsymbol{G} = \begin{bmatrix} 1101000 \\ 0110100 \\ 0011010 \\ 0001101 \end{bmatrix} \qquad \boldsymbol{v} = \boldsymbol{G}^{\boldsymbol{T}} \cdot \boldsymbol{u} = \begin{bmatrix} 1101000 \\ 0110100 \\ 0011010 \\ 0001101 \end{bmatrix} \begin{bmatrix} \mathbf{1} \\ 0 \\ \mathbf{1} \\ \mathbf{1} \end{bmatrix} \begin{matrix} 1101000 \\ \\ 0011010 \\ 0001101 \end{matrix}$$

$$\boldsymbol{v} = \underline{\qquad\qquad} \\ [\mathbf{1111111}]^{\boldsymbol{T}}$$

Totéž provedeme násobením informačního mnohočlenu $u(x)$ generujícím mnohočlenem $g(x)$:

$$u(x) = 1 + x^2 + x^3$$

$$v(x) = u(x) \cdot g(x) = (1 + x^2 + x^3) \cdot (1 + x + x^3) = (1 + x + x^3) + (x^2 + x^3 + x^5) + \\ + (x^3 + x^4 + x^6) = 1 + x + x^2 + x^3 + x^3 + x^3 + x^4 + x^5 + x^6 = \mathbf{1} + \boldsymbol{x} + \boldsymbol{x^2} + \boldsymbol{x^3} + \boldsymbol{x^4} + \boldsymbol{x^5} + \boldsymbol{x^6}$$

Je samozřejmé, že výsledný mnohočlen $v(x)$ odpovídá značce $\boldsymbol{v}$, kterou jsme získali výpočtem pomocí generující matice $\boldsymbol{G}$.

Zakódujme v systematickém kódu informační část $\boldsymbol{u} = [1\ 1\ 0\ 0]^{\boldsymbol{T}}$. Při systematickém kódování odpovídá levý krajní prvek značky koeficientu u nejvyšší mocniny a pravý krajní prvek absolutnímu členu polynomu.

$$u(x) = x^3 + x^2 \qquad u(x) \cdot x^{n-k} = (x^3 + x^2) \cdot x^3 = x^6 + x^5$$

$$u(x) \cdot x^{n-k} : \ g(x) = (x^6 + x^5) : (x^3 + x + 1) = x^3 + x^2 + x$$

$$\begin{array}{r} \underline{-(x^6 + x^4 + x^3)} \\ x^5 + x^4 + x^3 \\ \underline{-(x^5 + x^3 + x^2)} \\ x^4 + x^3 \\ \underline{-(x^4 + x^3 + x)} \\ \boldsymbol{x} \end{array} \quad = \text{ zbytek } r(x)$$

$$v(x) = u(x) \cdot x^{n-k} + r(x) = \boldsymbol{x^6} + \boldsymbol{x^5} + \boldsymbol{x}$$

Informační části $\boldsymbol{u} = [1\ 1\ 0\ 0]^{\boldsymbol{T}}$ tedy přísluší kódová značka $\boldsymbol{v} = [1\ 1\ 0\ 0\ 0\ 1\ 0]^{\boldsymbol{T}}$.

V tabulce na další stránce jsou uvedeny všechny kódové značky systematického i nesystematického kódování s generujícím mnohočlenem $g(x) = x^3 + x + 1$. Tyto značky bychom získali, kdybychom výše uvedenými postupy zakódovali všechny informační části od

$\boldsymbol{u} = [0\ 0\ 0\ 0]^{\boldsymbol{T}}$ do $\boldsymbol{u} = [1\ 1\ 1\ 1]^{\boldsymbol{T}}$. Na množinách značek si ověříme některé vlastnosti cyklických kódů.

č.	Informační část **u**				Nesystematické kódování kódová značka **v**							Systematické kódování kódová značka **v**							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	1	0	1	0	0	0	1	0	1	1	8
2	0	0	1	0	0	0	1	1	0	1	0	0	0	1	0	1	1	0	4
3	0	0	1	1	0	0	1	0	1	1	1	0	0	1	1	1	0	1	12
4	0	1	0	0	0	1	1	0	1	0	0	0	1	0	0	1	1	1	10
5	0	1	0	1	0	1	1	1	0	0	1	0	1	0	1	1	0	0	2
6	0	1	1	0	0	1	0	1	1	1	0	0	1	1	0	0	0	1	14
7	0	1	1	1	0	1	0	0	0	1	1	0	1	1	1	0	1	0	6
8	1	0	0	0	1	1	0	1	0	0	0	1	0	0	0	1	0	1	13
9	1	0	0	1	1	1	0	0	1	0	1	1	0	0	1	1	1	0	5
10	1	0	1	0	1	1	1	0	0	1	0	1	0	1	0	0	1	1	9
11	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1	0	0	0	1
12	1	1	0	0	1	0	1	1	1	0	0	1	1	0	0	0	1	0	7
13	1	1	0	1	1	0	1	0	0	0	1	1	1	0	1	0	0	1	15
14	1	1	1	0	1	0	0	0	1	1	0	1	1	1	0	1	0	0	3
15	1	1	1	1	1	0	0	1	0	1	1	1	1	1	1	1	1	1	11

Jak systematický, tak i nesystematický kód jsou opravdu cyklické. Množiny kódových značek uzavřené vzhledem k cyklickým posuvům jsou u obou kódů barevně označeny. Značky $[0\ 0\ 0\ 0\ 0\ 0\ 0]^{\mathrm{T}}$ a $[1\ 1\ 1\ 1\ 1\ 1\ 1]^{\mathrm{T}}$ jsou vůči cyklickému posuvu invariantní, tvoří tedy dvě jednoprvkové množiny, další dvě množiny jsou sedmiprvkové (můžeme je chápat jako „mateřskou značku“ a jejích šest posuvů o jednu pozici, protože sedmým posuvem dostaneme opět „mateřskou značku“).

Při bližším zkoumání zjistíme, že každá značka nesystematického kódu má svůj reverzní obraz v množině kódových značek systematického kódu (modrá čísla za sytematickými značkami ukazují, reverzí které nesystematické značky je daná systematická značka; např. reverzí systematické značky č. 13 je nesystematická značka č. 15). Je to důsledkem toho, že jsme při přechodu k systematickým cyklickým kódům „obrátili“ přiřazení koeficientů mnohočlenu prvkům značky.

Z množiny systematických značek můžeme vybrat značky, z nichž následně vytvoříme systematickou generující matici – $[1\ 0\ 0\ 0\ 1\ 0\ 1]^{\mathrm{T}}$, $[0\ 1\ 0\ 0\ 1\ 1\ 1]^{\mathrm{T}}$, $[0\ 0\ 1\ 0\ 1\ 1\ 0]^{\mathrm{T}}$, $[0\ 0\ 0\ 1\ 0\ 1\ 1]^{\mathrm{T}}$:

$$\boldsymbol{G} = \begin{bmatrix} 1000\ 101 \\ 0100\ 111 \\ 0010\ 110 \\ 0001\ 011 \end{bmatrix}, \quad \text{tedy} \quad \boldsymbol{H} = [-\boldsymbol{B}^{\boldsymbol{T}} | \boldsymbol{I}_{\boldsymbol{n-k}}] = \begin{bmatrix} 1110\ 100 \\ 0111\ 010 \\ 1101\ 001 \end{bmatrix}$$

Tyto matice jsou generující a kontrolní maticí *cyklického Hammingova kódu* (7,4) (od Hammingova kódu (7,4) z kapitoly 2.3.4 se liší pouze permutací řádků „zabezpečovací submatice“ B . To znamená, že tento kód opravuje jednoduché chyby.

Ukážeme jeho další vlastnost, která vyplývá z cykličností. Spočítejme syndromy pro tři konkrétní chybová slova:

$$\boldsymbol{e_1} = [1100000]^{\boldsymbol{T}} \qquad \boldsymbol{s_1} = \boldsymbol{H} \cdot \boldsymbol{e_1} = [010]^{\boldsymbol{T}}$$
$$\boldsymbol{e_2} = [1010000]^{\boldsymbol{T}} \qquad \boldsymbol{s_2} = \boldsymbol{H} \cdot \boldsymbol{e_2} = [011]^{\boldsymbol{T}}$$

$$\boldsymbol{e_3} = [1110000]^{\boldsymbol{T}} \qquad \boldsymbol{s_3} = \boldsymbol{H} \cdot \boldsymbol{e_3} = [100]^{\boldsymbol{T}}$$

Chybový vektor $\boldsymbol{e_1}$ je příkladem shlukové chyby délky 2, vektory $\boldsymbol{e_2}$ a $\boldsymbol{e_3}$ jsou příklady shlukových chyb délky 3. Přesněji: *shlukovou chybou délky* b rozumíme takové chybové slovo $\boldsymbol{e}$, jehož všechny chybové prvky (u binárních kódů jedničky) leží v úseku ohraničeném indexy i a $i + b - 1$, přičemž „krajní prvky shluku" e_i a e_{i+b-1} jsou jedničky.

Jestliže je v libovolném cyklickém kódu slovo $\boldsymbol{w}$ nekódovým slovem, je i jeho libovolný cyklický posuv nekódovým slovem (lze snadno dokázat sporem). Jinak řečeno – také množina nekódových slov je uzavřená vzhledem k cyklickému posuvu.

Vektory $\boldsymbol{e_1}$, $\boldsymbol{e_2}$ a $\boldsymbol{e_3}$ tedy jsou reprezentanty všech shluků chyb délky $b \leq 3$ (pomineme jednoduché chyby; 11 je jediný „vzorec" shluku délky $b = 2$, 101 a 111 jediné dva „vzorce" shluků délky $b = 3$). Ukázali jsme, že shluky $\boldsymbol{e_1}$, $\boldsymbol{e_2}$ a $\boldsymbol{e_3}$ generují nenulové syndromy. Díky cykličnosti jsou nekódovými slovy i všechny jejich cyklické posuvy, budou tedy také generovat nenulové syndromy. Cyklický Hammingův kód (7,4) tedy detekuje všechny shluky chyb délky $b \leq 3$.

Softwarová implementace kodérů/dekodérů systematických binárních cyklických kódů

Základní princip spočítá v implementaci dělení nad tělesem Z_2, jak jsem je poznali v předchozím textu. K urychlení výpočtu se používá pro konkrétní generující mnohočlen předem vypočítaná tabulka CRC_table, která obsahuje „zbytky po dělení" všech „dělenců" od 0 do 255:

```
for (dividend = 0; dividend < 256; ++dividend) {
 remainder = dividend << (WIDTH-8);/*posunutí dělitele*/
 for (bit = 8; bit > 0; --bit) {
  if (remainder & TOPBIT) {  /* je-li v nejvyšším bitu 1 */
   remainder = (remainder << 1) ^ POLYNOMIAL; /*posuň, přičti g(x)*/
  }
  else {
   remainder = (remainder << 1); /*posuň rámeček*/
  }
 }
crcTable[dividend] = remainder; /*zapiš zbytek do CRC_table*/
}
```

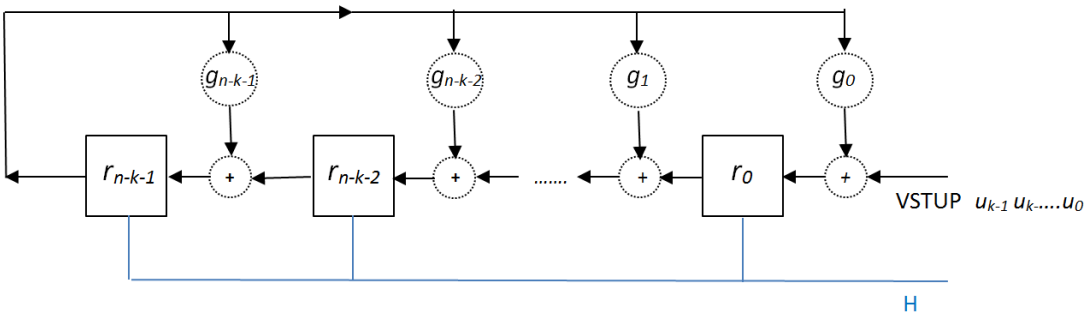
CRC_table se pak dál používá tím způsobem, že se jednotlivé byty (tedy osmice bitů) z datového bloku použijí jako index do CRC_table a „zbytky" z CRC_table se „průběžně XORují" datovými byty takto:

```
for (byte = 0; byte < nBytes; ++byte) {
  data = message[byte]^(remainder >> (WIDTH - 8));
  remainder = crcTable[data] ^ (remainder << 8);
}
```

Cennou pomůckou pro pochopení softwarové implementace může být zdroj [BARR], který dává k dispozici volně šiřitelný zdrojový kód pro implementaci v praxi nejpoužívanějších cyklických kódů 0x8005, 0x1021 a 0x04C11DB7 (viz odstavec Praktické použití systematických binárních cyklických kódů).

Hardwarová realizace kodérů systematických binárních cyklických kódů

Základním prvkem realizace je *lineární zpětnovazební posuvný registr LFSR (linear-feedback shift register).* Nastavení zpětných vazeb registru je určeno použitým generujícím mnohočlenem. LFSR pak realizuje dělení tímto generujícím mnohočlenem. Obecné schéma lze znázornit takto:



Popis funkce prvků schématu:

synchronní binární paměťový prvek typu D („buzení = příští stav“)

S náběžnou hranou pulzu hodinového signálu H se do prvku zapíše hodnota, která je na jeho vstupu, a současně se tato hodnota objeví i na výstupu prvku.

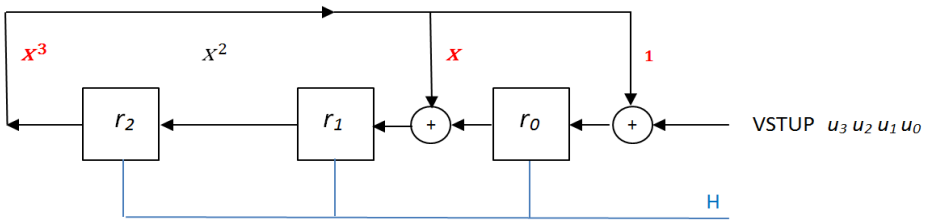
„Kvazipropojka“. Je-li koeficient g_i v generujícím mnohočlenu 1, je zpětná vazba přivedena na součtové hradlo před vstup příslušného paměťového členu. Je-li koeficient g_i roven 0, součtové hradlo před vstupem příslušného paměťového členu není.

Součtové hradlo, realizuje sčítání definované v tělese Z_2 (názvy v elektrotechnické terminologii - hradlo XOR, Executive OR, Nonekvivalence)

H Zdroj hodinového signálu, který definuje okamžik zápisu vstupů do paměťových prvků a synchronizuje vstup jednotlivých bitů dělence.

Zprava do registru vstupuje informační část následovaná $n-k$ nulami (dělenec). S každým hodinovým pulzem dojde ke vstupu jednoho znaku a k zápisu do paměťových prvků r_{n-k-1} až r_0 . Po *n* hodinových pulzech je v paměťových prvcích $r_{n-k-1}, r_{n-k-2}, \ldots\ldots, r_1, r_0$ zapsán zbytek po dělení $u(x)\cdot x^{n-k} : g(x)$.

Konkrétní uspořádání LFSR pro generující mnohočlen $g(x) = x^3 + x + 1$ a průběh zpracování konkrétního vstupního řetězce po jednotlivých taktech ukazuje následující obrázek a tabulka:



,

	1	2	3	4	5	6	7	8	9		
	0	0	**0**	**0**	**0**	**0**	**0**	**1**	**1**	u_3	**stav před prvním hodinovým pulzem**
	0	0	**0**	**0**	**1**	**1**	**1**	**1**	**1**	u_2	**stav po 1. pulzu**
	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	u_1	**stav po 2. pulzu**
q_3	**1**	1	**1**	**1**	**0**	**1**	**1**	**1**	**0**	u_0	**stav po 3. pulzu**
q_2	**1**	1	**0**	**0**	**0**	**1**	**1**	**1**	**0**		**stav po 4. pulzu**
q_1	**0**	0	**0**	**0**	**1**	**1**	**1**	**0**	**0**		**stav po 5. pulzu**
q_0	**0**	0	**1**	**1**	**0**	**0**	**0**	**0**	**0**		**stav po 6. pulzu**
	1	1	**0**	**0**	**1**	**0**	**0**	**1**	0		**stav po 7. pulzu**
		r_2		r_1			r_0				

Co říkají sloupce tabulky?

Sloupec 1 - výstup do zpětné vazby
Sloupec 2 – obsah prvku r_2
Sloupec 3 – výstup z prvku r_1
Sloupec 4 - obsah prvku r_1
Sloupec 5 – výstup z hradla XOR (= vstup r_1)
Sloupec 6 – výstup z prvku r_0
Sloupec 7 - obsah prvku r_0
Sloupec 8 – výstup z hradla XOR (= vstup r_0)
Sloupec 9 – vstup

V tabulce jsou žlutě podbareveny cifry vstupujícího dělence, zeleně pak cifry reprezentující cifry podílu. Stavy paměťových prvků v jednotlivých taktech jsou podbarveny růžově. Je zřejmé, že je zapotřebí $n-k$ hodinových pulsů, aby se jednička představující řád dělence dostala do paměťového prvku s nejvyšším indexem a následujících $n-k-1$ cifer dělence do dalších paměťových prvků. Teprve tehdy se začne projevovat vliv zpětné vazby (objeví se v ní jednička reprezentující řád podílu). Při každém dalším hodinovém pulzu je do zpětné vazby vyslána další cifra podílu. Po n pulzech jsou v paměťových prvcích zapsány koeficienty zbytku po dělení.

Zpracování popsané v tabulce tedy provedlo dělení mnohočlenů reprezentovaných slovy 1110000 (dělenec) a 1011 (dělitel) s výsledkem 1100 (podíl) a 100 (zbytek). Provedeme tento výpočet „ručně“:

$1110000 : 1011 = 1100$ reprezentuje podíl $q(x) = x^3 + x^2$

$+\ 1011$

0101000

$+1011$

000100

00100

0100 reprezentuje zbytek $r(x) = x^2$

Když porovnáme tento výpočet s tabulkou popisující zpracování vstupního řetězce ve zpětnovazebním registru, uvědomíme si, že od čtvrtého taktu (tj. pod čarou) je obsah paměťových prvků (rozšířený o symbol ze vstupního řetězce, jenž je o řádek výš) roven červeně zvýrazněnému posouvajícímu se „rámečku“ z ručního výpočtu. Je to dáno tím, že zpětná vazba realizuje právě přičítání hodnoty (0 nebo 1) · 1011 k „modifikovanému“ dělenci.

Stejný LSFR slouží jak ke kódování informační části, tak i ke kontrole přijaté značky. Při kódování do registru vstupuje k- prvková informační část doplněná $(n-k)$ nulami, po n hodinových pulzech jsou v registru kontrolní prvky (zbytek po dělení). V případě kontroly přijaté značky do registru vstoupí přijatá n-tice, pokud jsou po n hodinových pulzech v paměťových prvcích samé nuly, byla kontrolovaná n-tice kódovou značkou.

Praktické použití systematických binárních cyklických kódů

Systematické cyklické kódy (CRC – Cyclic Redundance Check) jsou používány pro zabezpečení dat v řadě komunikačních protokolů. Používají se pouze k detekci chyb, nikoli k jejich opravování. Všechny cyklické kódy Jsou schopné detekovat 100% shluků délky $b \leq n-k$ a vysoké procento shluků delších.

Nejpoužívanější standardizované kódy představuje následující tabulka:

Název	generující mnohočlen	standardní reprezentace	použití (standardy)
CRC-8	$x^8 + x^2 + x + 1$	0x07	vysokorychlostní protokol ATM (záhlaví)
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x + 1$		vysokorychlostní protokol ATM (vrstva AAL)
CRC-12	$x^{12} + x^{11} + x^3 + x^2 + x + 1$		Telekomunikační systémy
CRC-16	$x^{16} + x^{15} + x^2 + 1$	0x8005	„firemní“ protokol IBM
CCITT-16	$x^{16} + x^{12} + x^5 + 1$	0x1021	HDLC, X.25, V41, IEEE 802, V.42, AAL 5
CRC-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + +x^{12} + x^{11} + x^{10} + x^8 + x^7 + +x^5 + x^4 + x^2 + x + 1$	0x04C11DB7	IEE 802.3, Ethernet

Poznámka: Pod pojmem *standardní reprezentace* rozumíme reprezentaci generujícího mnohočlenu šestnáctkovým číslem. Vyjadřují se tak hlavně mnohočleny, jejichž stupeň je násobkem 8, což umožňuje zápis binárních koeficentů do osmic. Koeficient u členu s nejvyšší moc-

ninou je vždy 1, v hexadecimálním čísle jsou zobrazovány pouze koeficienty u mocnin stupně $n-k-1$ a nižšího, například

x^{16}		+		x^{12}				+			x^5			+		1
16	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1
	1				**0**				**2**				**1**			

Existují ale i alternativní způsoby reprezentace (například Koopmanův, který naopak nezobrazuje koeficient 1 u absolutního členu, takže „rámeček s červenými čísly posune o 1 bit doleva“ a polynom je pak popsán jako 0x8810) .

V komunikačních adaptérech jsou kódovací a dekódovací postupy implementovány hardwerově, ve vyšších vrstvách architektur podle modelu ISO/OSI sotwarově.

Při praktickém použití cyklických kódů nemusí mít kódové značky stejnou délku (při kódování je dělencem obvykle datový blok doplněný na sudou délku následovaný $n-k$ nulami), nicméně každý CRC polynom má standardem definovanou maximální délku bloku dat, jež může zabezpečit.

Je-li generující mnohočlen stupně $n-k$ primitivní, je tato maximální délka bloku rovna $2^{n-k}-1$ bitů. Primitivní mnohočlen dokáže kromě všech shluků délky menší nebo rovné $n-k$ detekovat i všechny jednoduché a dvojité chyby.

Je-li generující mnohočlen součinem primitivního mnohočlenu stupně $n-k-1$ a mnohočlenu $(x+1)$, je maximální délka bloku rovna $2^{n-k-1}-1$ bitů. Takový generující mnohočlen ovšem dokáže kromě všech shluků délky menší nebo rovné $n-k$, jednoduchých a dvojitých chyb detekovat i všechny chyby trojité, všechna chybová slova se sudým počtem jedniček a s pravděpodobností $p=1-2^{n-k-1}$ i náhodné shluky s délkou větší než $n-k$.

V tabulce zmiňované standardy často v této kapitole popsaný postup kódování informační části modifikují některým z následujících způsobů:

- počáteční stav paměťových prvků při hardwarové realizaci nejsou nuly, ale jedničky
- kontrolní znaky (tj. zbytek po dělení) se před připojením za datový blok invertují (tj. negují „bit po bitu“)
- před připojením za datový blok se provede reverze řetězce kontrolních znaků (tj. řetězec kontrolních znaků se připojí „pozpátku“)

Použitá literatura

Adámek, J.: Kódování, SNTL 1989.

Barr, M.: CRC Implementation Code in C/C++ (https://barrgroup.com/Embedded-Systems/How-To/CRC-Calculation-C-Code)

Beran, M.: Kodér/dekodér Reed-Mullerových kódů. Bakalářská práce. FEI, Univerzita Pardubice, 2014.

Koscielny, C.: Extended (24, 12) Binary Golay Code: Encoding and Decoding Procedures (https://www.maplesoft.com/applications/view.aspx?SID=1757&view=html), 2006.

Koopman, P.: 32-Bit Cyclic Redundancy Codes for Internet Applications. In: The International Conference on Dependable Systems and Networks (DSN) 2002.

Prchal, J.: Teorie pravděpodobnosti ve sdělovací technice, NADAS 1976.

Wallace, H.: Using The Golay Error Detection And Correction Code (http://aqdi.com/articles/using-the-golay-error-detection-and-correction-code-3/), 2003.

Wikipedia, the free encyclopedia.

Zeman, V.: Techniky protichybového zabezpečení v digitální komunikaci pro integrovanou výuku VUT a VŠB – TUO. (https://vut-vsb.cz/home/get-file?file=475), 2014.