


# Úvod do kódování

- samoopravné kódy: terminologie, princip
- blokové lineární kódy
- Hammingův kód
- cyklické kódy

a) kody, 18, b) P. Olsák, FEL ČVUT, c) P. Olsák 2010, d) BI-LIN, e) L, f) 2009/2010, g)  Viz p. d. 4/2010

## Samoopravné kódy, k čemu to je

- Data jsou uložena (nebo posílána do linky) *kodeřem* podle určitého pravidla (*kódování*). Posléze jsou čtena *dekodeřem* a restaurována do původní podoby.
- Kodeř může přidat k datům doplňující informaci (zhruba řečeno kontrolní součet) a umožnit tím dekodeřu, aby poznal, zda při přenosu dat došlo k chybě. Dokonce při vhodně zvoleném kódování může dekodeř chybu opravit.

*Kód* je množina slov (tj. úseků dat), které může generovat kodeř.

### Příklady kódů:

- ASCII (slova sedmibitová, ne všechna)
- Morseova abeceda (slova různě dlouhá, efektivní přenos)
- UTF-8 (slova různě dlouhá, délka rozpoznána podle prefixu)

BI-LIN, kody, 18, P. Olsák [3]

## Binární, blokový kód

je kód, kde jsou všechna slova stejně dlouhá.

**Definice:** Necht'  $A$  je množina znaků (abeceda). Slovo je konečná posloupnost znaků z množiny  $A$ . Počet znaků ve slově je *délka slova*.

*Kód*  $K$  je množina všech slov, která generuje kodeř.

Prvek kódu  $K$  se nazývá *kódové slovo*.

*Blokový kód*  $K$  obsahuje jen slova stejné délky.

*Binární kód* je kód se slovy nad abecedou  $A = \{0, 1\}$ .

### Příklady:

- ASCII je binární blokový kód délky 7.
- Moreseovka není binární a není blokový kód.
- UTF-8 je binární, ale ne blokový kód.

Dále se budeme zabývat jen binárními blokovými kódy

BI-LIN, kody, 18, P. Olsák [4]

## Lineární kód

Binární blokový kód  $K$  délky  $n$  je podmnožinou lin. prostoru  $\mathbf{Z}_2^n$ .

**Definice:** Je-li  $K$  lineární podprostor  $\mathbf{Z}_2^n$ , pak se kód nazývá *lineární*. Je-li dimenze kódu  $k$ , pak mluvíme o lineárním  $(n, k)$  kódu.

**Příklad:** Kód s kontrolním bitem parity je lineární. Kodeř přidává nulu nebo jedničku k informačním bitům tak, aby kódové slovo obsahovalo sudý počet jedniček. Množina všech slov délky  $n$  se sudým počtem jedniček je lineární podprostor lineárního prostoru  $\mathbf{Z}_2^n$ .

## Generující a kontrolní matice

*Generující matice* lineárního kódu  $K$  je matice, která v řádcích obsahuje bázi kódu.

*Kontrolní matice* lineárního kódu  $K$  je matice  $\mathbf{H}$ , pro kterou platí, že  $K$  je řešením soustavy  $\mathbf{H}\mathbf{x} = \mathbf{o}$ .

**Příklad:** Předpokládejme lineární  $(4, 3)$  kód s kontrolním bitem parity (přidávaný na konec slova za tři informační bity). Generující matice  $\mathbf{G}$  a kontrolní matice  $\mathbf{H}$  jsou:

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}, \quad \mathbf{H} = (1 \ 1 \ 1 \ 1)$$

**Pozorování:** Generující matice  $(n, k)$  kódu je typu  $(k, n)$  a kontrolní matice je typu  $(n - k, n)$ . Platí:  $\mathbf{G} \cdot \mathbf{H}^T = \mathbf{O}$ .

## Kodér a dekodér

Kodér lin.  $(n, k)$  kódu může převzít kódované slovo  $\vec{u}$  (délky  $k$ ) a vytvořit z něj kódové slovo  $\vec{v}$  (délky  $n$ ) maticovým násobením:

$$\vec{v} = \vec{u} \cdot \mathbf{G}.$$

Dekodér může zkontrolovat přijaté slovo  $\vec{w}$  pomocí testu:

$$\mathbf{H} \cdot \vec{w}^T = \mathbf{o}.$$

Kódování je *systematické*, jsou-li informační bity (ze slova  $\vec{u}$ ) beze změny zkopírovány do kódového slova a za nimi následují kontrolní bity. Pak může dekodér (po provedeném testu) rekonstruovat informační bity zkopírováním prvních  $k$  pozic přijatého slova.

**Pozorování:** Kódování je systematické, je-li generující matice tvaru  $\mathbf{G} = (\mathbf{E} | \mathbf{C})$ . Přidávané kontrolní bity pak kodér spočítá pomocí vzorce  $\vec{v}' = \vec{u} \cdot \mathbf{C}$ .

## Výpočet jedné matice, známe-li druhou

Je-li dána generující (resp. kontrolní) matice, vyřešíme homogenní soustavu rovnic s touto maticí a bázi řešení zapíšeme do řádků kontrolní (resp. generující) matice.

Pro systematický kód dokonce platí:

Je-li  $\mathbf{G} = (\mathbf{E} | \mathbf{C})$ , pak  $\mathbf{H} = (\mathbf{C}^T | \mathbf{E}')$ .

## Příklad: opakovací kód

Kodér vezme kódované slovo  $\vec{u}$  délky  $k$  a vytvoří kódové slovo délky  $n = 2k$  tak, kódové slovo je tvaru  $(\vec{u}, \vec{u})$ , tj. kódované slovo je zdvojené.

Generující matice tohoto kódu je  $\mathbf{G} = (\mathbf{E} | \mathbf{E})$  a kontrolní matice je také tvaru  $\mathbf{H} = (\mathbf{E} | \mathbf{E})$ . Uvědomte si, jak je kód pomocí  $\mathbf{G}$  generován a jak je pomocí  $\mathbf{H}$  kontrolován.

**Nevýhoda:** příliš mnoho kontrolních bitů za „málo muziky“.

## Hammingova váha, vzdálenost

**Definice:** *Hammingova váha* slova  $\vec{v}$  je počet jeho nenulových znaků. *Hammingova vzdálenost* dvou slov  $\vec{v}$  a  $\vec{w}$  je počet pozic, kde jsou znaky odlišné (pro binární kód je to váha slova  $\vec{v} + \vec{w}$ ).

Kód  $K$  objevuje  $t$  chyb, pokud pro každé slovo  $\vec{u} \in K$  a každé slovo  $\vec{e}$  váhy menší nebo rovno  $t$  platí  $\vec{u} + \vec{e} \notin K$ .

Kód  $K$  opravuje  $t$  chyb, pokud pro každé slovo  $\vec{u} \in K$  a každé slovo  $\vec{e}$  váhy menší nebo rovno  $t$  platí: slovo  $\vec{u}$  má od slova  $\vec{u} + \vec{e}$  nejmenší vzdálenost mezi kódovými slovy.

**Tvrzení 1:** Je-li nejmenší vzdálenost mezi kódovými slovy  $d$ , pak kód objevuje  $d - 1$  chyb a opravuje  $t < \frac{d}{2}$  chyb.

**Tvrzení 2:** Nejmenší vzdálenost mezi kódovými slovy *lineárního* kódu je rovna nejmenší váze nenulového kódového slova.

## Příklady

Kód s kontrolním bitem parity má nejmenší váhu nenulového slova 2, takže objevuje  $2 - 1 = 1$  chybu ve slově. Opravuje méně než  $2/2$  chyb, tedy neopravuje žádnou chybu.

Opakovací kód má rovněž nejmenší váhu nenulového slova 2.

Aby kód dokázal opravit jednu chybu ve slově, musí mít nejmenší váhu nenulového slova rovno třem.

## Syndrom

Dekodér vyhodnotí  $\mathbf{s} = \mathbf{H} \cdot \vec{w}^T$ . Tomuto vektoru  $\mathbf{s}$  říkáme *syndrom* přijatého slova  $\vec{w}$ . Přijaté slovo je kódové, právě když má nulový syndrom.

Kód rozpozná chybu  $\vec{e}$ , právě když  $\mathbf{s} = \mathbf{H} \cdot \vec{e}^T$  je nenulový vektor.

**Pozorování 1:** Syndrom nezávisí na kódovém slově (jen na chybovém slově):  $\mathbf{H} \cdot (\vec{v} + \vec{e})^T = \mathbf{H} \cdot \vec{v}^T + \mathbf{H} \cdot \vec{e}^T = \mathbf{0} + \mathbf{H} \cdot \vec{e}^T = \mathbf{H} \cdot \vec{e}^T$ .

**Pozorování 2:** Lin. kód má minimální vzdálenost dvou slov  $d$ , právě když každý výběr  $d - 1$  sloupců z kontrolní matice  $\mathbf{H}$  je lineárně nezávislý.

Jmenovitě: kód opravuje jednu chybu když každé dva sloupce kontrolní matice  $\mathbf{H}$  jsou LN, tj. jsou nenulové a vzájemně různé (to v  $\mathbf{Z}_2^{n-k}$  stačí). Kontrolní matice s touto vlastností je kontrolní matice *Hammingova kódu*.

## Hammingův kód

Sloupce kontrolní matice  $\mathbf{H}$  jsou prvky  $\mathbf{Z}_2^{n-k}$ . Počet nenulových a vzájemně různých sloupců je maximálně  $2^{n-k} - 1$ . Počet sloupců udává délku kódu  $n$ , tedy  $n = 2^{n-k} - 1$ . Délku kódu je tedy vhodné volit jako mocninu dvou bez jedné. Dostáváme tak Hammingovy kódy: (7, 4), (15, 11), (31, 26), (63, 57), ...

Příklad: Hammingův kód (7, 4) – délka 7, informační bity 4:

$$\mathbf{H} = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}, \quad \mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Výhoda tohoto uspořádání: index bitu, který je potřeba opravit, je zapsán v syndromu jako číslo ve dvojkové soustavě.

## Rozšířený Hammingův kód

je Hammingův kód, ke kterému kodér přidává kontrolní bit parity. Například (8, 4) kód má matice

$$\mathbf{H} = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}, \quad \mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}.$$

Kód opraví jednu chybu (v prvních třech bitech je syndrom jako v (7, 4) kódu a čtvrtý bit musí být 1) a odhalí dvě chyby (v prvních třech bitech syndromu je nenulové číslo a čtvrtý bit je 0).

Nejmenší vzdálenost dvou slov v tomto kódu je 4.

## Návrh počtu kontrolních bitů

Označme  $n$  délku binárního kódu,  $k$  dimenzi kódu (počet informačních bitů) a  $c = n - k$  počet kontrolních bitů.

Lineární kód nemůže opravit více rozdílných chyb než je počet nenulových syndromů. Těch je  $2^c - 1$ . Počet různých chyb s váhou jedna je  $n$ . Proto, chceme-li opravit jednu chybu, musí  $2^c - 1 \geq n$ .

Počet různých chyb (včetně stavu „bez chyby“) s váhou nejvýše  $m$  je rovno

$$\binom{n}{0} + \binom{n}{1} + \dots + \binom{n}{m}$$

Chceme-li opravovat  $m$  chyb ve slově, musí tedy počet kontrolních bitů splňovat:

$$2^c \geq \binom{n}{0} + \binom{n}{1} + \dots + \binom{n}{m}$$

Kódy navržené tak, že zde nastává rovnost, se nazývají *perfektní*.

## Cyklické kódy

jsou běžně užívané samoopravné kódy (např. při zápisu/čtení CD). Viz google: CRC (cyclic redundancy check).

**Definice:** Kód  $K$  se nazývá *cyklický*, pokud

- je lineární a navíc
- je-li  $\vec{v}$  kódové slovo, pak cyklický posun  $\vec{v}$  je také kódové slovo.

Vhodná matematická reprezentace slov délky  $n$  jsou polynomy:

$$\vec{v} = (a_0, a_1, a_2, \dots, a_{n-1}) \leftrightarrow v(x) = a_0 + a_1x + a_2x^2 + \dots + a_{n-1}x^{n-1}$$

Cyklický posun slova  $\vec{v}$  o jednu pozici popíšeme násobením polynomu  $v(x)$  polynomem  $x$  a ztotožněním  $x^n = x^0$ , neboli násobením v okruhu  $\mathbf{Z}_p[x]/(x^n - 1)$ .

Od této chvíle nerozlišujeme mezi pojmem „slovo“ a „polynom“.

## Základní vlastnosti cyklického kódu

**Definice:** Nenulový polynom cyklického kódu nejmenšího stupně nazýváme *generující polynom*.

**Tvrzení:** Je-li  $K$  cyklický kód délky  $n$  a  $g$  je jeho generující polynom, pak

$$K = \{f \cdot g; \text{st } f < n - \text{st } g\}$$

Důkaz:  $v = (f_{m-1}x^{m-1} + \dots + f_1x + f_0) \cdot g$  je lineární kombinace cyklických posunů, takže leží v  $K$ .

Obráceně, necht'  $v \in K$ , pak vydělíme  $v$  polynomem  $g$  se zbytkem:  $v = f \cdot g + z$ , protože  $v \in K, f \cdot g \in K$ , musí  $z \in K$ .

Protože  $\text{st } z < \text{st } g$  a polynom  $g$  má nejmenší stupeň, musí  $z = 0$ .

## Vlastnosti generujícího polynomu

**Tvrzení:** Necht'  $g$  je generující polynom  $(n, k)$  cyklického kódu  $K$ .

- polynom  $g$  má stupeň  $n - k$ ,
- $\{g, x \cdot g, x^2 \cdot g, \dots, x^{k-1} \cdot g\}$  je báze kódu,
- polynom  $x^n - 1$  je dělitelný polynomem  $g$ .

Důkaz:  $K = \{f \cdot g; \text{st} f < n - \text{st} g\}$ , takže  $\dim K = n - \text{st} g$ , neboli stupeň  $g = n - \dim K$ .

Puntík druhý: zřejmé.

Puntík třetí: polynom  $x^n - 1$  vydělíme polynomem  $g$  se zbytkem. V polynomech mod  $x^n - 1$  je zbytek roven  $-f \cdot g$ , takže leží v kódu a má menší stupeň, než  $g$ , takže zbytek je nulový.

## Generující polynom: postačující podmínka

**Tvrzení:** Aby byl polynom  $g$  generující polynom nějakého cyklického kódu, stačí, aby dělil polynom  $x^n - 1$  beze zbytku.

Důkaz: Zjistíme, že lin. obal všech cyklických posunů  $g$  neobsahuje nenulový polynom st. menšího než  $g$ . Necht'  $f$  je libovolný polynom.

$$f \cdot g = z \pmod{x^n - 1}, \quad \text{tj. } f \cdot g = u \cdot (x^n - 1) + z$$

Je třeba ověřit, že  $z = 0$  nebo  $\text{st} z \geq \text{st} g$ . Protože je  $f \cdot g$  dělitelný  $g$  a  $u \cdot (x^n - 1)$  je dělitelný  $g$ , musí též  $z$  být dělitelný  $g$ , takže  $z = v \cdot g$ .

**Návrh cyklického kódu:** Zvolíme délku bloku  $n$ , rozložíme polynom  $x^n - 1$  na součin ireducibilních polynomů a generující polynom  $g$  zvolíme jako součin *některých* takto nalezených ireducibilních polynomů. Stupeň  $g$  je počet kontrolních bitů kódu.

**Úmluva:** Všechny gen. polynomy stejného kódu se liší až na skalární násobek. Volme takový, co má u nejvyšší mocniny jedničku.

## Odhalení souvislé chyby

*Souvislá chyba délky  $t$*  je chyba měnící kódové slovo v úseku některých po sobě jdoucích  $t$  bitů, jinde je slovo nezměněno. Počet chyb (váha chybového slova) nemusí být  $t$ , ale je menší nebo rovna  $t$ .

**Pozorování:** Cyklický  $(n, k)$  kód odhaluje všechny souvislé chyby délky  $n - k$ .

Důkaz: Na souvislou chybu  $\vec{e}$  můžeme provést (opakovaně) cyklický posun a získat polynom  $\vec{e}'$ , který je stupně menšího než  $n - k$ . Takže  $\vec{e}'$  ani  $\vec{e}$  není kódové slovo.

**Poznámka:** toto je důvod, proč se v praxi používají cyklické kódy. Chyby se totiž rády v konkrétním technickém prostředí soustřeďují do bloků (drupouty, škrábance na CD atd.).

Existují cyklické  $(n, k)$  kódy, které navíc *umějí opravit* všechny souvislé chyby délky  $(n - k)/2$ .

## Příklad: Cyklický Hammingův kód

Sestavme  $(7, 4)$  cyklický kód, který má generující polynom  $x^3 + x + 1$ . Je to generující polynom, protože dělí polynom  $x^7 - 1$ . Kód má následující generující a kontrolní matici

$$\mathbf{G} = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}, \quad \mathbf{H} = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix},$$

takže vidíme, že  $\mathbf{H}$  má různé a nenulové sloupce. Je to tedy Hammingův  $(7, 4)$  kód.

Hammingův  $(7, 4)$  kód, který kóduje podle této  $\mathbf{G}$  a používá tuto kontrolní matici  $\mathbf{H}$  umí odhalit i tři *souvislé* chyby. Od Hammingova kódu ze strany [12] se liší pořadím bitů kódového slova.

## Generující a kontrolní matice

Protože cyklický kód má bázi  $g, x \cdot g, x^2 \cdot g, \dots, x^{k-1} \cdot g$ , kde  $g$  je generující polynom,  $g(x) = g_0 + g_1x + g_2x^2 + \dots + g_{n-k}x^{n-k}$ , je generující matice tvaru

$$\mathbf{G} = \begin{pmatrix} g_0 & g_1 & g_2 & \dots & g_{n-k} & 0 & 0 & \dots & 0 & 0 \\ 0 & g_0 & g_1 & \dots & g_{n-k-1} & g_{n-k} & 0 & \dots & 0 & 0 \\ 0 & 0 & g_0 & \dots & g_{n-k-2} & g_{n-k-1} & g_{n-k} & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & \dots & g_0 & g_1 & \dots & g_{n-k-1} & g_{n-k} \end{pmatrix}$$

Polynom  $h = (x^n - 1)/g$  se nazývá *kontrolní polynom*. Dá se ukázat, že matice s koeficienty kontrolního polynomu  $h_k, h_{k-1}, \dots, h_1, h_0$  umístěnými (v tomto pořadí) opakovaně „podél vedlejší diagonály“, je maticí kontrolní. Ta se v případě cyklických kódů v dekóderu příliš nevyužívá.

## Kodér a dekodér cyklického kódu

**Kódování podle generující matice** není systematické. Kodér z informačních bitů  $\vec{u}$  vytvoří kódové slovo  $\vec{u} \cdot \mathbf{G}$ . Fakticky tedy vytvoří kódové slovo ve tvaru  $u \cdot g$ .

Dekodér spočítá *syndrom* přijatého slova jako zbytek po dělení generujícím polynomem. Je-li nulový, je přijaté slovo kódové. Výsledek dělení obsahuje informační bity.

**Pozorování:** Syndrom nezávisí na kódovaném slovu, ale pouze na chybě:

$$f \cdot g + e = s_1 \text{ mod } g, \quad e = s_2 \text{ mod } g, \quad \text{pak } s_1 = s_2.$$

Důkaz:  $f \cdot g + e = r_1 \cdot g + s_1, e = r_2 \cdot g + s_2$ .  $s_1 - s_2$  je násobek  $g$  se stupněm menším, takže  $s_1 - s_2 = 0$ .

## Systematické kódování

**Kodér** z informačních bitů  $(u_1, u_2, \dots, u_k)$  sestaví polynom:

$$u(x) = u_1x^{n-1} + u_2x^{n-2} + \dots + u_{k-1}x^{n-k+1} + u_kx^{n-k},$$

vypočítá  $z$  jako zbytek po dělení  $u$  polynomem  $g$  a odešle kódové slovo  $u-z$ . Proč je kódové? Je  $u = f \cdot g + z$ . Protože  $f \cdot g$  je násobek  $g$ , musí i  $u-z$  být násobek  $g$ . Navíc součet  $u-z$  nepoškodí posledních  $k$  informačních bitů.

**Dekodér** spočítá syndrom  $s$  jako zbytek po dělení přijatého slova polynomem  $g$ . Je-li  $s = 0$ , je přijaté slovo kódové. Posledních  $k$  bitů obsahuje informaci.

O analýze syndromu si povíme za chvíli.

## Zbytek po dělení polynomu polynomem

se v případě polynomů nad  $\mathbf{Z}_2$  hledá snadno. V příkladu zapisujeme bity v opačném pořadí než dosud, tj.

$$(a_{n-1}, a_{n-2}, \dots, a_1, a_0) \leftrightarrow a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \dots + a_1x + a_0.$$

**Příklad:** Nechť  $g = (1011)$ . Chceme kódovat informaci (1111). Sestavíme polynom  $u = (1111000)$  a dělíme ho polynomem  $g$ :

kodér:	dekodér:
1111000	1111111
1011	1011
0100000	0100111
1011	1011
0001100	0001011
1011	1011
0000111 = $z$ ,	0000000 = $s$ (syndrom)

## Analýza syndromu

Sestavíme tabulku chyb a jejich syndromů:

$\vec{e}_1 \leftrightarrow \vec{s}_1, \vec{e}_2 \leftrightarrow \vec{s}_2, \dots, \vec{e}_m \leftrightarrow \vec{s}_m$ . Tabulku vyplníme (dřív, než začneme kódovat) tak, že pro každou chybu  $e_i$  spočítáme zbytek při dělení polynomem  $g$  a dostaneme  $s_i$ .

Kdybychom měli v paměti uloženu tuto tabulku, pak pro každý syndrom  $\vec{s}_i$  dekodér najde zpětně  $\vec{e}_i$  a přijaté slovo  $\vec{w}$  opraví takto:  $\vec{v} = \vec{w} - \vec{e}_i$ .

Problém: paměťová náročnost + nutnost pro každé přijaté slovo prohledat tabulku.

## Analýza syndromu podle Meggitta

Učinné pozorování na příkladu (7, 4) cyklického kódu. Tabulka  $\vec{e}_i \leftrightarrow \vec{s}_i$ , která obsahuje všechny chyby váhy 1, vypadá takto:

$$\begin{aligned} e_1 = x^0 &\leftrightarrow s_1 = 1 \\ e_2 = x^1 &\leftrightarrow s_2 = x \\ e_3 = x^2 &\leftrightarrow s_3 = x^2 \\ e_4 = x^3 &\leftrightarrow s_4 = x + 1 \\ e_5 = x^4 &\leftrightarrow s_5 = x^2 + x \\ e_6 = x^5 &\leftrightarrow s_6 = x^2 + x + 1 \\ e_7 = x^6 &\leftrightarrow s_7 = x^2 + 1 \quad \dots \text{ syndrom posledního bitu} \end{aligned}$$

Pro syndromy platí:  $s_{i+1} = x \cdot s_i \bmod g$ . Přitom  $s_8 = s_1$ . Je tedy možné „protočit syndromy“ postupnou aplikací operace  $x \cdot s_i \bmod g$ .

V jednom okamžiku se z každého syndromu stane syndrom posledního bitu. Děláme-li současně cyklický posun přijatého slova, dostal se opravovaný bit na poslední pozici. Opravíme ho tam.

## Algoritmus podle Meggitta

Sestavme seznam všech syndromů, které odpovídají všem chybám, které mají na poslední pozici jedničku (seznam všech syndromů posledního bitu). Uložme tento seznam do paměti dekodéru. Seznam zdaleka neobsahuje všechny syndromy.

Nechť délka kódu je  $n$ . Dekodér provede postupně  $n$  cyklických posunů přijatého slova (tím ho dostane nakonec do původního stavu) a současně cyklicky protáčí syndrom podle vzorce  $s_{i+1} = x \cdot s_i \bmod g$ . Kdykoli se syndrom shoduje s některým syndromem posledního bitu (ze seznamu), opraví dekodér poslední bit (cyklicky pounutého) přijatého slova.

Opravuje-li kód jedinou chybu, obsahuje seznam jediný syndrom posledního bitu. Opravuje-li dvě chyby, pak seznam obsahuje  $n$  syndromů. Výpočet probíhá s lineární složitostí (existuje dobře popsaná hw implementace pomocí hradel).

## Korekce souvislých chyb

Existují cyklické kódy, které opravují souvislé chyby délky  $t$ . Dá se ukázat, že pro takové kódy platí: pokud při „protáčení syndromu“ dospějeme k syndromu stupně menšího než  $t$ , pak lze naráz opravit v odpovídajícím (cyklicky posunutém) přijatém slově všechny kontrolní bity přímo podle (protočeného) syndromu.

Inspirace: podívejte se na první řádek tabulky na str. [26].

## Příklady „větších“ cyklických kódů

- Golay code je perfektní kód opravující tři chyby. Je to cyklický (23, 12) kód s generujícím polynomem:

$$1 + x^2 + x^4 + x^5 + x^6 + x^{10} + x^{11}$$

- CRC 32 je metoda počítání kontrolních součtů (syndromů) dat libovolné délky s generujícím polynomem:

$$1 + x + x^2 + x^4 + x^5 + x^7 + x^8 + x^{10} + x^{11} + x^{12} + x^{16} + x^{22} + x^{23} + x^{26} + x^{32}$$

K hlubšímu zkoumání této problematiky můžete použít:

Jiří Adámek: Foundations of Coding, A Wiley-Interscience publication, 1991, ISBN 0-471-62187-0.

Poznámka: Prof. Jiří Adámek byl v letech 1990–1994 vedoucí naší katedry, nyní působí na University of Braunschweig.