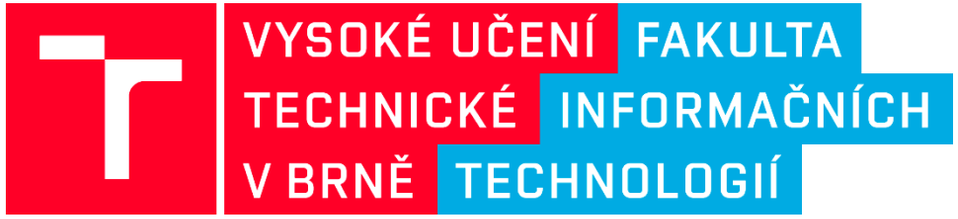
Kódy

Zdeněk Vašíček, Vojtěch Mrázek

2025



Obsah

- Huffmanovo kódování
  - bezeztrátový kód pro odstranění redundance
- Hammingův kód
  - kód pro zabezpečení dat proti chybě
- Kód zbytkových tříd
  - kód pro efektivní realizaci aritmetických operací

Základní pojmy

- zdrojová abeceda
  - množina symbolů, které se mohou vyskytovat ve zprávě (abeceda zpráv)
- kódová abeceda
  - množina symbolů, které se mohou vyskytovat v zakódované zprávě (abeceda kódových slov)
- kódovací předpis
  - návod, jak transformovat slova ze zdrojové do kódové abecedy a naopak (nutné respektovat počet znaků, po kterých je zdrojová zpráva zpracovávána)
- kodové slovo versus nekódové slovo
  - taková kombinace znaků kódové abecedy, která se vyskytuje/nevyskytuje v zakódované zprávě
- Hammingova vzdálenost
  - nejmenší počet znaků kódové abecedy v nichž se dvojice kódových slov liší (počítáno přes všechna kódová slova)
  - užitečná metrika pro zabezpečovací kódy (2 – SED, 3 – SEC, 4 – SEC,DED, 5 – DEC)
- Míry kódů (redundance, teoretická optimální délka kódu, ...)

Příklad kódů a jejich vlastnosti

Příklad:

- zdrojová abeceda {0,1}
- kódová abeceda {L,H}
- kódovací předpis:
  $0 \rightarrow LH$, $1 \rightarrow HL$

- počet znaků po kterých se čte (zpracovává) zpráva:
- kódová slova:
- nekódová slova:
- Hammingova vzdálenost:
  => kód schopný chybu v jednom znaku
- Použití: přenos dat v Ethernetu (Manchester kódování)

Příklad kódů a jejich vlastnosti

Příklad:

- zdrojová abeceda {0,1}
- kódová abeceda {L,H}
- kódovací předpis:
  $0 \rightarrow LH$, $1 \rightarrow HL$

- počet znaků po kterých se čte (zpracovává) zpráva:
- kódová slova:
- nekódová slova:
- Hammingova vzdálenost:
  => kód schopný chybu v jednom znaku
- Použití: přenos dat v Ethernetu (Manchester kódování)

Příklad:

- zdrojová abeceda {0,1}
- kódová abeceda {L,H}
- kódovací předpis:
  $0 \rightarrow LLL$, $1 \rightarrow HHH$

- počet znaků po kterých se čte zpráva:
- kódová slova:
- nekódová slova:
- Hammingova vzdálenost:
  => - kód schopný chybu v jednom znaku
- Použití: zabezpečení přenosu proti chybě (neefektivní – redundance %)

Příklad kódů a jejich vlastnosti

Příklad:

- zdrojová abeceda {0,1}
- kódová abeceda {L,H}
- kódovací předpis:
  $0 \rightarrow$ LH, $1 \rightarrow$ HL

- počet znaků po kterých se čte (zpracovává) zpráva: 1
- kódová slova: LH, HL
- nekódová slova: LL, HH
- Hammingova vzdálenost: 2
  => SED - kód schopný detekovat chybu v jednom znaku
- Použití: přenos dat v Ethernetu (Manchester kódování)

Příklad:

- zdrojová abeceda {0,1}
- kódová abeceda {L,H}
- kódovací předpis:
  $0 \rightarrow$ LLL, $1 \rightarrow$ HHH

- počet znaků po kterých se čte zpráva: 1
- kódová slova: LLL, HHH
- nekódová slova: LLH, LHL, LHH, HLL, HLH, HHL
- Hammingova vzdálenost: 3
  => SEC – kód schopný detekovat a opravit chybu v jednom znaku
- Použití: zabezpečení přenosu proti chybě (neefektivní – redundance 200%)

Zabezpečení pomocí parity

Příklad:

- zdrojová abeceda {0,1}
- kódová abeceda {0,1}
- kódovací předpis:
  $0000 \rightarrow 00000$, $0001 \rightarrow 00101$, $0010 \rightarrow 00110$, $0011 \rightarrow 00011$,
  $0100 \rightarrow 01100$, $0101 \rightarrow 01001$, $0110 \rightarrow 01010$, $0111 \rightarrow 01111$,
  $1000 \rightarrow 10100$, $1001 \rightarrow 10001$, $1010 \rightarrow 10010$, $1011 \rightarrow 10111$,
  $1100 \rightarrow 11000$, $1101 \rightarrow 11101$, $1110 \rightarrow 11110$, $1111 \rightarrow 11011$,

- počet znaků po kterých se zpráva zpracovává:
- varianta kódu: sudá/lichá parita
- kódová slova:
- nekódová slova:
- Hammingova vzdálenost:
- zabezpečovací schopnosti:

Q: Jak se zabezpečí zpráva, která má délku 10 bitů?

Huffmanovo kódování

- nejefektivnější prefixový kód pro odstranění redundance
- pro sestrojení kódovacího předpisu je zapotřebí znát četnosti jednotlivých kódovaných symbolů
- použití:
  - kódování instrukcí,
  - součást mnoha technik pro kompresi obrazu, zvuku a dokumentů (MPG, JPEG, MP3, ZIP, 7z, …)

Huffmanovo kódování - příklad

- kódování po blocích velikosti M=128 B, zdrojová abeceda {A…J}, kódovaná zpráva:

```
AECACCAICCHDCACAHDCAFCCHBGCGECIHACFCHCHBCADCJIHC
HCDHCGBDCCGHCGJCAHHJCHCEEECIHCEBCHHCCCACDDCHGHHA
HHCHCGDHIEDCCCDCHHCHAHHICAHHHEHA
```

- Analýzou zprávy získáme četnost výskytu jednotlivých symbolů zprávy.

symbol	počet výskytů (m_i)	četnost výskytu (f_i)
A	14	0.109 (7/64)
B	4	0.031 (1/32)
C	42	0.328 (21/64)
D	10	0.078 (5/64)
E	8	0.063 (1/16)
F	2	0.016 (1/64)
G	7	0.055 (7/128)
H	32	0.250 (1/4)
I	6	0.047 (3/64)
J	3	0.023 (3/128)

Huffmanovo kódování - příklad

- kódování po blocích velikosti M=128 B, zdrojová abeceda {A…J}, kódovaná zpráva

```
AECACCAICCHDCACAHDCAFCCHBGCGECIHACFCHCHBCADCJIHC
HCDHCGBDCCGHCGJCAHHJCHCEEECIHCEBCHHCCCACDDCHGHHA
HHCHCGDHIEDCCCDCHHCHAHHICAHHHEHA
```

- Analýzou zprávy získáme četnost výskytu jednotlivých symbolů zprávy.

symbol	počet výskytů (m_i)	četnost výskytu (f_i)
A	14	0.109 (7/64)
B	4	0.031 (1/32)
C	42	0.328 (21/64)
D	10	0.078 (5/64)
E	8	0.063 (1/16)
F	2	0.016 (1/64)
G	7	0.055 (7/128)
H	32	0.250 (1/4)
I	6	0.047 (3/64)
J	3	0.023 (3/128)

Konstrukce kódových slov

1. Jednotlivé symboly prohlásíme za uzly (listy) stromu. Postupně po dvojicích spojujeme uzly s nejmenším ohodnocením, vytváříme uzly nové dokud nezkonstruujeme celý Huffmanův strom.
2. **Systematicky** ohodnotíme hrany stromu (např. hrana vedoucí do uzlu s menším ohodnocením 0, jinak 1).
3. Cesta z vrcholu až k listu tvoří kódové slovo symbolu odpovídajícího danému listu.

Huffmanovo kódování - příklad

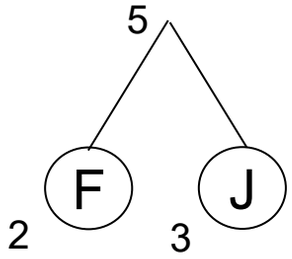
Q: Sestrojte Huffmanův kód na základě znalosti počtu výskytů jednotlivých symbolů

symbol	počet výskytů
C	42
H	32
A	14
D	10
E	8
G	7
I	6
B	4
J	3
F	2

Huffmanovo kódování - příklad

Q: Sestrojte Huffmanův kód na základě znalosti počtu výskytů jednotlivých symbolů

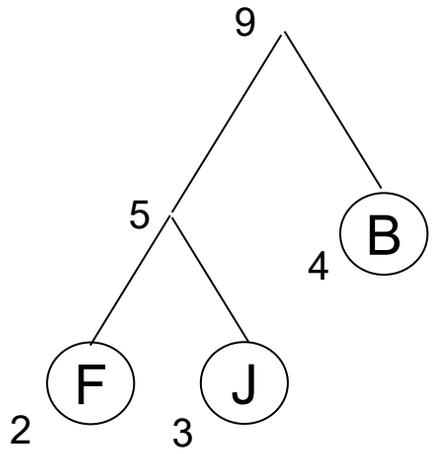
symbol	počet výskytů
C	42
H	32
A	14
D	10
E	8
G	7
I	6
B	4
~~J~~	~~3~~
~~F~~	~~2~~



Huffmanovo kódování - příklad

Q: Sestrojte Huffmanův kód na základě znalosti počtu výskytů jednotlivých symbolů

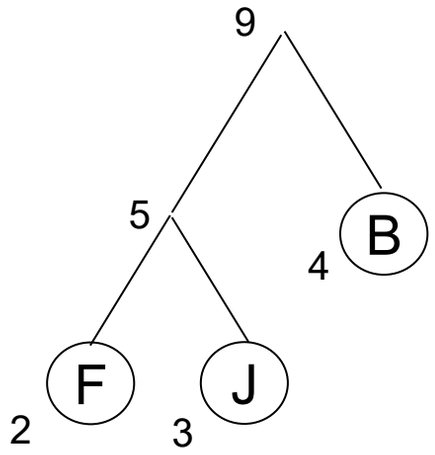
symbol	počet výskytů
C	42
H	32
A	14
D	10
E	8
G	7
I	6
~~B~~	~~4~~
~~J~~	~~3~~
~~F~~	~~2~~

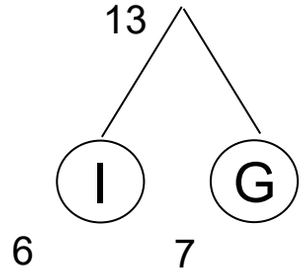


Huffmanovo kódování - příklad

Q: Sestrojte Huffmanův kód na základě znalosti počtu výskytů jednotlivých symbolů

symbol	počet výskytů
C	42
H	32
A	14
D	10
E	8
~~G~~	~~7~~
~~I~~	~~6~~
~~B~~	~~4~~
~~J~~	~~3~~
~~F~~	~~2~~

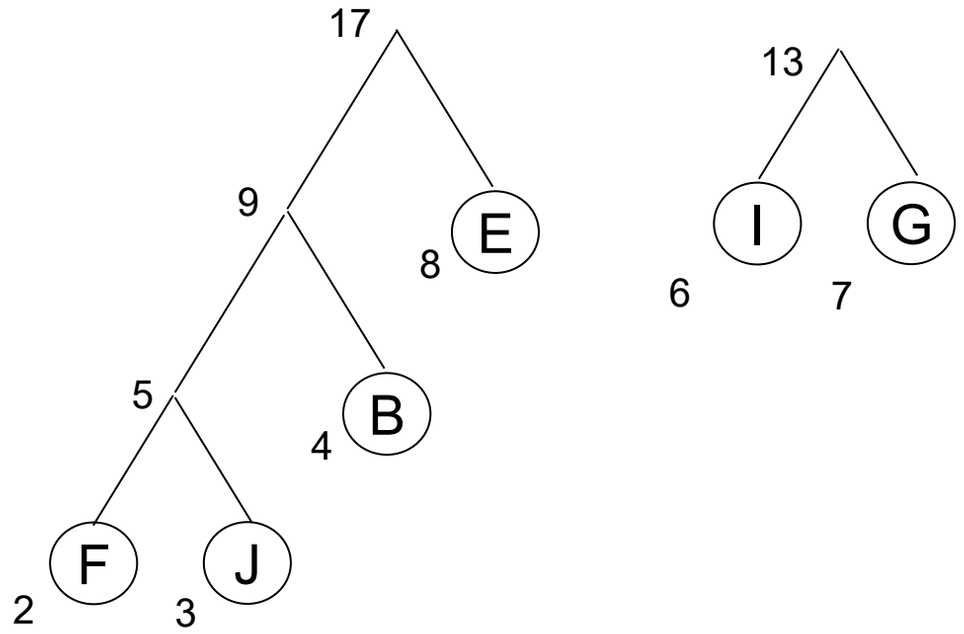




Huffmanovo kódování - příklad

Q: Sestrojte Huffmanův kód na základě znalosti počtu výskytů jednotlivých symbolů

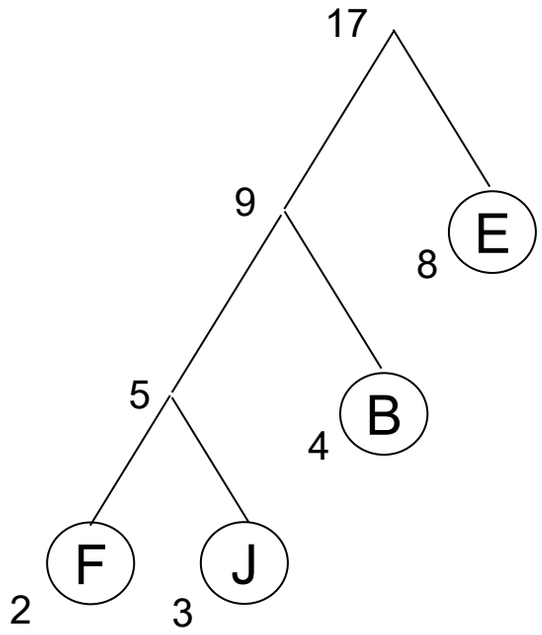
symbol	počet výskytů
C	42
H	32
A	14
D	10
~~E~~	~~8~~
~~G~~	~~7~~
~~I~~	~~6~~
~~B~~	~~4~~
~~J~~	~~3~~
~~F~~	~~2~~

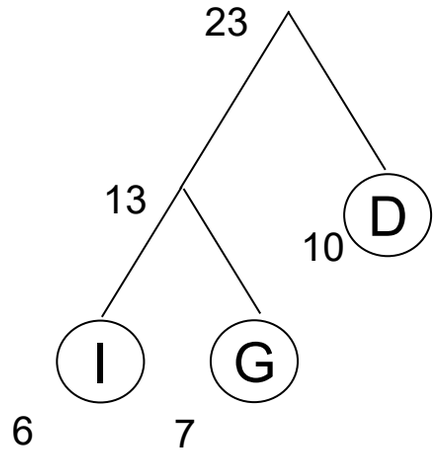


Huffmanovo kódování - příklad

Q: Sestrojte Huffmanův kód na základě znalosti počtu výskytů jednotlivých symbolů

symbol	počet výskytů
C	42
H	32
A	14
~~D~~	~~10~~
~~E~~	~~8~~
~~G~~	~~7~~
~~I~~	~~6~~
~~B~~	~~4~~
~~J~~	~~3~~
~~F~~	~~2~~

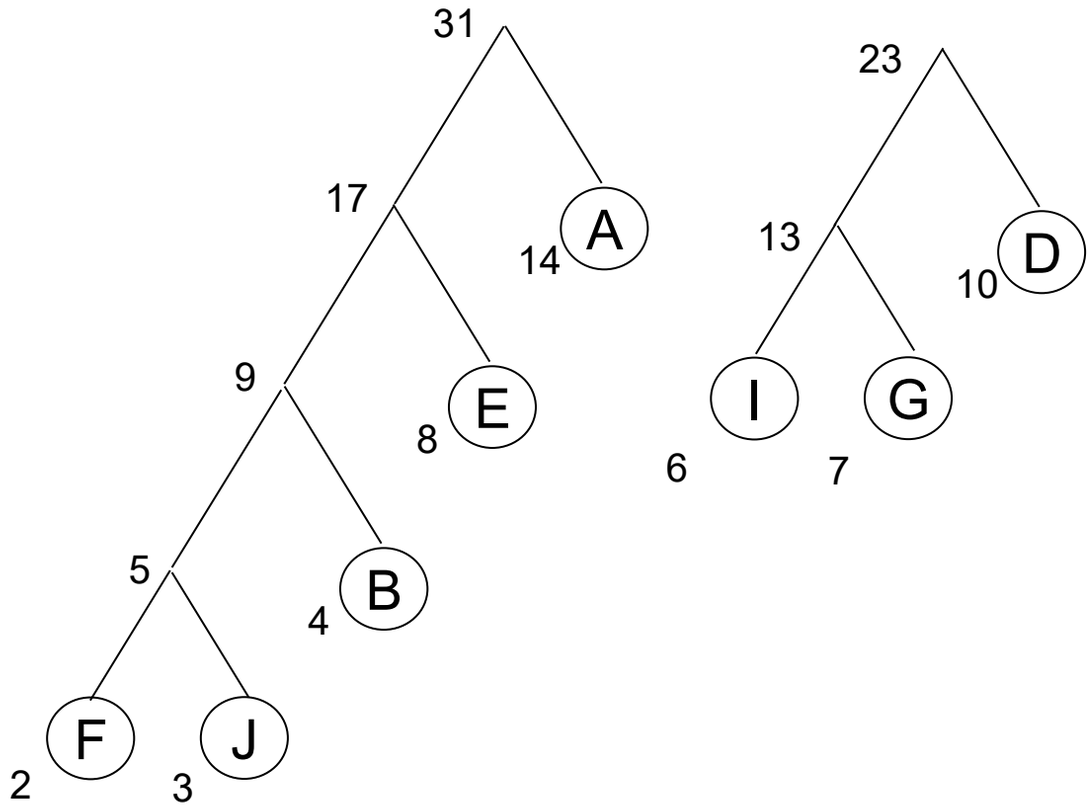




Huffmanovo kódování - příklad

Q: Sestrojte Huffmanův kód na základě znalosti počtu výskytů jednotlivých symbolů

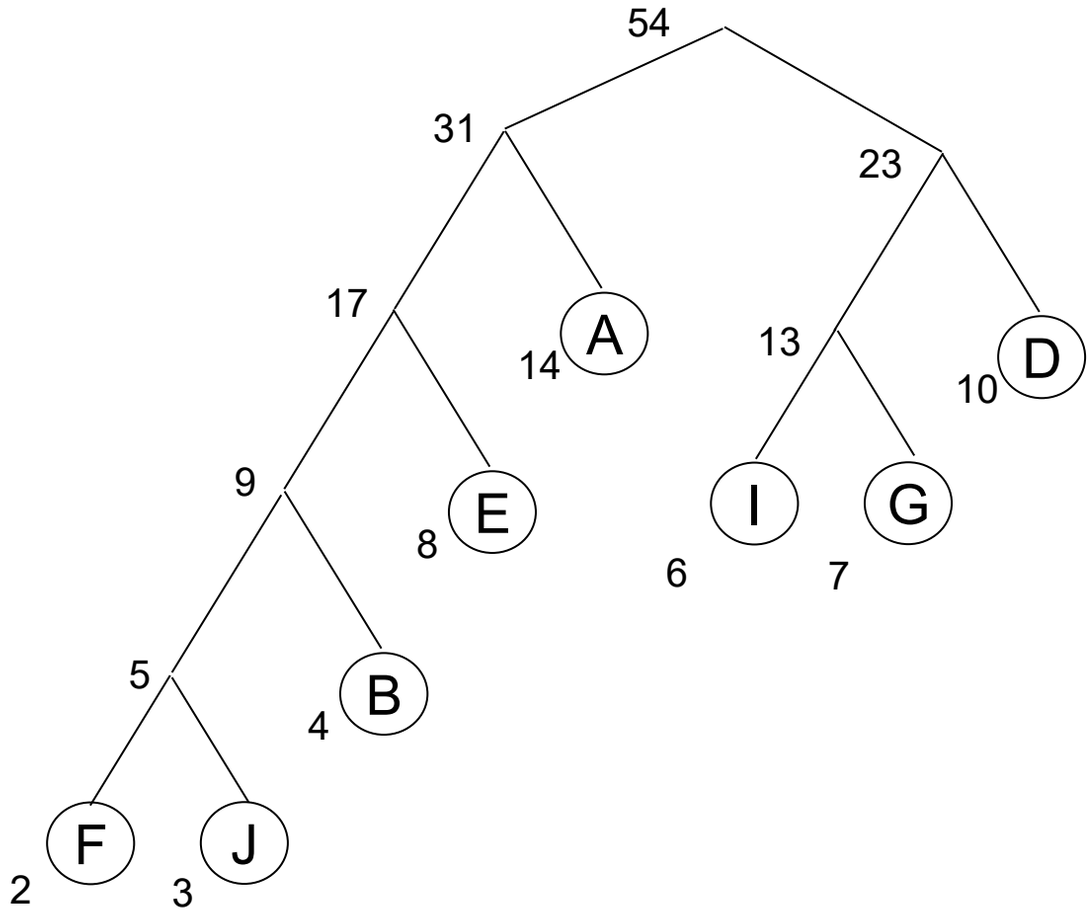
symbol	počet výskytů
C	42
H	32
~~A~~	~~14~~
~~D~~	~~10~~
~~E~~	~~8~~
~~G~~	~~7~~
~~I~~	~~6~~
~~B~~	~~4~~
~~J~~	~~3~~
~~F~~	~~2~~



Huffmanovo kódování - příklad

Q: Sestrojte Huffmanův kód na základě znalosti počtu výskytů jednotlivých symbolů

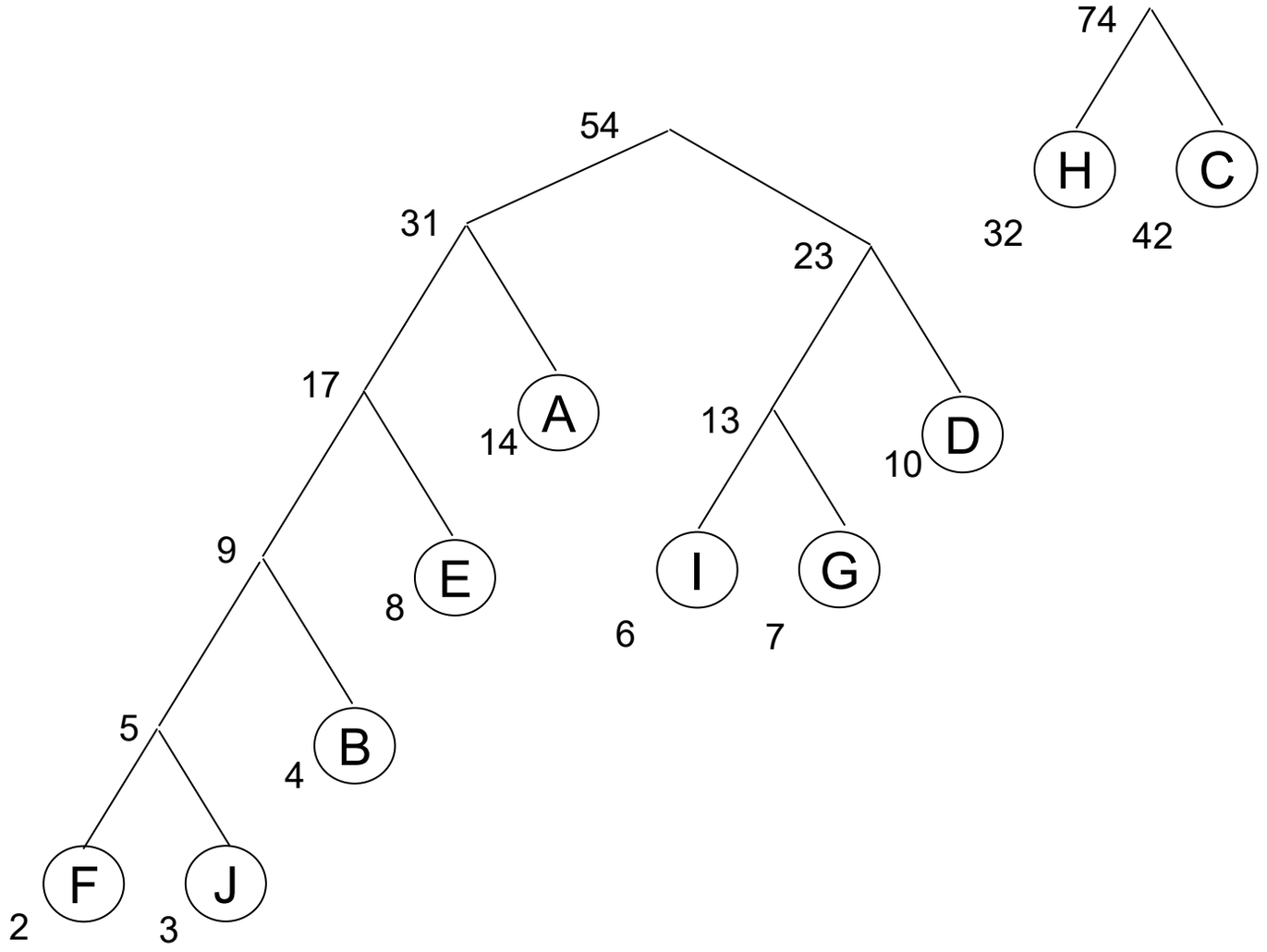
symbol	počet výskytů
C	42
H	32
A	14
D	10
E	8
G	7
I	6
B	4
J	3
F	2



Huffmanovo kódování - příklad

Q: Sestrojte Huffmanův kód na základě znalosti počtu výskytů jednotlivých symbolů

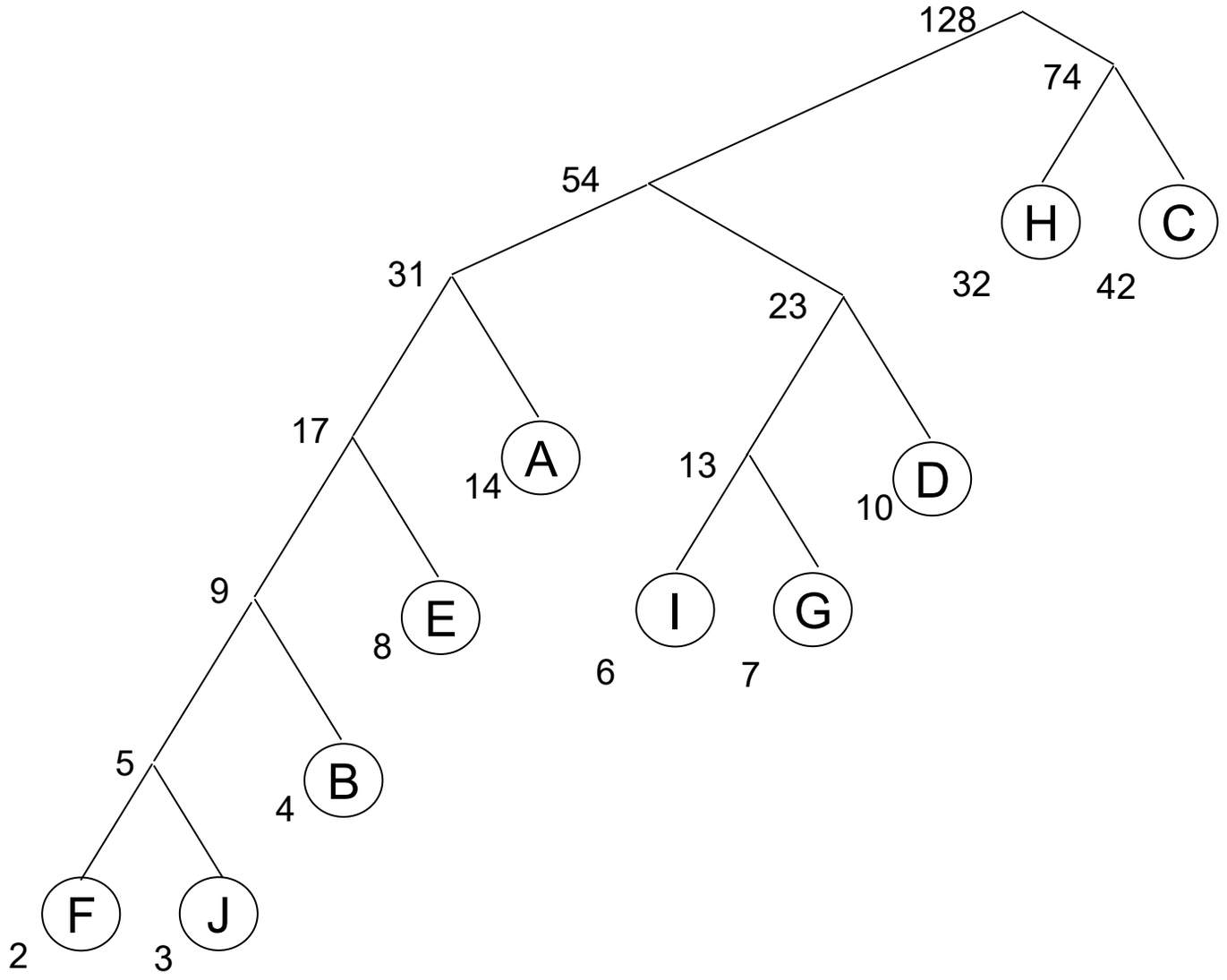
symbol	počet výskytů
C	42
H	32
A	14
D	10
E	8
G	7
I	6
B	4
J	3
F	2



Huffmanovo kódování - příklad

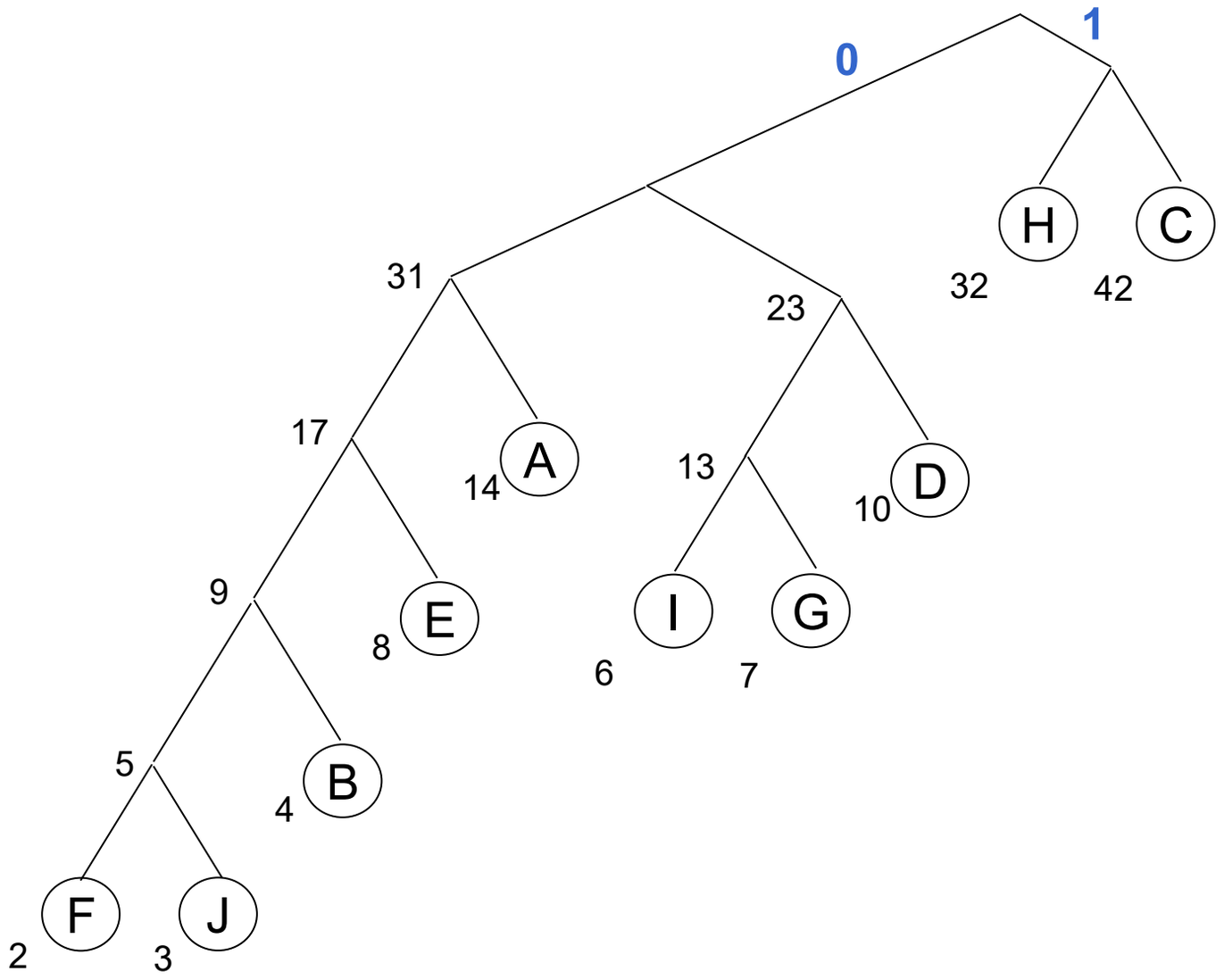
Q: Sestrojte Huffmanův kód na základě znalosti počtu výskytů jednotlivých symbolů

symbol	počet výskytů
C	42
H	32
A	14
D	10
E	8
G	7
I	6
B	4
J	3
F	2



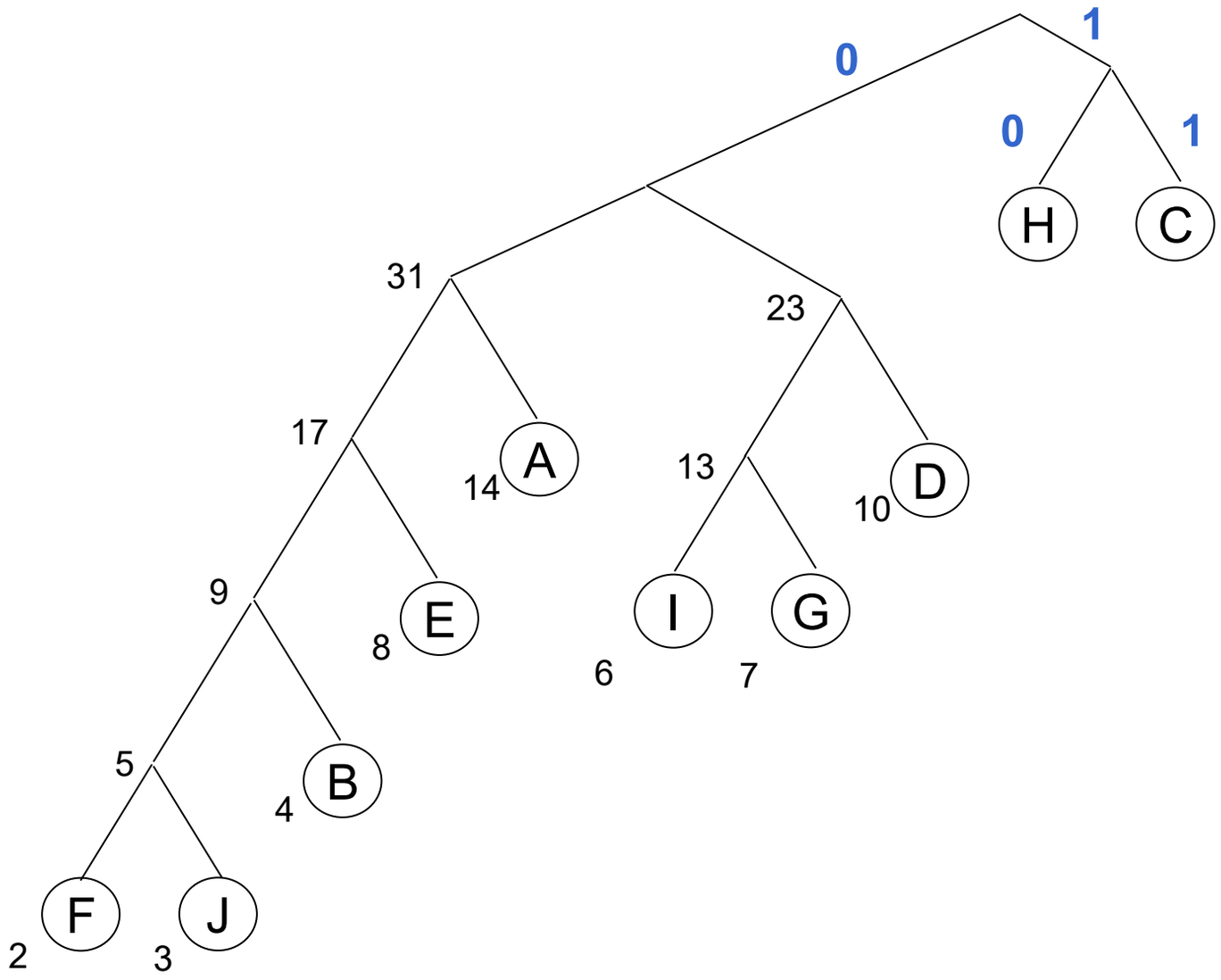
Huffmanovo kódování - příklad

Q: Sestrojte Huffmanův kód na základě znalosti počtu výskytů jednotlivých symbolů



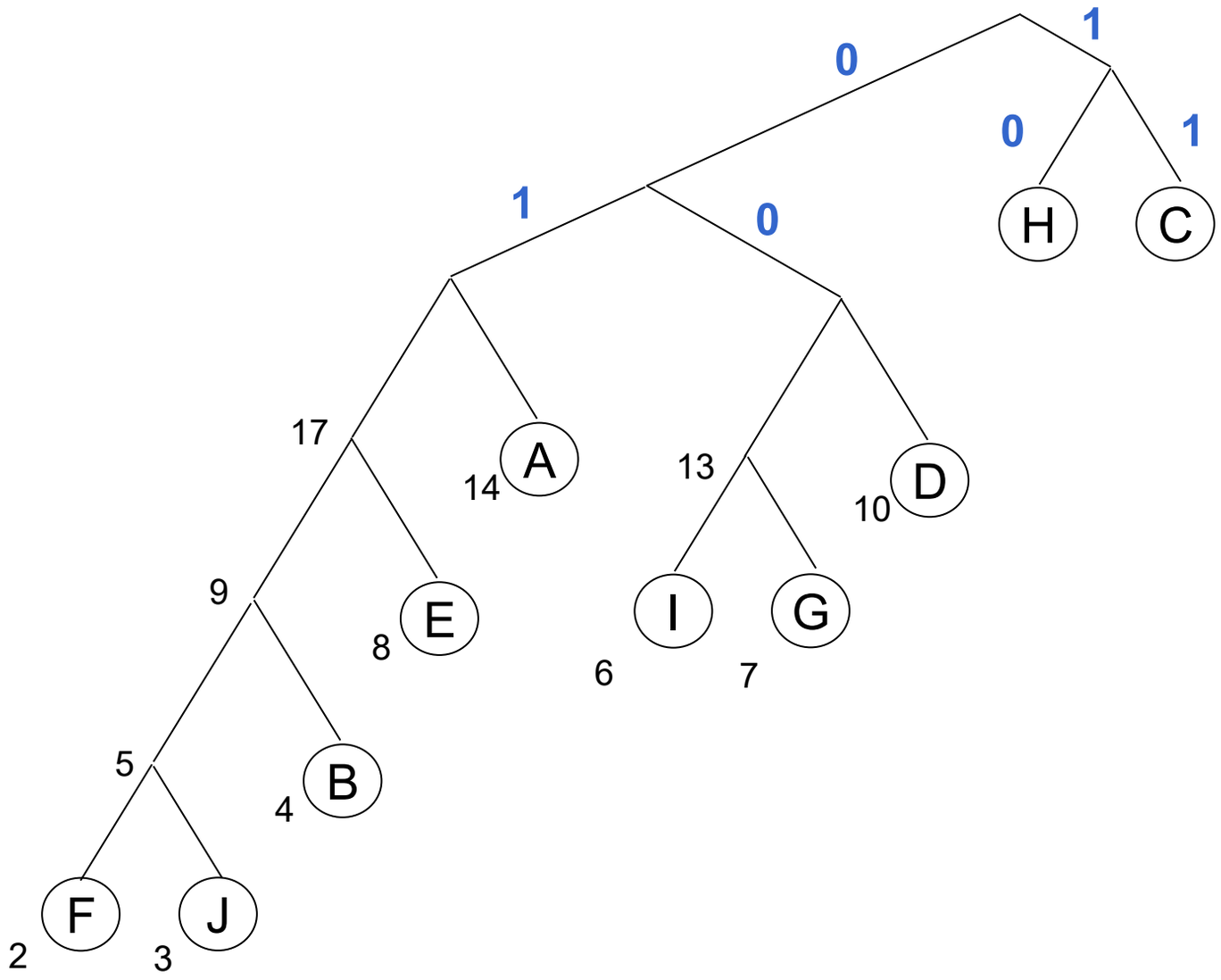
Huffmanovo kódování - příklad

Q: Sestrojte Huffmanův kód na základě znalosti počtu výskytů jednotlivých symbolů



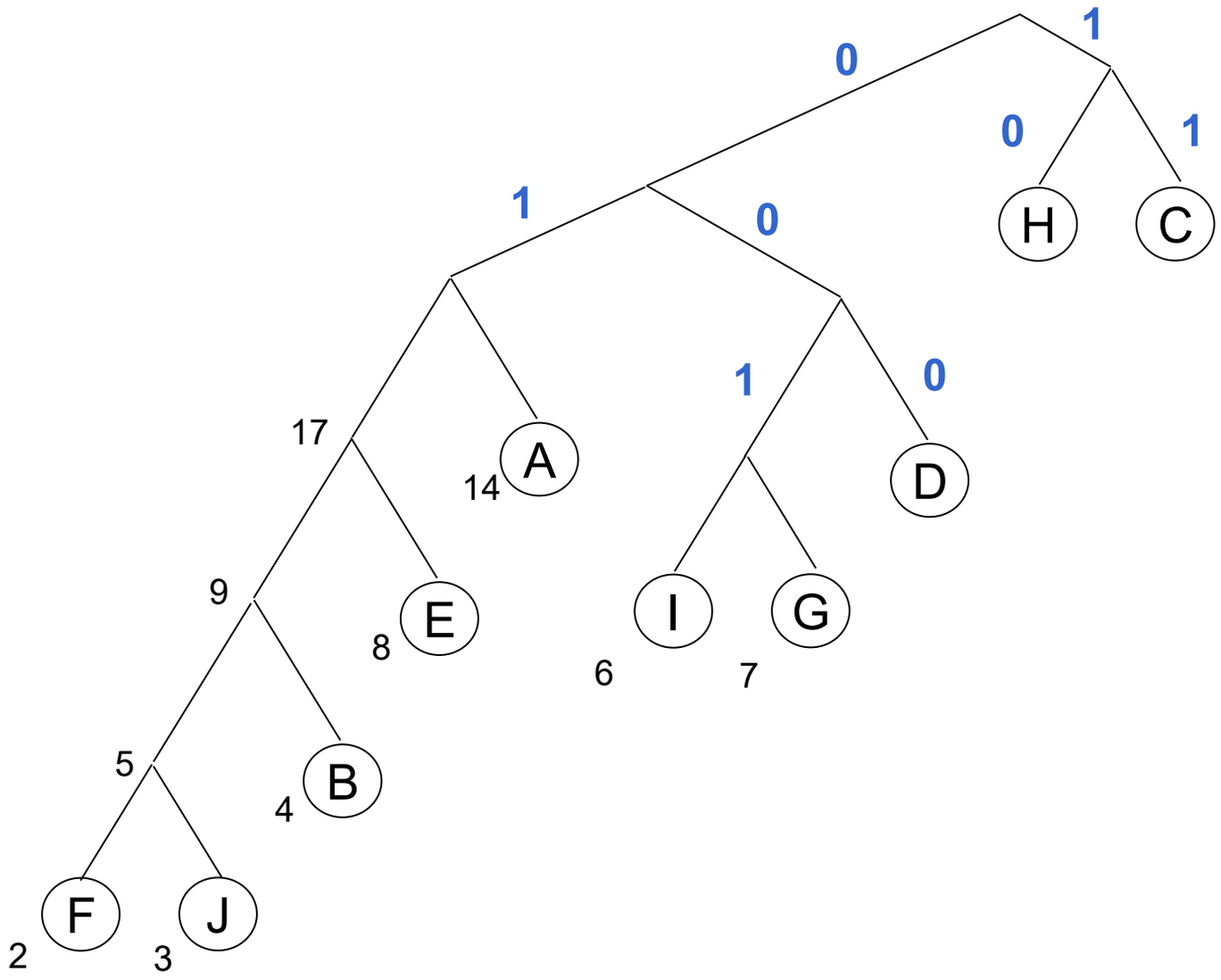
Huffmanovo kódování - příklad

Q: Sestrojte Huffmanův kód na základě znalosti počtu výskytů jednotlivých symbolů



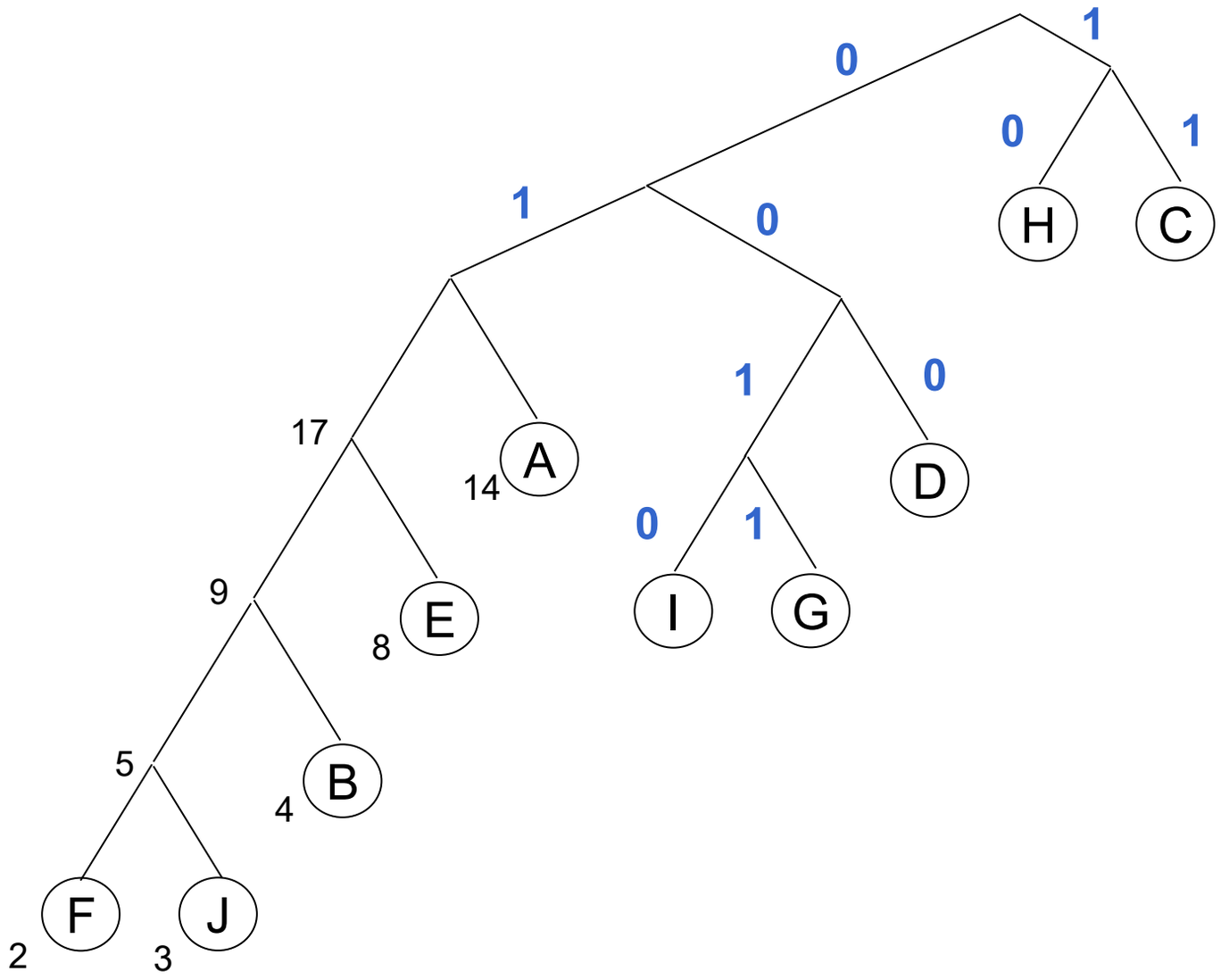
Huffmanovo kódování - příklad

Q: Sestrojte Huffmanův kód na základě znalosti počtu výskytů jednotlivých symbolů



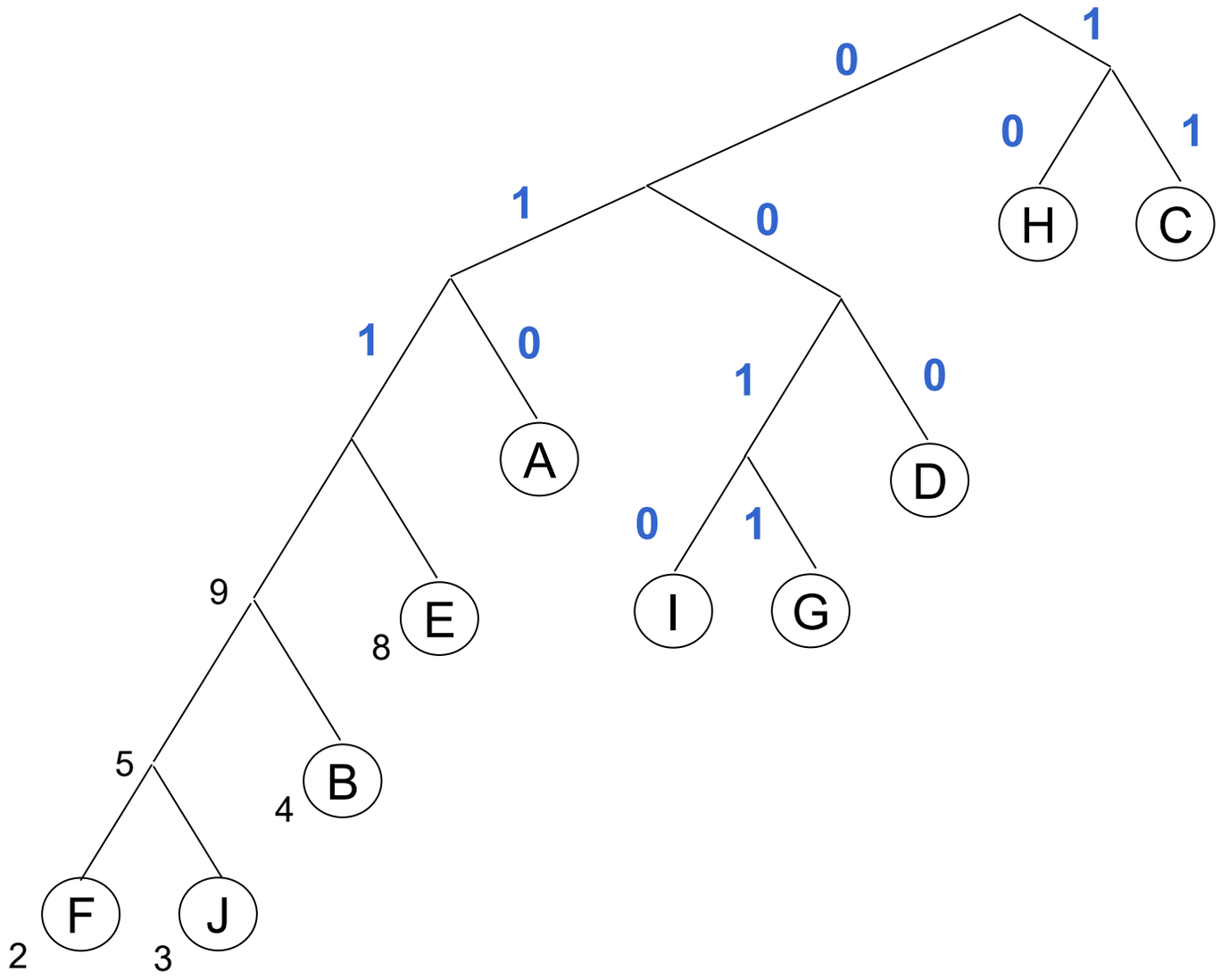
Huffmanovo kódování - příklad

Q: Sestrojte Huffmanův kód na základě znalosti počtu výskytů jednotlivých symbolů



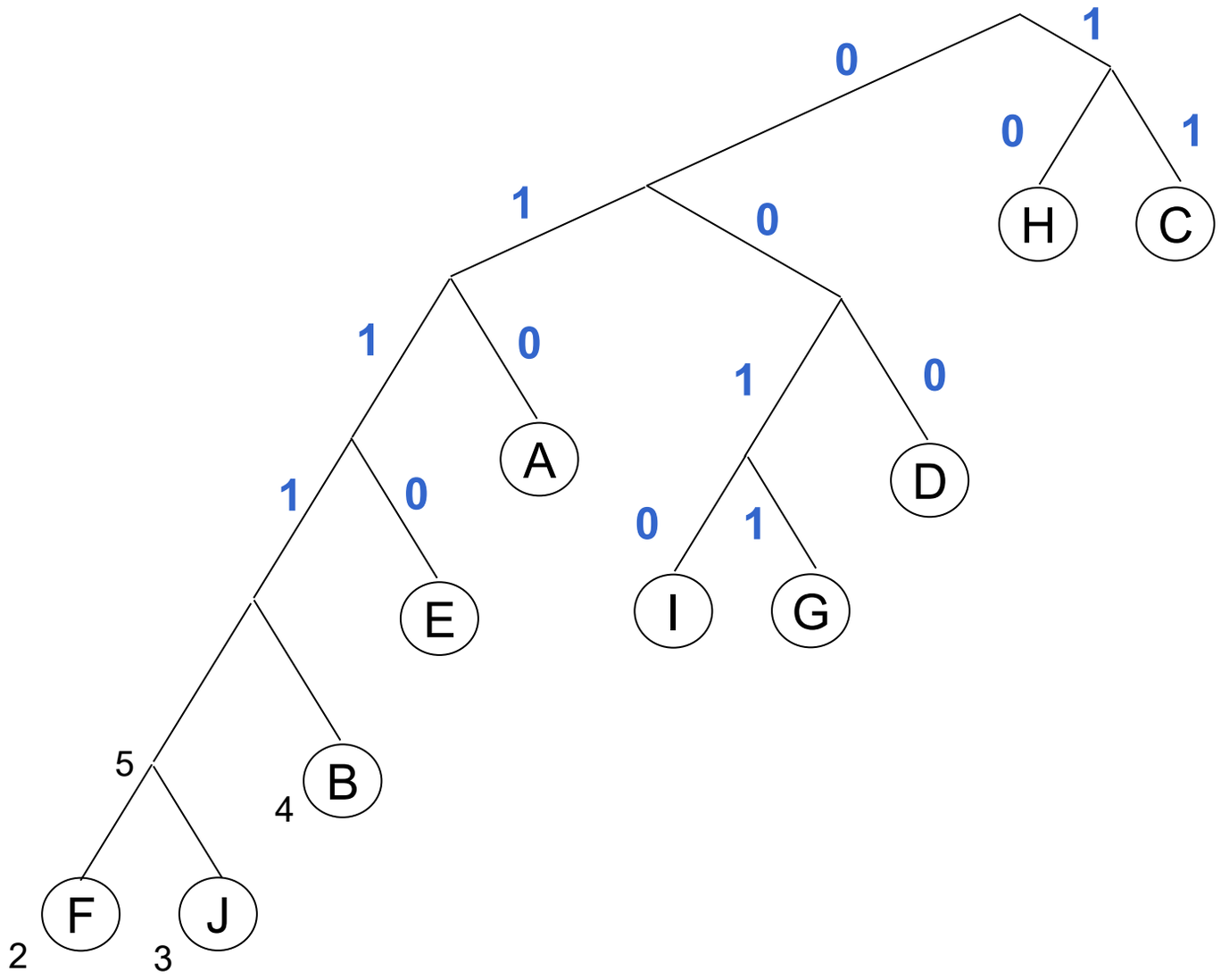
Huffmanovo kódování - příklad

Q: Sestrojte Huffmanův kód na základě znalosti počtu výskytů jednotlivých symbolů



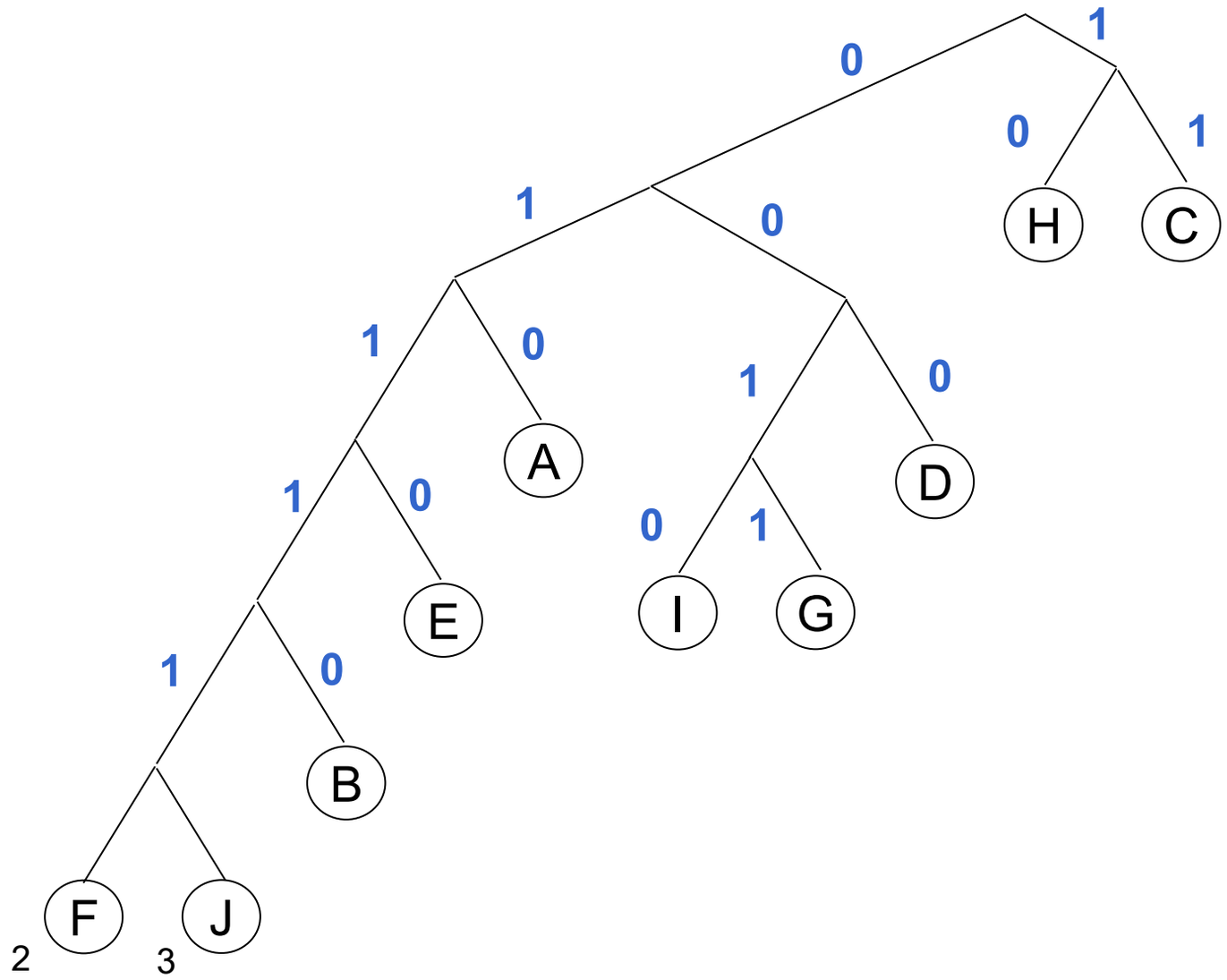
Huffmanovo kódování - příklad

Q: Sestrojte Huffmanův kód na základě znalosti počtu výskytů jednotlivých symbolů



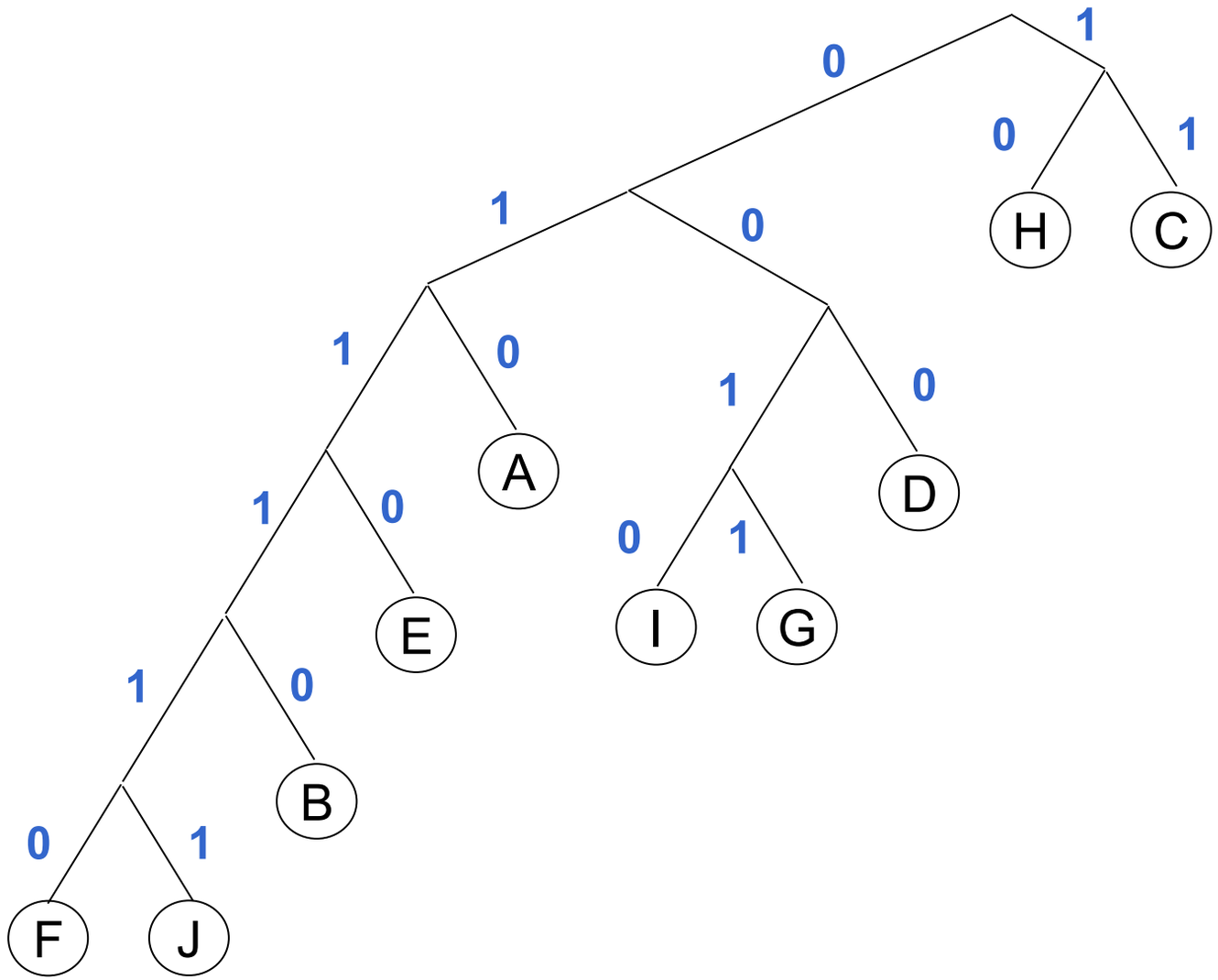
Huffmanovo kódování - příklad

Q: Sestrojte Huffmanův kód na základě znalosti počtu výskytů jednotlivých symbolů



Huffmanovo kódování - příklad

Q: Sestrojte Huffmanův kód na základě znalosti počtu výskytů jednotlivých symbolů



Huffmanovo kódování - příklad

Tabulka symbolů a odpovídajících kódových slov (častěji se vyskytující symboly mají přiřazeno kratší kódové slovo)

symbol	počet výskytů (m_i)	kódové slovo	L_i
A	14	010	3
B	4	01110	5
C	42	11	2
D	10	000	3
E	8	0110	4
F	2	011110	6
G	7	0011	4
H	32	10	2
I	6	0010	4
J	3	011111	6

$L_{dyn} = 2.76$ $L_{opt} = 2.72$

$$L_{dyn} = \sum_{i=1}^{N} L_i f_i = \frac{1}{M} \sum_{i=1}^{N} L_i m_i \qquad f_i = \frac{m_i}{M}$$

$$L_{opt} = -\sum_{i=1}^{N} f_i \log_2 f_i = \log_2 M - \frac{1}{M} \sum_{i=1}^{N} m_i \log_2 m_i$$

$$R = \frac{L_{dyn} - L_{opt}}{L_{dyn}}$$

M=128 (délka původní zprávy)
N=10 (počet kódových slov)

Q: Jak vypadá prvních 16 bitů zakódované zprávy?

Q: Dekódujte zprávu 01110010011110

Q: Jaká je střední dyn. délka kódu, teor. optimální délka kódu, redundance kódu?

Hammingův kód (HK)

- nejefektivnější kód (minimální možná redundance) pro zabezpečení dat proti výskytu chyby v jednom symbolu s možností chybu opravit (SEC kód)
- kódové slovo obsahuje kontrolní bity, jejichž umístění je dáno pozicí bitu ve slově
  - kontrolní bity jsou tvořeny pomocí funkce XOR z informačních bitů (vhodně zvolené pokrytí informačních bitů sadou parit – sudá parita)
  - je-li index pozice bitu ve slově mocnina 2, jedná se o bit kontrolní (C); v opačném případě se jedná o bit informační (I)
- Hammingův kód je separovatelný a má Hammingovu vzdálenost 3

Paritní bit

- Paritní bit je typ kódu, který přidává redundantní bit k datovému slovu obsahující informaci o počtu jedničkových bitů ve slově.
- Paritní bit je určen k jednoduché detekci chyby ve slově (speciální případ 1-bitového CRC, polynom x+1).
- Rozlišujeme dva typy: lichá a sudá parita

8bitová data	kódové slovo s paritním bitem	
	sudá parita	lichá parita
00000000	**0**00000000	**1**00000000
10010001	**1**10010001	**0**10010001
11001001	**0**11001001	**1**11001001
11111111	**0**11111111	**1**11111111

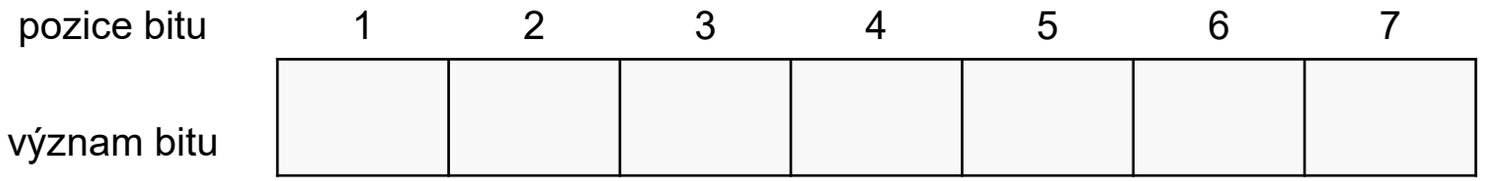
- Výpočet: funkce XOR

Hammingův kód (HK)

- před použitím je nutné definovat počet bitů kódového slova
  - HK(n,k) nebo Hamming(n,k) je běžně používané označení, kde
    - *n* je délka kódového slova (počet bitů),
    - *k* je počet informačních bitů a
    - platí, že $k = n - m$, kde *m* je počet kontrolních bitů a $n = 2^m - 1$
  - příklad používaných kódů
    - HK(7,4) – 3 kontrolní bity (právě 3 bity jsou zapotřebí k zakódování 8 kombinací)
    - HK(15,11) – 4 kontrolní bity (právě 4 bity jsou zapotřebí k zakódování 16 kombinací)
    - HK(127,120) – 7 kontrolních bitů
  - čím větší *n*, tím je redundance kódu menší (poměr *k/n* příznivější)

Hammingův kód – sestavení kódu HK(7,4)

- je-li index pozice bitu mocnina 2, jedná se o bit kontrolní, jinak datový (informační)



Hammingův kód – sestavení kódu HK(7,4)

- je-li index pozice bitu mocnina 2, jedná se o bit kontrolní, jinak datový (informační)

pozice bitu	1	2	3	4	5	6	7
význam bitu	C1	C2	I3	C4	I5	I6	I7

Hammingův kód – sestavení kódu HK(7,4)

- je-li index pozice bitu mocnina 2, jedná se o bit kontrolní, jinak datový (informační)

pozice bitu	1	2	3	4	5	6	7
význam bitu	C1	C2	I3	C4	I5	I6	I7

- pokrytí kontrolními (paritními) bity

	C1	C2	I3	C4	I5	I6	I7
C1							
C2							
C4							

Hammingův kód – sestavení kódu HK(7,4)

- je-li index pozice bitu mocnina 2, jedná se o bit kontrolní, jinak datový (informační)

pozice bitu	1	2	3	4	5	6	7
význam bitu	C1	C2	I3	C4	I5	I6	I7

- pokrytí kontrolními (paritními) bity

	C1	C2	I3	C4	I5	I6	I7
C1	X		X		X		X
C2		X	X			X	X
C4				X	X	X	X

Hammingův kód – sestavení kódu HK(7,4)

- je-li index pozice bitu mocnina 2, jedná se o bit kontrolní, jinak datový (informační)

pozice bitu	1	2	3	4	5	6	7
význam bitu	C1	C2	I3	C4	I5	I6	I7

- pokrytí kontrolními (paritními) bity

	C1	C2	I3	C4	I5	I6	I7
C1	X		X		X		X
C2		X	X			X	X
C4				X	X	X	X

- Generující rovnice (pro výpočet kontrolních bitů):

$C_1 = I_3 \text{ xor } I_5 \text{ xor } I_7$ $C_2 = I_3 \text{ xor } I_6 \text{ xor } I_7$ $C_4 = I_5 \text{ xor } I_6 \text{ xor } I_7$

Hammingův kód - příklad

Zakódujte v HK(7,4) slovo 1101 (MSB vlevo)

C1 = I3 xor I5 xor I7 C2 = I3 xor I6 xor I7 C4 = I5 xor I6 xor I7

Pozn: operace xor v tomto případě odpovídá sčítání modulo 2

Hammingův kód - příklad

Zakódujte v HK(7,4) slovo 1101 (MSB vlevo)

pozice bitu	7	6	5	4	3	2	1
význam bitu	I7	I6	I5	C4	I3	C2	C1

C1 = I3 xor I5 xor I7 C2 = I3 xor I6 xor I7 C4 = I5 xor I6 xor I7

Pozn: operace xor v tomto případě odpovídá sčítání modulo 2

Hammingův kód - příklad

Zakódujte v HK(7,4) slovo 1101 (MSB vlevo)

pozice bitu	7	6	5	4	3	2	1
význam bitu	I7	I6	I5	C4	I3	C2	C1

pozice bitu	7	6	5	4	3	2	1
hodnota bitu	1	1	0	C4	1	C2	C1

C1 = I3 xor I5 xor I7 C2 = I3 xor I6 xor I7 C4 = I5 xor I6 xor I7

Pozn: operace xor v tomto případě odpovídá sčítání modulo 2

Hammingův kód - příklad

Zakódujte v HK(7,4) slovo 1101 (MSB vlevo)

pozice bitu	7	6	5	4	3	2	1
význam bitu	I7	I6	I5	C4	I3	C2	C1

pozice bitu	7	6	5	4	3	2	1
hodnota bitu	1	1	0	C4	1	C2	C1

pozice bitu	7	6	5	4	3	2	1
hodnota bitu	1	1	0	0	1	1	0

C1 = I3 xor I5 xor I7 C2 = I3 xor I6 xor I7 C4 = I5 xor I6 xor I7

Pozn: operace xor v tomto případě odpovídá sčítání modulo 2

Hammingův kód – detekce a oprava chyby

- Chybu v jednom bitu lze detekovat na základě hodnoty tzv. chybového syndromu S, který určuje pozici chyby ve slově.

$$S_1 = C'_1 \text{ xor } I'_3 \text{ xor } I'_5 \text{ xor } I'_7$$
$$S_2 = C'_2 \text{ xor } I'_3 \text{ xor } I'_6 \text{ xor } I'_7$$
$$S_4 = C'_4 \text{ xor } I'_5 \text{ xor } I'_6 \text{ xor } I'_7$$

- Jednotlivé bity syndromu testují, zda-li odpovídající paritní bity jsou správné. Pokud nikoliv, tak vzhledem k rozložení paritních bitů získáme pozici chyby v přijatém slově.

Hammingův kód - příklad

Dekódujte v HK(7,4) přijaté slovo 1001100 a 1001000 (MSB vlevo)

$S_1 = C'_1 \text{ xor } I'_3 \text{ xor } I'_5 \text{ xor } I'_7$ $S_2 = C'_2 \text{ xor } I'_3 \text{ xor } I'_6 \text{ xor } I'_7$ $S_4 = C'_4 \text{ xor } I'_5 \text{ xor } I'_6 \text{ xor } I'_7$

Hammingův kód - příklad

Dekódujte v HK(7,4) přijaté slovo 1001100 a 1001000 (MSB vlevo)

pozice bitu	7	6	5	4	3	2	1
hodnota bitu	1	0	0	1	1	0	0

$S_1 = C'_1 \text{ xor } I'_3 \text{ xor } I'_5 \text{ xor } I'_7$ $S_2 = C'_2 \text{ xor } I'_3 \text{ xor } I'_6 \text{ xor } I'_7$ $S_4 = C'_4 \text{ xor } I'_5 \text{ xor } I'_6 \text{ xor } I'_7$

Hammingův kód - příklad

Dekódujte v HK(7,4) přijaté slovo 1001100 a 1001000 (MSB vlevo)

pozice bitu	7	6	5	4	3	2	1
hodnota bitu	1	0	0	1	1	0	0

$S_1 = 0$ $S_2 = 0$ $S_4 = 0$ $S = 000$ přenos proběhl bez chyby nebo nastal větší počet chyb

$S_1 = C'_1 \text{ xor } I'_3 \text{ xor } I'_5 \text{ xor } I'_7$ $S_2 = C'_2 \text{ xor } I'_3 \text{ xor } I'_6 \text{ xor } I'_7$ $S_4 = C'_4 \text{ xor } I'_5 \text{ xor } I'_6 \text{ xor } I'_7$

Hammingův kód - příklad

Dekódujte v HK(7,4) přijaté slovo 1001100 a 1001000 (MSB vlevo)

pozice bitu	7	6	5	4	3	2	1
hodnota bitu	1	0	0	1	1	0	0

$S_1 = 0$ $S_2 = 0$ $S_4 = 0$ $S = 000$ přenos proběhl bez chyby nebo nastal větší počet chyb

pozice bitu	7	6	5	4	3	2	1
hodnota bitu	1	0	0	1	0	0	0

$S_1 = C'_1 \text{ xor } I'_3 \text{ xor } I'_5 \text{ xor } I'_7$ $S_2 = C'_2 \text{ xor } I'_3 \text{ xor } I'_6 \text{ xor } I'_7$ $S_4 = C'_4 \text{ xor } I'_5 \text{ xor } I'_6 \text{ xor } I'_7$

Hammingův kód - příklad

Dekódujte v HK(7,4) přijaté slovo 1001100 a 1001000 (MSB vlevo)

pozice bitu	7	6	5	4	3	2	1
hodnota bitu	1	0	0	1	1	0	0

$S_1 = 0$ $S_2 = 0$ $S_4 = 0$ $S = 000$ přenos proběhl bez chyby nebo nastal větší počet chyb

pozice bitu	7	6	5	4	3	2	1
hodnota bitu	1	0	0	1	0	0	0

$S_1 = 1$ $S_2 = 1$ $S_4 = 0$ $S = 011$ bit na pozici 3 má chybou hodnotu, opravíme na opačnou

$S_1 = C'_1 \text{ xor } I'_3 \text{ xor } I'_5 \text{ xor } I'_7$ $S_2 = C'_2 \text{ xor } I'_3 \text{ xor } I'_6 \text{ xor } I'_7$ $S_4 = C'_4 \text{ xor } I'_5 \text{ xor } I'_6 \text{ xor } I'_7$

Hammingův kód - příklad

Dekódujte v HK(7,4) přijaté slovo 1001100 a 1001000 (MSB vlevo)

pozice bitu	7	6	5	4	3	2	1
hodnota bitu	1	0	0	1	1	0	0

$S_1 = 0$ $S_2 = 0$ $S_4 = 0$ $S = 000$ přenos proběhl bez chyby nebo nastal větší počet chyb

pozice bitu	7	6	5	4	3	2	1
hodnota bitu	1	0	0	1	1	0	0

$S_1 = 1$ $S_2 = 1$ $S_4 = 0$ $S = 011$ bit na pozici 3 má chybou hodnotu, opravíme na opačnou

$S_1 = C'_1 \text{ xor } I'_3 \text{ xor } I'_5 \text{ xor } I'_7$ $S_2 = C'_2 \text{ xor } I'_3 \text{ xor } I'_6 \text{ xor } I'_7$ $S_4 = C'_4 \text{ xor } I'_5 \text{ xor } I'_6 \text{ xor } I'_7$

Rozšířený Hammingův kód

- Přidáním paritního bitu přes všechny bity HK(n,k) lze rozšířit Hammingův kód na kód typu SEC-DED, který je schopen detekovat dvojchybu (d=4), označovaný jako HK(n+1,k)

- Příklad pro HK(8,4):
  - přidáme kontrolní paritní bit C0:

    C0 = C1 xor C2 xor I3 xor C4 xor I5 xor I6 xor I7
  - obdobně rozšíříme syndrom chyby:

    S0 = C'0 xor C'1 xor C'2 xor I'3 xor C'4 xor I'5 xor I'6 xor I'7
  - rozšíříme syndrom o příznak R

    R = S1 or S2 or S4
  - Chyby klasifikujeme následovně

R	S_0	Význam
0	0	Bez chyby
0	1	Neopravitelná chyba (porucha hlídače, vícenásobná chyba)
1	0	Neopravitelná 2-chyba, 4-chyba, atd.
1	1	Opravitelná 1-chyba

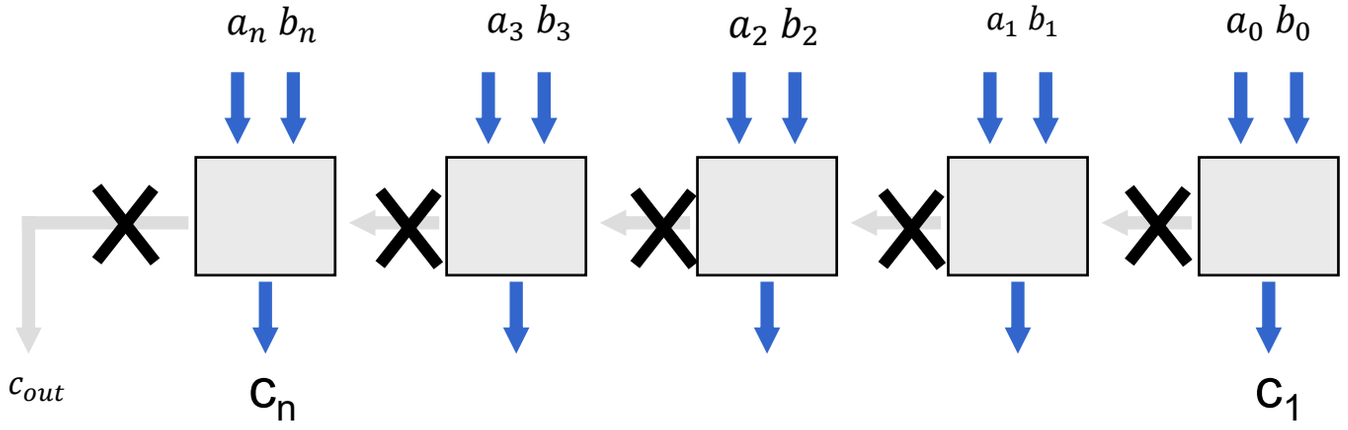
Hammingův kód

- Q: Dekódujte v HK(7,4) přijaté slovo 1001010 obsahující dvojchybu (původní hodnota byla 1001100).
- Q: Dekódujte v HK(15,11) slovo 11011110101101110
- Q: Co se stane nastane-li v HK(7,4) dvojchyba a co v HK(15,11)?
- Q: Kolik kontrolních bitů je použito HK má-li kódové slovo délku 31 bitů?
- Q: Kolik kontrolních bitů je zapotřebí na vytvoření HK má-li informační slovo délku 31 bitů?
- Q: Zakódujte slovo 1010 v rozšířeném HK(8,4).
- Q: Dekódujte slovo 10101010 v rozšířeném HK(8,4).

Kód zbytkových tříd (KZT, RNS)

- Kód zbytkových tříd je systém reprezentace dat, pomocí kterého je možné realizovat sčítání, odčítání a násobení bez přenosů, což má významný dopad na výkonnost zejména v případě mnohabitových operandů (např. v kryptografii RSA algoritmus používá 2048bitů)

- Princip:



$$c_i = (a_i \text{ op } b_i) \bmod m_i$$

- Známé problémy: obecné dělení, test na přetečení, porovnání, test na znaménko jsou obtížně realizovatelné operace

Volba modulů RNS

- Nejprve je nutné vhodně zvolit jednotlivé moduly dle požadovaného dynamického rozsahu
- Dynamický rozsah je roven nejmenšímu společnému násobku modulů.
  (rozklad na prvočísla, vezmene u každého prvočísla nejvyšší mocninu)
- Příklady:

Moduly	Moduly rozložené na prvočísla	Dynamický rozsah	
(3,6)	(3, 2*3)	2*3 = 6	totožný rozsah
(6)	(2*3)	2*3 = 6	
(3,7,9)	(3, 7, 3^2)	3^2*7 = 63	totožný rozsah
(7, 9)	(7, 3^2)	3^2*7 = 63	

- Pro efektivní využití reprezentace je vhodné mít největší společný dělitel všech modulů (po dvojicích) roven jedné (tzn.navzájem nesoudělná čísla, ideálně prvočísla).

Převod do RNS

- Máme-li definované vhodné moduly $\{m_1,\ldots,m_N\}$, pak číslo X v binárním tvaru lze v RNS reprezentovat jako $\{x_1,\ldots,x_N\}$, kde $x_i = X \bmod m_i$

- Příklady:

$$X = 7 \rightarrow \langle 1|1|2\rangle_{RNS(2|3|5)}$$

$$\begin{aligned} 2 \cdot 3 + 1 &= 7 \quad 7 \bmod 2 = 1 \\ 3 \cdot 2 + 1 &= 7 \quad 7 \bmod 3 = 1 \\ 5 \cdot 1 + 2 &= 7 \quad 7 \bmod 5 = 2 \end{aligned}$$

$$X = 70 \rightarrow \langle 1|0|0\rangle_{RNS(3|5|7)}$$

$$\begin{aligned} 3 \cdot 23 + 1 &= 70 \quad 70 \bmod 3 = 1 \\ 5 \cdot 14 + 0 &= 70 \quad 70 \bmod 5 = 0 \\ 7 \cdot 10 + 0 &= 70 \quad 70 \bmod 7 = 0 \end{aligned}$$

Převod z RNS zpět

- Zpětný převod je výpočetně mnohem náročnější. V praxi se využívá buď výpočet založený na algoritmu CRT (Chinese Remainder Theorem) nebo MRS (Mixed Radix System).

- Máme-li číslo $\{x_1,\ldots,x_N\}$, moduly $\{m_1,\ldots,m_N\}$ a $M=m_1m_2\ldots m_N$, pak X získáme jako

$$X = \sum_{i=1}^{N} x_i M_i N_i \bmod M = (x_1 M_1 N_1 + \cdots + x_N M_N N_N) \bmod M$$

  kde M_i odpovídá součinu hodnot všech modulů kromě i-tého a
  $N_i = M_i^{-1} \bmod m_i$ je multiplikativní inverze M_i v Z_{mi}

- Multiplikativní inverze x na tělese Z_{mi} je takové číslo x^{-1}, pro které platí, že $x \cdot x^{-1} \equiv 1$

Příklady:

V případě, že $M_i = 15$ a $m_i = 2$ musí být splněno, že $15 * N_i \bmod 2 = 1$, což splňuje $N_i = 1$

V případě, že $M_i = 35$ a $m_i = 3$ musí být splněno, že $35 * N_i \bmod 3 = 1$, což splňuje $N_i = 2$

Převod z RNS zpět

- Zpětný převod je výpočetně mnohem náročnější. V praxi se využívá buď výpočet založený na algoritmu CRT (Chinese Remainder Theorem) nebo MRS (Mixed Radix System).

- Máme-li číslo $\{x_1,\ldots,x_N\}$, moduly $\{m_1,\ldots,m_N\}$ a $M=m_1m_2\ldots m_N$, pak X získáme jako

$$X = \sum_{i=1}^{N} x_i M_i N_i \bmod M = (x_1 M_1 N_1 + \cdots + x_N M_N N_N) \bmod M$$

  kde M_i odpovídá součinu hodnot všech modulů kromě i-tého a
  $N_i = M_i^{-1} \bmod m_i$ je multiplikativní inverze M_i v Z_{mi}

- Příklady:

$$(1|1|2)_{RNS(2|3|5)} \rightarrow X = (1 \cdot (3 \cdot 5) \cdot 1 + 1 \cdot (2 \cdot 5) \cdot 1 + 2 \cdot (2 \cdot 3) \cdot 1) \bmod(2 \cdot 3 \cdot 5) = 7$$

$$(1|0|0)_{RNS(3|5|7)} \rightarrow X = (1 \cdot (5 \cdot 7) \cdot 2 + 0 \cdot (3 \cdot 7) \cdot 1 + 0 \cdot (3 \cdot 5) \cdot 1) \bmod(3 \cdot 5 \cdot 7) = 70$$

Zakódování čísel a aritmetické operace
sčítání

No	m_1 2	m_2 3	m_3 5
0	0	0	0
1	1	1	1
2	0	2	2
3	1	0	3
4	0	1	4
5	1	2	0
6	0	0	1
7	1	1	2
8	0	2	3
9	1	0	4
10	0	1	0
11	1	2	1
12	0	0	2
15	1	0	0
.....			
29	1	2	4
30	0	0	0

Opakuje se
30 = 2*3*5

$c_i = (a_i + b_i) \bmod m_i$ bez přenosu!

```
 1  0  3    (3)
+0  1  4    (4)
-------
 1  1  2
```

$(1|1|2)_{RNS(2|3|5)} \rightarrow X = (1 \cdot 15 + 1 \cdot 10 + 2 \cdot 6) \bmod 30 = 7$

```
 1  0  4     (9)
+1  2  1    (11)
-------
 0  2  0
```

$(0|2|0)_{RNS(2|3|5)} \rightarrow X = (0 \cdot 15 + 2 \cdot 10 + 0 \cdot 6) \bmod 30 = 20$

Zakódování čísel a aritmetické operace
odčítání

No	m_1 2	m_2 3	m_3 5
0	0	0	0
1	1	1	1
2	0	2	2
3	1	0	3
4	0	1	4
5	1	2	0
6	0	0	1
7	1	1	2
8	0	2	3
9	1	0	4
10	0	1	0
11	1	2	1
12	0	0	2
.....			
27	1	0	2
28	0	1	3
29	1	2	4
30	0	0	0

Opakuje se

30 = 2*3*5

$$c_i = (a_i - b_i) \bmod m_i$$

```
 0  0  1  (6)
-1  2  0  (5)
-------------
 1  1  1
```

$$(1|1|1)_{RNS(2|3|5)} \rightarrow X = (1 \cdot 15 + 1 \cdot 10 + 1 \cdot 6) \bmod 30 = 1$$

```
 1  0  3  (3)
-0  0  1  (6)
-------------
 1  0  2
```

V této reprezentaci může být výsledkem pouze kladné číslo! (tj 30 – 3)

$$(1|0|2)_{RNS(2|3|5)} \rightarrow X = (1 \cdot 15 + 0 \cdot 10 + 2 \cdot 6) \bmod 30 = 27$$

Zakódování čísel a aritmetické operace
násobení

No	m_1 2	m_2 3	m_3 5
0	0	0	0
1	1	1	1
2	0	2	2
3	1	0	3
4	0	1	4
5	1	2	0
6	0	0	1
7	1	1	2
8	0	2	3
9	1	0	4
10	0	1	0
11	1	2	1
12	0	0	2
13	1	1	3
14	0	2	4
.....			
29	1	2	4
30	0	0	0

Opakuje se
30 = 2*3*5

$c_i = (a_i * b_i) \bmod m_i$

```
 0  1  4  (4)
*1  2  0  (5)
-----------
 0  2  0
```

$(0|2|0)_{RNS(2|3|5)} \rightarrow X = (0 \cdot 15 + 2 \cdot 10 + 0 \cdot 6) \bmod 30 = 20$

```
 1  0  3   (3)
*1  0  0  (15)
-----------
 1  0  0
```

Výsledkem operace je
(3*15) mod 30!

$(1|0|0)_{RNS(2|3|5)} \rightarrow X = (1 \cdot 15 + 0 \cdot 10 + 0 \cdot 6) \bmod 30 = 15$

Kód zbytkových tříd

- Q: Jaký je dynamický rozsah systému RNS(7|15)?
- Q: Jaký je dynamický rozsah systému RNS(3|6)?
- Q: Převeďte číslo 120 do systému RNS(11|15).
- Q: Převeďte číslo $(1|2|3)_{RNS(3|5|7)}$ na číslo v desítkové soustavě.
- Q: Sečtěte čísla 5 a 8 v RNS(3|6) a výsledek převeďte na číslo v desítkové soustavě.
- Q: Z množiny {3,5,6,7,12,18} zvolte tři moduly tak, aby výsledný RNS systém měl největší možný dynamický rozsah.