Fakulta přírodovědně humanitní a pedagogická, Technická univerzita v Liberci

Matematika pro informatiky II

Doc. RNDr. Miroslav Koucký, CSc.

Liberec, 2016

Obsah

Předmluva

Hlavním cílem předkládaného textu je seznámit čtenáře se základy teorie šifrování, s myšlenkami vybraných kompresních metod a se základy detekčních/opravných kódů. Studium těchto skript vyžaduje znalosti vybraných partií matematiky, které čtenář nalezne ve skriptech Matematika pro informatiky I.

Při nakládání s daty se obvykle setkáváme se třemi zásadními okruhy problémů

- Množství dat → kompresní metody (bezeztrátová komprese, ztrátová komprese)
- Spolehlivost dat (ochrana dat před šumem) → teorie kódování
- Bezpečnost dat (ochrana před neautorizovaným přístupem) → kryptologie (kryptografie, kryptoanalýza; steganografie)

1. Úvod do šifrování

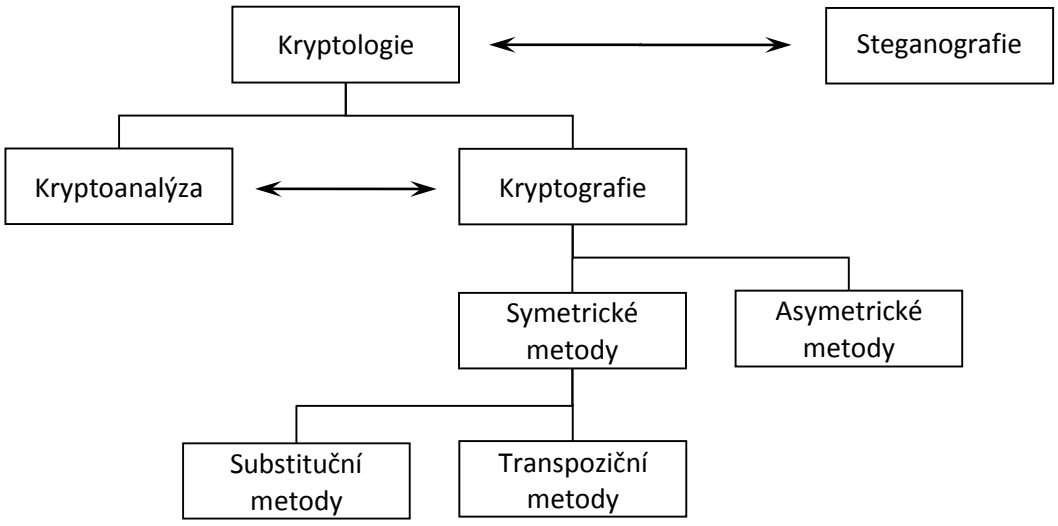
Tato kapitola je stručným úvodem do problematiky šifrování (kryptologie) a seznámí čtenáře se základními pojmy a vybranými šifrovacími metodami. Stručně a zjednodušeně řečeno, smyslem šifrování je ochrana dat před neautorizovaným přístupem.

Kryptografie (kryptos = skrytý, graphein = psát)
Vědecká disciplína, která se zabývá metodami ochrany dat před neautorizovaným přístupem, resp. nakládáním s daty. Je přirozené, že snaha o ochranu dat před neautorizovaným přístupem vede k „protireakci“, tj. vyvolává snahu o prolomení kryptografické ochrany.

Kryptoanalýza
Vědecká disciplína, která se zabývá metodami prolomení kryptografické ochrany. Kryptoanalytické metody jsou v případě klasických substitučních šifrovacích metod obvykle založeny na tzv. frekvenční analýze, která odhaduje identitu znaků (resp. slov) na základě porovnání frekvence jejich výskytu v daném jazyce a v zašifrovaném textu.

Kryptologie
Označení pro vědeckou disciplínu, která zahrnuje jak kryptografii, tak i kryptoanalýzu.

Steganografie (steganos = schovaný, graphein = psát)
Ochránit data před neautorizovaným přístupem lze v zásadě dvěma způsoby – učinit data „nesrozumitelnými“ (kryptografická ochrana) nebo „utajit“ jejich samotnou existenci (steganografie technické a lingvistická).



1. 1. Základní pojmy

Otevřená abeceda

Otevřenou abecedou rozumíme konečnou množinu A, jejíž prvky tvoří znaky používané k zápisu nezašifrovaných zpráv. Jde např. o českou abecedu doplněnou o cifry a další speciální symboly. V těchto skriptech se pro jednoduchost omezíme na znaky anglické abecedy. V celé řadě metod budeme znaky otevřené abecedy nahrazovat jejich pořadím, přičemž použijeme Z_{26}, tj. soustavu nejmenších nezáporných zbytků modulo 26, viz následující tabulka.

a	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*k*	*l*	*m*
0	1	2	3	4	5	6	7	8	9	10	11	12

n	*o*	*p*	*q*	*r*	*s*	*t*	*u*	*v*	*w*	*x*	*y*	*z*
13	14	15	16	17	18	19	20	21	22	23	24	25

Otevřený text

Otevřeným textem rozumíme zprávu určenou k zašifrování, tj. libovolné slovo nad otevřenou abecedou. Jde tedy konečný řetězec $\boldsymbol{m} = m_1 \dots m_n$, kde $m_i \in A$ (n je délka). Otevřený text zapisujeme obvykle malými písmeny.

Prostor otevřených textů

Množinu všech otevřených textů budeme značit M a nazývat prostorem otevřených textů.
Zřejmě $M \subseteq A^*$, kde A^* označuje množinu všech konečných slov nad abecedou A.

Šifrová abeceda

Šifrovou abecedou rozumíme konečnou množinu B znaků používaných k zápisu zašifrovaných zpráv. V případě $B = \{0,1\}$ mluvíme o binárním šifrování.

Zašifrovaný text (šifrový text)

Konečný řetězec $\boldsymbol{c} = c_1 \dots c_n$ znaků šifrové abecedy, který vzniknul zašifrováním některého otevřeného textu $\boldsymbol{m} \in M$. Konkrétní zašifrovaný text budeme zapisovat obvykle velkými znaky.

Prostor šifrových textů

Množinu všech šifrových textů (vzniklých zašifrováním otevřených textů z prostoru M) budeme značit C a nazývat prostorem šifrových textů.

Klíč, prostor klíčů

Klíčem rozumíme uspořádanou dvojici $k = (e, d)$, kde e je šifrovací klíč - parametr šifrovací metody a d je dešifrovací klíč - parametr dešifrovací metody.
Množina všech klíčů tvoří tzv. prostor klíčů, značíme K. Jedním ze základních požadavků je, aby prostor klíčů byl dostatečně obsáhlý a prakticky znemožňoval „uhádnout“ klíč metodou hrubé síly, tj. systematickým prohledáním prostoru klíčů.

Šifrování

Proces transformace otevřeného textu do zašifrovaného textu.

Zjednodušeně řečeno, lze šifrování chápat jako exaktně definovaný proces převedení otevřeného textu do „nesrozumitelné“ podoby zašifrovaného textu.

Šifrovací transformace/funkce
Šifrovací transformací (funkcí) rozumíme vzájemně jednoznačné zobrazení $E_e: M \to C$ definované pro všechny (šifrovací) klíče z prostoru klíčů K.
Vzájemná jednoznačnost zobrazení E_e je nutnou podmínkou pro možnost zpětného dešifrování.

Dešifrování
Dešifrování je inverzní proces k šifrování, tedy jde o proces převedení zašifrovaného textu do podoby otevřeného textu.

Dešifrovací transformace/funkce
Dešifrovací transformací (funkcí) rozumíme zobrazení $D_d: C \to M$, které je inverzní k zobrazení $E_e: M \to C$, kde $(e, d) \in K$.

Šifrovací systém
Uspořádaná trojice $(\mathcal{E}, \mathcal{D}, K)$, kde
$K = \{(e, d)\}$ je prostor klíčů,
$\mathcal{E} = \{E_e | (e, d) \in K\}$ je množina šifrovacích transformací,
$\mathcal{D} = \{D_d | (e, d) \in K\}$ je množina dešifrovacích transformací,
tvoří šifrovací systém, jestliže

$$\forall k = (e, d) \in K \ \ \forall \boldsymbol{m} \in M \qquad\qquad D_d\big(E_e(\boldsymbol{m})\big) = \boldsymbol{m}$$

Interpretace - každý klíč (e, d) jednoznačně definuje dvojici transformací E_e a D_d (šifrovací a jí příslušnou dešifrovací), které jsou navzájem inverzní.

Kerchoffův princip
Bezpečnost šifrovacího systému nesmí záviset na utajení (de)šifrovacího algoritmu, ale pouze na utajení klíče.

Symetrické (klasické) šifrovací metody
Šifrovací metody, kde dešifrovací klíč je výpočetně snadné odvodit ze šifrovacího klíče.

Asymetrické šifrovací metody (s veřejným klíčem)
Šifrovací metody, kde dešifrovací klíč je výpočetně složité odvodit ze šifrovacího klíče.

Transpoziční metody
Šifrovací metody, ve kterých znaky otevřeného textu mění svou pozici, ale nemění svou identitu.

Substituční metody
Šifrovací metody, ve kterých znaky otevřeného textu mění svou identitu, ale nemění svou pozici.
Zjednodušeně řečeno, šifrování využívá tzv. substituční schémata, která přiřazují znakům otevřené abecedy znaky šifrové abecedy (resp. slova nad šifrovou abecedou).

Monoalfabetické šifry
Šifrovací metody využívající pouze jedno substituční schéma.
(Každý znak otevřené abecedy je zašifrován vždy na jeden pevně daný znak šifrové abecedy (resp. slovo nad šifrovou abecedou).

Homofonní šifry
Šifrovací metody, kde znaky šifrového textu mají teoreticky stejnou frekvenci výskytu.

Polyalfabetické šifry
Šifrovací metody využívající více substitučních schémat, která systematicky (tj. dle exaktně definovaných pravidel) střídají.

1.2. Symetrické šifry, transpozice a substituce

Jednoduchá transpozice
Šifrovací klíč: $\pi \in S_d$, kde $d \in N - \{0,1\}$.
Nejprve je otevřený text rozdělen na bloky d po sobě jdoucích znaků, tj. $m = m^{(1)} \dots m^{(k)}$, kde $m^{(i)} = m_1^{(i)} \dots m_d^{(i)}$ je i-tý blok. Následně každý blok $m^{(i)}$ zašifrujeme pomocí transformace:

$$E_\pi\left(m_1^{(i)} \dots m_d^{(i)}\right) = m_{\pi(1)}^{(i)} \dots m_{\pi(d)}^{(i)}, i = 1, \dots k.$$

Dešifrovací klíč: $\pi^{-1} \in S_d$, kde π^{-1} označuje inverzní permutaci k π.
Nejprve zašifrovaný text rozdělíme na bloky d po sobě jdoucích znaků, tj. $c = c^{(1)} \dots c^{(k)}$, kde $c^{(i)} = c_1^{(i)} \dots c_d^{(i)}$ je i-tý blok. Následně každý blok $c^{(i)}$ dešifrujeme pomocí transformace:

$$D_{\pi^{-1}}\left(c_1^{(i)} \dots c_d^{(i)}\right) = c_{\pi^{-1}(1)}^{(i)} \dots c_{\pi^{-1}(d)}^{(i)}, i = 1, \dots k.$$

Poznámky
- Transpoziční šifra je bloková šifra délky d, tj. šifra, která nejprve rozdělí otevřený text na bloky d po sobě jdoucích znaků. Každý blok pak zašifruje jako celek.
- Pokud délka otevřeného textu není násobkem čísla d, doplníme text libovolnými znaky na délku rovnou prvnímu násobku čísla d většímu než n.

Příklad
Uvažujte jednoduchou transpozici s klíčem $\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 1 & 5 & 2 & 4 \end{pmatrix}$.

a) Zašifrujte text „koloseum“.

otevřený text:	k	o	l	o	s	e	u	m	x	y
zašifrovaný text:	L	K	S	O	O	M	E	Y	U	X

b) Dešifrujte text „IRMUDUEMNT“, který vzniknul zašifrováním otevřeného textu pomocí jednoduché transpozice s šifrovacím klíčem $\rho = (142)(35)$. (tentokrát je šifrovací klíč zapsán ve tvaru součinu disjunktních cyklů)
Dešifrovací klíč $\rho^{-1} = (124)(35)$

zašifrovaný text:	I	R	M	U	D	U	E	M	N	T
otevřený text:	r	u	d	i	m	e	n	t	u	m

Afinní šifra

Šifrovací klíč: (a, b), kde $a, b \in Z_{26},\ NSD(a, 26) = 1$

Šifrovací funkce: $E_{(a,b)}(x_1 \dots x_n) = c_1 \dots c_n$,

kde x_i je číselná reprezentace i-tého znaku otevřeného textu,

$c_i = \big((a \cdot x_i + b) \bmod 26\big)$ je číselná reprezentace i-tého znaku šifrového textu.

Dešifrovací klíč: (a^{-1}, b), kde a^{-1} je inverzní prvek k $a \mod 26$

Dešifrovací funkce: $D_{(a^{-1},b)}(c_1 \dots c_n) = x_1 \dots x_n$, kde $x_i = (a^{-1} \cdot (c_i - b) \bmod 26)$.

Poznámky

- Zdůvodněte požadavek $NSD(a, 26) = 1$.
- Při šifrování nejprve převedeme otevřený text $m = m_1 \dots m_n$ na číselný řetězec $x_1 \dots x_n$ např. tak, že každý znak nahradíme jeho pořadím v rámci uvažované otevřené abecedy - viz tabulka č. 1. Analogicky, při dešifrování nejprve převedeme zašifrovaný text na číselný řetězec $c_1 \dots c_n$.

Příklad

Uvažujte afinní šifru s šifrovacím klíčem $e = (a = 17, b = 24)$.

a) Zašifrujte text „vista“.
   Průběh šifrování lze zapsat následovně:
   vista → $(21,8,18,19,0) \xrightarrow{C_i=(17x_i+24\ mod\ 26)} \boldsymbol{c} = (17,4,18,9,24)$ → RESJY

b) Dešifrujte text „BOWLC“.
   Nejprve určíme a^{-1} jako nejmenší nezáporný zbytek modulo 26, který vyhovuje kongruenci $17a^{-1} \equiv 1\ (mod\ 26)$. Např. z tabulky č. 5 určíme, že $a^{-1} = 23$ a tedy dešifrovací funkce má tvar $x_i = (23(C_i - 24)\ \ mod\ 26)$, tj. $x_i = (23C_i + 20\ \ mod\ 26)$
   Průběh dešifrování lze zapsat následovně:
   BOWLC → $(1,14,22,11,2) \xrightarrow{x_i=(23C_i+20\ \ mod\ 26)} \boldsymbol{m} = (17,4,6,13,14)$ → regno

Jednoduchá substituce

Šifrovací klíč: $\pi \in S_{26}$

Šifrovací funkce: $E_\pi(m_1 \dots m_n) = \pi(m_1) \dots \pi(m_n)$

Dešifrovací klíč: $\pi^{-1} \in S_{26}$, kde π^{-1} označuje inverzní permutaci k π

Šifrovací funkce: $D_{\pi^{-1}}(c_1 \dots c_n) = \pi^{-1}(c_1) \dots \pi^{-1}(c_n)$

Poznámky

- V případě monoalfabetických šifer tvoří šifrovací klíč tzv. substituční schéma, což je vzájemně jednoznačné zobrazení otevřené abecedy na šifrovou abecedu. V případě jednoduché substituce je toto zobrazení definováno permutací.

- Alternativní způsob zadání šifrovacího klíče využívá šifrování označované jako substituce s klíčovým slovem. V tomto případě tvoří šifrovací klíč uspořádaná dvojice $(k, \text{textový\_řetezec})$, kde $k \in Z_{26}$. Číslo k definuje pozici (číslujeme od 0), odkud začneme postupně umisťovat znaky textového řetězce (opakující se znaky vynecháváme). V další fázi postupně doplníme chybějící znaky.

Příklad

Otevřený text „aqua fontis“ zašifrujte pomocí jednoduché substituce. Jako šifrovací klíč použijte:

a) $\pi = \begin{pmatrix} a & b & c & d & e & f & g & h & i & j & k & l & m & n & o & p & q & r & s & t & u & v & w & x & y & z \\ D & I & K & W & J & T & Y & V & B & Z & X & P & G & R & A & S & U & L & O & M & C & F & Q & E & N & H \end{pmatrix}$

Schématický zápis šifrování může vypadat následovně

otevřený text:	a	q	u	a	f	o	n	t	i	s
zašifrovaný text:	D	U	C	D	T	A	R	M	B	O

b) $(7, \text{regnum Bohemiae})$

Nejprve na základě klíče vygenerujeme příslušnou permutaci definující substituční schéma. Od 7. znaku (tj. od písmene h) doplňujeme text „regnumBohemiae“ (opakující se znaky vynecháme). V další fázi postupně doplníme chybějící znaky otevřené abecedy.

$\pi = \begin{pmatrix} a & b & c & d & e & f & g & h & i & j & k & l & m & n & o & p & q & r & s & t & u & v & w & x & y & z \\ S & T & V & W & X & Y & Z & R & E & G & N & U & M & B & O & H & I & A & C & D & F & J & K & L & P & Q \end{pmatrix}$

otevřený text:	a	q	u	a	f	o	n	t	i	s
zašifrovaný text:	S	I	F	S	Y	O	B	D	E	C

Hillova šifra

Šifrovací klíč: $H = \left(h_{i,j}\right)_{i,j=1}^{d}$, kde $h_{i,j} \in Z_{26}$

Nejprve rozdělíme otevřený text $\boldsymbol{m}$ na bloky d po sobě jdoucích znaků, tj. $\boldsymbol{m} = \boldsymbol{m}^{(1)} \dots \boldsymbol{m}^{(k)}$, kde $\boldsymbol{m}^{(i)} = m_1^{(i)} \dots m_d^{(i)}$. Následně každý blok $\boldsymbol{m}^{(i)}, i = 1, \dots, k$ převedeme na číselný řetězec $\boldsymbol{x}^{(i)} = \left(x_1^{(i)}, \dots, x_d^{(i)}\right)$, který zašifrujeme pomocí transformace:

$$\boldsymbol{Y}^{(i)} = \boldsymbol{x}^{(i)} \cdot H \ (mod\ 26)$$

$\boldsymbol{Y}^{(i)} = \left(Y_1^{(i)}, \dots, Y_d^{(i)}\right)$ je číselný vektor reprezentující i-tý blok zašifrovaného textu $\boldsymbol{Y} = \boldsymbol{Y}^{(1)} \dots \boldsymbol{Y}^{(k)}$.

Dešifrovací klíč: H^{-1}, tj. matice inverzní k H modulo 26

Dešifrování probíhá zcela analogicky k šifrování, tj. šifrový text rozdělíme na bloky $\boldsymbol{Y}^{(i)}, i = 1, \dots, k$, délky d, které dešifrujeme pomocí inverzní transformace:

$$\boldsymbol{x}^{(i)} = \boldsymbol{Y}^{(i)} \cdot H^{-1} \ (mod\ 26).$$

Poznámky

- Hillova šifra je bloková šifra délky d, tj. pokud není délka otevřeného textu násobkem čísla d, doplníme ho libovolnými znaky na délku rovnou nejbližšímu většímu násobku čísla d.
- Zřejmou podmínkou pro jednoznačné dešifrování je existence inverzní matice H^{-1} (modulo 26). Lze ukázat, že nutnou a postačující podmínkou je v tomto případě $NSD(\det H, 26) = 1$, kde $\det H$ označuje determinant matice H. Připomeňme, že platí $H \cdot H^{-1} \equiv I \ (mod\ 26)$.

- Výpočet H^{-1} se provádí v soustavě Z_{26} a lze využít standardní postupy, např. Gaussovu metodu, determinanty apod.

Příklad

Uvažujte Hillovu šifru s klíčem $H = \begin{pmatrix} 13 & 12 & 21 \\ 22 & 15 & 7 \\ 21 & 3 & 1 \end{pmatrix}$.

a) Zašifrujte text „tarsus“.
Průběh šifrování lze zapsat následovně:
Číselná reprezentace otevřeného textu: tarsus → $(19,0,17,18,20,18)$, ze které sestavíme číselné vektory $\boldsymbol{x^{(i)}}$ délky 3 (řád šifrovací matice). Následně šifrujeme dle vztahu $\boldsymbol{Y^{(i)}} = \boldsymbol{x^{(i)}} \cdot H \ (mod\ 26)$.

$\begin{pmatrix} 19 & 0 & 17 \\ 18 & 20 & 18 \end{pmatrix}\begin{pmatrix} 13 & 12 & 21 \\ 22 & 15 & 7 \\ 21 & 3 & 1 \end{pmatrix} = \begin{pmatrix} 604 & 279 & 416 \\ 1052 & 570 & 536 \end{pmatrix} \equiv_{26} \begin{pmatrix} 6 & 19 & 0 \\ 12 & 24 & 16 \end{pmatrix}$ → GTAMYQ.

b) Dešifrujte text „QASNAL“.
Průběh dešifrování lze popsat následovně – nejprve určíme dešifrovací klíč, tj. matici H^{-1}.

$\left(\begin{array}{ccc|ccc} 13 & 12 & 21 & 1 & 0 & 0 \\ 22 & 15 & 7 & 0 & 1 & 0 \\ 21 & 3 & 1 & 0 & 0 & 1 \end{array}\right) \sim \left(\begin{array}{ccc|ccc} 21 & 3 & 1 & 0 & 0 & 1 \\ 0 & 25 & 8 & 1 & 0 & 13 \\ 0 & 23 & 1 & 0 & 1 & 20 \end{array}\right) \sim \ldots \sim \left(\begin{array}{ccc|ccc} 1 & 0 & 0 & 10 & 19 & 21 \\ 0 & 1 & 0 & 17 & 20 & 23 \\ 0 & 0 & 1 & 25 & 9 & 11 \end{array}\right)$, tedy $H^{-1} = \begin{pmatrix} 10 & 19 & 21 \\ 17 & 20 & 23 \\ 25 & 9 & 11 \end{pmatrix}$

Dále dostáváme QASNAL →$(16,0,18,13,0,11)$, tedy

$\begin{pmatrix} 16 & 0 & 18 \\ 13 & 0 & 11 \end{pmatrix}\begin{pmatrix} 10 & 19 & 21 \\ 17 & 20 & 23 \\ 25 & 9 & 11 \end{pmatrix} = \begin{pmatrix} 610 & 466 & 534 \\ 405 & 346 & 395 \end{pmatrix} \equiv_{26} \begin{pmatrix} 12 & 24 & 14 \\ 15 & 8 & 4 \end{pmatrix}$ → myopie

<u>Vigenèrova šifra</u>

Šifrovací klíč: $\pi_0, \ldots, \pi_{d-1} \in S_{26}$
Šifrovací funkce: $E_{(\pi_0,\ldots,\pi_{d-1})}(m_1 \ldots m_n) = c_1 \ldots c_n$, kde $c_i = \pi_{i \bmod d}(m_i)$

Dešifrovací klíč: $\pi_0^{-1}, \ldots, \pi_{d-1}^{-1} \in S_{26}$, kde π_i^{-1} označuje inverzní permutaci k π_i
Dešifrovací funkce: $D_{(\pi_0^{-1},\ldots,\pi_{d-1}^{-1})}(c_1 \ldots c_n) = m_1 \ldots m_n$, kde $m_i = \pi^{-1}{}_{i \bmod d}(c_i)$

Poznámky

- Vigenèrova šifra je polyalfabetická substituční šifra, jejíž klíč tvoří d cyklicky se opakujících substitučních schémat (šifrových abeced) definovaných permutacemi $\pi_0, \ldots, \pi_{d-1}$.
- Speciálním případem je šifrování pomocí tzv. Vigenèrova čtverce, jehož první řádek tvoří otevřená abeceda a následující řádky reprezentují substituční abecedy vzniklé pouhým posunutím (viz tab. č. 4 v příloze). Šifrovací klíč tak tvoří vektor $(k_0, \ldots, k_{d-1}), k_i \in Z_{26}$ a šifrovací funkce má tvar $E_{(k_0,\ldots,k_{d-1})}(m_1 \ldots m_n) = c_1 \ldots c_n$, kde $c_i = (m_i + k_{i \bmod d})\ mod\ 26$. Dešifrovací funkce má tvar $m_i = (c_i - k_{i \bmod d})\ mod\ 26$.

Příklad

Uvažujte Vigenèrovu šifru s klíčovým slovem „sera“.

a) Zašifrujte text „circumicio“

klíč:	s	e	r	a	s	e	r	a	s	e
otevřený text:	c	i	r	c	u	m	i	c	i	o
zašifrovaný text:	U	M	I	C	M	Q	Z	C	A	S

b) Dešifrujte text „SKXRWWJIG“

klíč:	s	e	r	a	s	e	r	a	s
zašifrovaný text:	S	K	X	R	W	W	J	I	G
otevřený text:	a	g	g	r	e	s	s	i	o

1.3. Binární blokové šifry

Ze zřejmých důvodů převládají v současné době šifrovací metody, které používají binární otevřenou i šifrovací abecedu, tj. $A = B = \{0,1\}$ a tedy šifrují bitový řetězec reprezentující otevřený text na bitový řetězec tvořící šifrový text (obvykle stejné délky).

Poznámky

- V rámci binárního šifrování se používají standardní bitové (logické) operace, zejména pak tzv. vylučující nebo (or exklusive, resp. jen xor) označované $\oplus$.
  Platí $1 \oplus 0 = 0 \oplus 1 = 1; 1 \oplus 1 = 0 \oplus 0 = 0$
- Bitové operace lze rozšířit na operace mezi bitovými řetězci stejné délky tak, že se provedou bitové operace mezi sobě odpovídajícími bity obou bitových řetězců. Např. $1010 \oplus 1100 = 0110$.
- Jsou-li $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \{0,1\}^n$, potom operace $\oplus$ je asociativní, komutativní, má neutrální prvek $\mathbf{0}$ (nulový bitový řetězec délky n) a navíc $\boldsymbol{x} \oplus \boldsymbol{x} = \mathbf{0}$.
- Pro převod otevřeného textu na binární řetězec budeme využívat ASCII tabulku (viz tab. č. 2 v příloze).

Vernamova šifra

Vernamova šifra je bloková šifra, tj. nejprve rozdělíme binární reprezentaci otevřeného textu na po sobě jdoucí bitové řetězce délky n, tj. $\boldsymbol{m} = \boldsymbol{m}^{(1)} \dots \boldsymbol{m}^{(k)}$, kde $\boldsymbol{m}^{(i)} = \left(m_1^{(i)} \dots m_n^{(i)}\right), m_j^{(i)} \in \{0,1\}$. Každý z bitových řetězců $\boldsymbol{m}^{(i)}$ zašifrujeme na bitový řetězec $\boldsymbol{c}^{(i)}$ délky n, tj. výsledný zašifrovaný text má tvar $\boldsymbol{c} = \boldsymbol{c}^{(1)} \dots \boldsymbol{c}^{(k)}$, kde $\boldsymbol{c}^{(i)} = \left(c_1^{(i)} \dots c_n^{(i)}\right), c_j^{(i)} \in \{0,1\}$.

Šifrovací klíč: $\boldsymbol{e} = (e_1 \dots e_d)$, kde $e_i \in \{0,1\}$
Šifrovací funkce: $\boldsymbol{c}^{(i)} = \boldsymbol{m}^{(i)} \oplus \boldsymbol{e}$, kde $\oplus$ je symbol pro operaci xor.

Dešifrovací klíč: $\boldsymbol{e} = (e_1 \dots e_n)$, kde $e_i \in \{0,1\}$
Dešifrovací funkce: $\boldsymbol{m}^{(i)} = \boldsymbol{c}^{(i)} \oplus \boldsymbol{e}$

Poznámky

- Snadno se přesvědčíme, že dešifrování probíhá korektně, neboť
$$\boldsymbol{c}^{(i)} \oplus \boldsymbol{e} = \left(\boldsymbol{m}^{(i)} \oplus \boldsymbol{e}\right) \oplus \boldsymbol{e} = \boldsymbol{m}^{(i)} \oplus (\boldsymbol{e} \oplus \boldsymbol{e}) = \boldsymbol{m}^{(i)} \oplus \mathbf{0} = \boldsymbol{m}^{(i)}$$
- Šifrovací klíč lze zadat pomocí klíčového slova, jehož binární reprezentace tvoří skutečný klíč $\boldsymbol{e}$.

Příklad

Uvažujte Vernamovu šifru s klíčovým slovem „ico“.

a) Zašifrujte text „secus“

Bitová reprezentace klíče: ico = $(01101001\ 01100011\ 01101111)$

otevřený text:	s	e	c	u	s
binární reprezentace:	01110011	01100101	01100011	01110101	01110011
klíč:	01101001	01100011	01101111	01101001	01100011
zašifrovaný text:	00011010	00000110	00001100	00011100	00010000

b) Dešifrujte text (00001111000011000001110100011010)

zašifrovaný text:	00001111	00001100	00011101	00011010
klíč:	01101001	01100011	01101111	01101001
binární reprezentace:	01100110	01101111	01110010	01110011
otevřený text:	f	o	r	s

Důležitou třídu šifer tvoří tzv. Feistelovy šifry, jejichž speciálním případem jsou např. dobře známé šifry DES, NDS. Jde o blokové šifry, které nejprve rozdělí šifrovaný text na po sobě jdoucí bitové řetězce délky $2n$. Každý takový bitový řetězec je pak v několik na sebe navazujících fázích zašifrován na bitový řetězec délky $2n$.

Feistelova šifra

Feistelova šifra je bloková šifra. Nejprve proto binární reprezentaci otevřeného textu $\boldsymbol{m}$ rozdělíme na po sobě jdoucí bitové řetězce $\boldsymbol{m}^{(i)}$ délky $2n$, tj. $\boldsymbol{m} = \boldsymbol{m}^{(1)} \dots \boldsymbol{m}^{(k)}$. Každý z bitových řetězců $\boldsymbol{m}^{(i)}$ pak zašifrujeme v d na sebe navazujících fázích na bitový řetězec $\boldsymbol{c}^{(i)}$ délky $2n$, tj. výsledný zašifrovaný text má tvar $\boldsymbol{c} = \boldsymbol{c}^{(1)} \dots \boldsymbol{c}^{(k)}$.

Šifrovací klíč: $(f_1, \dots, f_d)$, kde $f_i: \{0,1\}^n \to \{0,1\}^n$

Označme $\boldsymbol{m}^{(i)} = \left(m_0^{(i)}, m_1^{(i)}\right)$ bitový řetězec délky $2n$ rozdělený na dva podřetězce $m_0^{(i)}, m_1^{(i)}$, každý délky n. Vlastní šifrovací proces probíhá následovně:

1. fáze: $\left(m_0^{(i)}, m_1^{(i)}\right) \xrightarrow{f_1} \left(m_1^{(i)}, m_2^{(i)}\right)$, kde $m_2^{(i)} = m_0^{(i)} \oplus f_1\left(m_1^{(i)}\right)$

2. fáze: $\left(m_1^{(i)}, m_2^{(i)}\right) \xrightarrow{f_2} \left(m_2^{(i)}, m_3^{(i)}\right)$, kde $m_3^{(i)} = m_1^{(i)} \oplus f_2\left(m_2^{(i)}\right)$

⋮

d. fáze: $\left(m_{d-1}^{(i)}, m_d^{(i)}\right) \xrightarrow{f_d} \left(m_d^{(i)}, m_{d+1}^{(i)}\right)$, kde $m_{d+1}^{(i)} = m_{d-1}^{(i)} \oplus f_d\left(m_d^{(i)}\right)$

závěr: $\boldsymbol{c}^{(i)} = \left(m_{d+1}^{(i)}, m_d^{(i)}\right)$.

Dešifrovací klíč: $(f_d, \dots, f_1)$

Označme $\boldsymbol{c}^{(i)} = \left(c_0^{(i)}, c_1^{(i)}\right)$ bitový řetězec délky $2n$ rozdělený na dva podřetězce $c_0^{(i)}, c_1^{(i)}$, každý délky n. Vlastní dešifrování probíhá analogicky k šifrování, pouze klíče používáme v obráceném pořadí.

1. fáze: $\left(c_0^{(i)}, c_1^{(i)}\right) \xrightarrow{f_d} \left(c_1^{(i)}, c_2^{(i)}\right)$, kde $c_2^{(i)} = c_0^{(i)} \oplus f_d\left(c_1^{(i)}\right)$

2. fáze: $\left(c_1^{(i)}, c_2^{(i)}\right) \xrightarrow{f_{d-1}} \left(c_2^{(i)}, c_3^{(i)}\right)$, kde $c_3^{(i)} = c_1^{(i)} \oplus f_{d-1}\left(c_2^{(i)}\right)$

⋮

d. fáze: $\left(c_{d-1}^{(i)}, c_d^{(i)}\right) \xrightarrow{f_1} \left(c_d^{(i)}, c_{d+1}^{(i)}\right)$, kde $c_{d+1}^{(i)} = c_{d-1}^{(i)} \oplus f_1\left(c_d^{(i)}\right)$

závěr: $\boldsymbol{m} = \left(c_{d+1}^{(i)}, c_d^{(i)}\right)$

Poznámka

Celá řada dnes používaných šifer patří do třídy Feistelových šifer. Jako příklady lze uvést – RC5, RC6, DES (DEA-1), 3DES apod.

- DES (Data Encryption Standard)
  Vyvíjeno firmou IBM (ve spolupráci s NSA) od 70. let 20 století. Šifrují se vždy 64 bitové bloky (tj. $2n = 64$) v 16 fázích (tj. $d = 16$). Klíč tvoří 56 bitový řetězec s tím, že klíče pro jednotlivé fáze jsou různé 48 bitové podřetězce výše zmíněného 56 bitového klíče.
- NDS (New Data Seal)
  Šifrují se 128 bitové bloky (tj. $2n = 128$), používá se 16 fází (tj. $d = 16$) a klíč tvoří pro všechny kroky zobrazení $f: \{0,1\}^8 \to \{0,1\}^8$. Snadno spočteme, že existuje 2^{2048} možností pro volbu f. Pro představu, jde o číslo:
  323170060713110073007148766886699519604441026697154840321303454275246551388678
  908931972014115229134636887179609218980194941195591504909210950881523864482831
  206308773673009960917501977503896521067960576383840675682767922186426197561618
  380943384761704705816458520363050428875758915410658086075523991239303855219143
  333896683424206849747865645694948561760353263220580778056593310261927084603141
  502585928641771167259436037184618573575983511523016459044036976132332872312271
  256847108202097251571017269313234696785425806566979350459972683529986382155251
  663894373355436021354332296046453184786049521481935558536110595962306 56

Příklad

Uvažujte dvou krokovou Feistelovu šifru s klíčem (f_1, f_2), kde

$$f_1(x_1, x_2, x_3, x_4) = (\overline{x_1}, x_2, \overline{x_2 \oplus x_3}, x_4),\ f_2(x_1, x_2, x_3, x_4) = (\overline{x_1} \oplus x_4, \overline{x_2}, x_3, x_2 \oplus \overline{x_4}).$$

a) Zašifrujte text ks, b) dešifrujte binární řetězec 1000110110111011.
(pro binární reprezentaci otevřeného textu užijte ASCII kód).
Řešení.

a) $k = (01101011)$, tedy $\boldsymbol{m}^{(1)} = \left(m_0^{(1)}, m_1^{(1)}\right) = (0110)(1011)$

$(0110,1011) \xrightarrow{f_1} (1011,0111) \xrightarrow{f_2} (0111,1000)$, tedy $\boldsymbol{c}^{(1)} = (10000111)$.

$s = (01110011)$, tedy $\boldsymbol{m}^{(2)} = \left(m_0^{(2)}, m_1^{(2)}\right) = (0111)(0011)$

$(0111,0011) \xrightarrow{f_1} (0011,1110) \xrightarrow{f_2} (1110,0001)$, tedy $\boldsymbol{c}^{(2)} = (00011110)$.

Text ks byl zašifrován na bitový řetězec 1000011100011110.

b) $\boldsymbol{c} = 1000110110111011$, tedy $\boldsymbol{c}^{(1)} = 10001101$ a $\boldsymbol{c}^{(2)} = 10111011$

$\boldsymbol{c}^{(1)}$: $(1000,1101) \xrightarrow{f_2} (1101,0001) \xrightarrow{f_1} (0001,0110)$, tedy $\boldsymbol{m}^{(1)} = (01100001) = a$

$\boldsymbol{c}^{(2)}$: $(1011,1011) \xrightarrow{f_2} (1011,0101) \xrightarrow{f_1} (0101,0110)$, tedy $\boldsymbol{m}^{(2)} = (01100101) = e$

Binární řetězec 1000011100011110 je dešifrován na text *ae*.

1.4. Asymetrická šifra RSA

RSA šifra

Bloková asymetrická šifra (pojmenovaná po autorech Rivest, Shamir, Adleman), která je vyvíjena od roku 1977 a kterou lze dnes považovat prakticky za nejbezpečnější šifru. Nejprve je binární reprezentace otevřeného textu $\boldsymbol{m}$ rozdělená na po sobě jdoucí bitové řetězce $\boldsymbol{m}_i$ délky n, tj. $\boldsymbol{m} = \boldsymbol{m}_1 \dots \boldsymbol{m}_k$. Každý z bitových řetězců $\boldsymbol{m}_i$ je pak zašifrován na bitový řetězec $\boldsymbol{c}_i$ délky n, tj. výsledný zašifrovaný text má tvar $\boldsymbol{c} = \boldsymbol{c}_1 \dots \boldsymbol{c}_k$.

Šifrovací klíč: (n, e), kde n, e jsou vhodně zvolená velká přirozená čísla

Šifrovací transformace: $\boldsymbol{c}_i = (\boldsymbol{m}_i^e \ mod \ n)$

Dešifrovací klíč: (n, d), kde d je vhodně zvolené přirozené číslo

Dešifrovací transformace: $\boldsymbol{m} = \left(\boldsymbol{c}_i^d \ mod \ n\right)$

Poznámky

- Přirozené číslo n má řádově několik stovek cifer a je součinem dvou dostatečně velkých prvočísel p, q, tj. $n = pq$. Číslo e je zvoleno tak, že platí $NSD\left(e, \varphi(n)\right) = 1$, kde φ označuje Eulerovu funkci (vzhledem k volbě n je $\varphi(n) = (p-1)(q-1)$). Číslo d je pak inverzní prvek k e modulo $\varphi(n)$, tj. $de \equiv 1 \ mod \ (p-1)(q-1)$. Nyní snadno nahlédneme, že dešifrování skutečně „funguje“, tj. dešifrovací transformace je inverzní k šifrovací transformaci.
  Zřejmě platí
  $$\boldsymbol{c}_i^d = (\boldsymbol{m}_i^e)^d = \boldsymbol{m}_i^{de} = \boldsymbol{m}_i^{1+t(p-1)(q-1)} = \boldsymbol{m}_i \cdot \boldsymbol{m}_i^{t(p-1)(q-1)}.$$
  Z Eulerovy věty dostáváme
  $$\boldsymbol{m}_i^{t(p-1)(q-1)} = \left(\boldsymbol{m}_i^{(p-1)}\right)^{t(q-1)} \equiv 1 \ (mod \ p) \ \wedge \ \boldsymbol{m}_i^{t(p-1)(q-1)} = \left(\boldsymbol{m}_i^{(q-1)}\right)^{t(p-1)} \equiv 1 \ (mod \ q),$$
  tedy $\boldsymbol{c}_i^d \equiv \boldsymbol{m}_i \ (mod \ p) \ \wedge \ \boldsymbol{c}_i^d \equiv \boldsymbol{m}_i \ (mod \ q)$ a proto $\boldsymbol{c}_i^d \equiv \boldsymbol{m}_i \ (mod \ n)$.
- Zjednodušeně řečeno, bezpečnost šifrovací metody RSA se odvíjí od výpočetní složitosti nalezení kanonického rozkladu velkého přirozeného čísla n. Znalost tohoto rozkladu je totiž nezbytná pro výpočet dešifrovacího klíče d jako řešení kongruence $de \equiv 1 \ mod \ \varphi(n)$.

Příklad

Uvažujte RSA šifrování s veřejným klíčem $(n, e) = (268\,951, 13\,009)$. a) zašifrujte text spinus, b) dešifrujte text 259 339 209 545.

Řešení.

a) Nejprve textový řetězec převedeme na číselný pomocí např. tab. č. 1; bloky tvoří tři znaky) spinus = (18,15,08,13,20,18), tj $\boldsymbol{m} = \boldsymbol{m}_1\boldsymbol{m}_2 = 181508{,}132018$.
   $\boldsymbol{c}_1 = (181\,508^{13\,009} \ mod \ 268\,951) \ \rightarrow \ \boldsymbol{c}_1 = 3\,997$
   $\boldsymbol{c}_2 = (132\,018^{13\,009} \ mod \ 268\,951) \ \rightarrow \ \boldsymbol{c}_2 = 157\,704$
   tedy $\boldsymbol{c} = \boldsymbol{c}_1\boldsymbol{c}_2 = 003\,997\,157\,704$
b) Vzhledem k nepříliš velké hodnotě n určíme snadno kanonický rozklad $n = 599 \cdot 449$ a tedy i dešifrovací klíč d jako řešení kongruence $13009d \equiv 1 \ mod \ \varphi(n)$, tj. $d = 89\,521$.
   Zašifrovaný text $\left(259\,339\,209\,545\right)$ rozdělíme na bloky $\boldsymbol{c} = \boldsymbol{c}_1\boldsymbol{c}_2 = 259\,339{,}209\,545$, tedy
   $\boldsymbol{m}_1 = (259\,339^{89\,521} \ mod \ 268\,951) \ \rightarrow \ \boldsymbol{m}_1 = 201\,908$

$\boldsymbol{m}_2 = (209\,545^{89\,521}\ mod\ 268\,951) \rightarrow \boldsymbol{m}_2 = 110\,818.$
$\boldsymbol{m} = \boldsymbol{m}_1\boldsymbol{m}_2 = 20\ 19\ 08\ 11\ 08\ 18 = utilis$

2. Úvod do kódování

Cílem následujících části textu je seznámit čtenáře se dvěma tématy z oblasti kódování. Jednak s elementárními výsledky z oblasti bezeztrátových kompresních metod, zejména pak s Huffmanovou konstrukcí nejkratšího kódu a s aritmetickými kódy (nultého řádu), dále pak s elementárními výsledky z teorie detekčních, resp. opravných kódů (error-correcting codes), zejména pak s lineárními kódy.

2.1. Základní pojmy

Zdrojová abeceda
Konečná množina $A = \{a_1, \dots, a_r\}$, jejíž prvky budeme nazývat zdrojové znaky. Zdrojovou abecedu interpretujeme jako množinu znaků, které používáme k zápisu původní, tj. nezakódované zprávy (např. anglická/česká abeceda spolu s ciframi 0, 1,…, 9 a dalšími speciálními symboly).

Kódová abeceda
Konečná množina $B = \{b_1, \dots, b_n\}$, jejíž prvky budeme nazývat kódové znaky.
Kódovou abecedu interpretujeme jako množinu znaků, které používáme ke kódování (tj. k zápisu zakódované zprávy). Má-li kódová abeceda n znaků, mluvíme o n-znakovém kódu. Speciálně, kdy $n = 2$, tj. kódová abeceda obsahuje dva znaky (nejčastěji 0, 1), mluvíme o binárním kódu/kódování. V případě $n = 3$ mluvíme o ternárním kódování apod.

Kódování
Kódováním rozumíme libovolné prosté zobrazení K zdrojové abecedy A do množiny B^* (množina všech konečných slov nad abecedou B), tj. $K: A \rightarrow B^*$.

Poznámka
- Kódování lze interpretovat jako „předpis“, který každému zdrojovému znaku $a \in A$ přiřadí slovo $K(a) \in B^*$ vytvořené ze znaků kódové abecedy. Slovo $K(a)$ nazýváme kódové slovo příslušné zdrojovému znaku a.
- Vlastnost „K je prosté“ zajišťuje přirozený požadavek, aby různým znakům zdrojové abecedy odpovídaly různá kódová slova.

Kód
Kódem rozumíme množinu všech kódových slov, tj. množinu $K = \{K(a) \in B^* | \ a \in A\ \}$.

Poznamenejme, že v další části skript nebudeme zcela striktně rozlišovat mezi pojmy kódování (zobrazení) a kód (množina kódových slov) a budeme v obou případech používat označení K.

Kódování zdrojových zpráv

Je-li $K: A \longrightarrow B^*$ kódování, potom zobrazení $K^*: A^* \longrightarrow B^*$ definované pro libovolné slovo $a_{i1} \dots a_{ik}$ nad A vztahem

$$K^*(a_{i1} \dots a_{ik}) = K(a_{i1}) \dots K(a_{ik})$$

(tj. zřetězení kódových slov $K(a_{i1}), \dots, K(a_{ik})$) nazýváme kódováním zdrojových zpráv.

Poznámka

Přirozeným požadavkem je, aby také zobrazení K^* bylo prosté (zdůvodněte). Tato vlastnost však není bezprostředním důsledkem skutečnosti, že zobrazení K je prosté. Tento fakt vede k následující definici.

Jednoznačně dekódovatelné kódování

Řekneme, že K je jednoznačně dekódovatelné kódování, jestliže kódování zdrojových zpráv K^* je prosté zobrazení.

Prefixový kód

Kód nazýváme prefixovým kódem, jestliže žádné kódové slovo není prefixem jiného kódového slova.

Blokový kód

Kód, jehož všechna kódová slova mají stejnou délku, nazýváme blokovým kódem. Počet znaků kódového slova nazýváme délkou blokového kódu.

Poznámky

- Každý prefixový kód je zřejmě jednoznačně dekódovatelný a zakódované zprávy lze dekódovat průběžně „znak po znaku“, tj. není nutné čekat na přijetí celé zprávy. (Zdůvodněte!) Prefixové kódy proto tvoří nejdůležitější třídu kódů.
- Každý blokový kód je prefixový a tedy i jednoznačně dekódovatelný. (Zdůvodněte!)

S pochopitelných důvodů se obvykle snažíme zkonstruovat kódy, které mají co nejkratší kódová slova. Přirozeně tak vzniká otázka, jaké podmínky musí splňovat délky kódových slov u prefixových kódů. Odpověď dává následující tvrzení.

Tvrzení - Kraftova nerovnost

Nechť A je r-znaková zdrojová abeceda. Potom existuje n-znakový prefixový kód zdrojové abecedy A s délkami kódových slov $d_1, \dots, d_r$ právě tehdy, jestliže $\sum_{i=1}^{r} n^{-d_i} \leq 1$.

Důkaz.

Je-li $r = 1$, musí existovat alespoň jedno slovo (nad n-znakovou abecedou) délky d_1, tj. $n^{d_1} \geq 1$, odtud $n^{-d_1} \leq 1$. Je-li $r = 2$, musí být počet všech slov délky d_2 alespoň o 1 větší, než počet slov délky d_2, které mají prefix $K(a_1)$, tj. $n^{d_2-d_1} + 1 \leq n^{d_2}$, tedy $n^{-d_1} + n^{-d_2} \leq 1$. Analogicky pro obecné r musí být počet slov délky d_r alespoň o 1 větší, než počet slov délky d_r, která mají prefixy $K(a_1), \dots, K(a_{r-1})$, tj. $n^{d_r-d_1} + \dots + n^{d_r-d_{r-1}} + 1 \leq n^{d_r}$. Odtud $n^{-d_1} + \dots + n^{-d_{r-1}} + n^{-d_r} \leq 1$.

Poznámka

- V případě binárního kódování má Kraftova nerovnost zřejmě tvar $\sum_{i=1}^{r} 2^{-d_i} \leq 1$.

Tvrzení - McMillanova věta
Pro každé jednoznačně dekódovatelné kódování platí Kraftova nerovnost.

Poznámky
- Důsledkem výše uvedených tvrzení je skutečnost, že se lze bez újmy na obecnosti omezit pouze na prefixové kódy. Zjednodušeně řečeno jsou prefixové kódy stejně obecné jako všechny jednoznačně dekódovatelné kódy, avšak mají navíc tu dobrou vlastnost, že je lze dekódovat průběžně (není třeba čekat na celou zprávu). Z těchto důvodů se v další části skript omezíme pouze na prefixové kódy.
- Kraftova nerovnost dává odpověď na otázku existence prefixového kódu s předepsanými délkami kódových slov. Z praktického hlediska je rozumné požadovat, aby kódová slova nebyla přiřazována znakům zdrojové abecedy nahodile, ale tak, že znaky s vysokou četností (frekvencí, pravděpodobností) výskytu budou zakódována na kratší slova než znaky s nízkou četností. Z těchto důvodů budeme u zdrojové abecedy obvykle uvádět i četnosti jednotlivých znaků. Běžně tak budeme psát $A = \begin{matrix} Znak \\ Pst. \end{matrix}\left|\begin{matrix} a_1 & \dots & a_r \\ p_1 & \dots & p_r \end{matrix}\right.$, kde $p_i > 0, \sum_{i=1}^{r} p_i = 1$, resp. $A = \left\{\begin{matrix} a_1 & \dots & a_r \\ p_1 & \dots & p_r \end{matrix}\right\}$.

Definice - střední délka kódového slova
Nechť $A = \begin{Bmatrix} a_1 & \dots & a_r \\ p_1 & \dots & p_r \\ d_1 & \dots & d_r \end{Bmatrix}$ je zdrojová abeceda, kde p_i označuje četnost znaku a_i a d_i délku kódového slova $K(a_i)$, potom $\bar{d} = \sum_{i=1}^{r} d_i p_i$ nazýváme střední délkou kódového slova.

Definice - nejkratší kód
Nejkratším n-znakovým kódem zdrojové abecedy A rozumíme takový n-znakový prefixový kód zdrojové abecedy, který má ze všech n-znakových prefixových kódů dané abecedy nejmenší střední délku kódového slova.

Poznámky
- Je zřejmé, že nejkratší kód není určen jednoznačně a to nejenom z důvodu možné záměny kódů jednotlivých znaků kódové abecedy (např. u binárního kódu záměna symbolů 0 a 1).
- Návod jak zkonstruovat nejkratší kód dává následující Huffmanova konstrukce nejkratšího kódu.

2.2. Huffmanova konstrukce

Huffmanova konstrukce nejkratšího kódu - binární varianta
Konstrukce nejkratšího binárního kódu probíhá ve dvou na sebe navazujících fázích - redukce a zpětná rekonstrukce.
- Fáze redukce – spočívá v opakované redukci (nahrazení) dvou nejméně četných znaku zdrojové abecedy jedním znakem dle schématu:
  Je-li $A = \left\{\begin{matrix} a_1 & \dots & a_r \\ p_1 & \dots & p_r \end{matrix}\right\}$ zdrojová abeceda seřazená dle četnosti výskytu znaku (tj. $p_1 \geq \dots \geq p_r$), potom redukovaná abeceda má tvar $A^R = \left\{\begin{matrix} a_1 & \dots & a_{r-2} & a^* \\ p_1 & \dots & p_{r-2} & p^* \end{matrix}\right\}$, kde $p^* = p_{r-1} + p_r$.

Nově vzniklou redukovanou abecedu A^R opakovaně redukujeme (po opětovném seřazení znaků dle četností) do okamžiku, než dostaneme abecedu se dvěma znaky (pro tuto abecedu již umíme sestrojit nejkratší binární kód).

- Fáze zpětné rekonstrukce - základem je následující tvrzení:
  Jestliže $\{K(a_1), \dots, K(a_{r-2}), K(a^*)\}$ je nejkratší kód redukované abecedy $A^R = \{a_1, \dots, a_{r-2}, a^*\}$, potom $\{K(a_1), \dots, K(a_{r-2}), K(a^*)0, K(a^*)1\}$ je nejkratší kód neredukované abecedy $A = \{a_1, \dots, a_{r-2}, a_{r-1}, a_r\}$.

Poznámky

Huffmanovu konstrukci nejkratšího kódu znázorníme pomocí binárního kořenového stromu. Při jeho konstrukci budeme využívat následující standardizovaný postup:

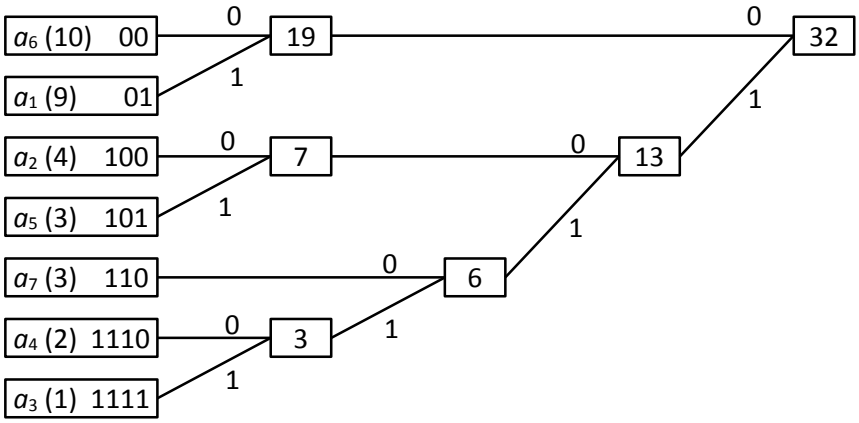
- Nejprve zapíšeme znaky zdrojové abecedy (s jejich četnostmi výskytu) do sloupce, přičemž znaky jsou seřazeny nerostoucím způsobem dle četnosti výskytu. Každý znak původní zdrojové abecedy bude tvořit list výsledného stromu.
- Následně opakujeme redukce dvou nejméně pravděpodobných znaků, přičemž redukovaný znak vytvoří nový vrchol konstruovaného stromu. Tento vrchol umístíme na úroveň vrcholu - redukovaného znaku umístěného výše a spojíme hranou s vrcholy reprezentujícími redukované znaky. Redukce ukončíme v situaci, kdy provedeme redukci abecedy mající již jen dva znaky. Touto poslední redukcí vznikne vrchol – kořen.
- Zpětná rekonstrukce spočívá v přiřazení nejkratšího kódu jednotlivým znakům (tj. listům) původní zdrojové abecedy následovně: z každého uzlu, který není listem, vychází dvě hrany k uzlům, jejichž redukcí uzel vzniknul. „Horní“ hraně přiřadíme znak 0, „spodní“ znak 1. Kódové slovo reprezentující znak původní neredukované abecedy pak tvoří binární slovo, které vznikne zřetězením symbolů na cestě od kořene k listu. Výsledkem je tedy sestrojení substitučního schématu zapsaného v tabulce $\begin{matrix} a_1 & \dots & a_r \\ K(a_1) & \dots & K(a_r) \end{matrix}$, kde $K(a_i)$ je kódové slovo reprezentující znak a_i.

Poznamenejme, že pokud budeme dodržovat výše uvedená pravidla standardizované konstrukce, bude výsledný kód jednoznačný. Této skutečnosti budeme později využívat, zejména u adaptivních kompresních metod, kde je jednoznačnost zásadní pro možnost správného dekódování (dekomprese).

Příklad

Pomocí Huffmanovy konstrukce nalezněte nejkratší binární kód zdrojové abecedy $A = \left\{ \begin{matrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 \\ 9/32 & 4/32 & 1/32 & 2/32 & 3/32 & 10/32 & 3/32 \end{matrix} \right\}$, spočtěte střední délku kódového slova.

Řešení.



Substituční schéma má tvar $\begin{matrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 \\ 01 & 100 & 1111 & 1110 & 101 & 00 & 110 \end{matrix}$. Střední délka kódového slova: $\bar{d} = \frac{1}{32}(2 \cdot 10 + 2 \cdot 9 + 3 \cdot 4 + 3 \cdot 3 + 3 \cdot 3 + 4 \cdot 2 + 4 \cdot 1) = {}^{5}/_{2}$.

Huffmanova konstrukce nejkratšího kódu - obecná varianta

Konstrukce nejkratšího n-árního kódu ($B = \{b_1, \dots, b_n\}$ je kódová abeceda) probíhá zcela analogicky binárnímu případu, tj. ve dvou na sebe navazujících fázích - redukce a zpětná rekonstrukce.

- Fáze redukce - opakovaně provádíme redukce, přičemž u první provedeme redukci posledních s nejméně četných znaků zdrojové abecedy, kde $s \in \{2, \dots, n\}$, navíc s volíme tak, že musí platit $(n-1)|(r-s)$. Ve všech následujících fázích již redukujeme právě n nejméně četných znaků, přičemž redukce ukončíme v situaci, kdy zredukujeme abecedu s právě n znaky (pro tuto abecedu již umíme sestrojit nejkratší n-ární kód).

  Výše popsanou opakovanou redukci znázorníme pomocí n-árního kořenového stromu (listy reprezentují znaky zdrojové abecedy, kořen pak poslední redukci). Při tvorbě stromu dodržujeme stejná pravidla jako u konstrukce binárního stromu (umístění vrcholů, spojení hranami).
- Základem zpětné rekonstrukce je následující tvrzení: Je-li $\{K(a_1), \dots, K(a_{r-s}), K(a^*)\}$ je nejkratší kód redukované abecedy $A^R = \{a_1, \dots, a_{r-s}, a^*\}$, potom $\{K(a_1), \dots, K(a_{r-s}), K(a^*)b_1, \dots, K(a^*)b_s\}$ je nejkratší kód neredukované abecedy $A = \{a_1, \dots, a_{r-s}, \dots, a_r\}$.

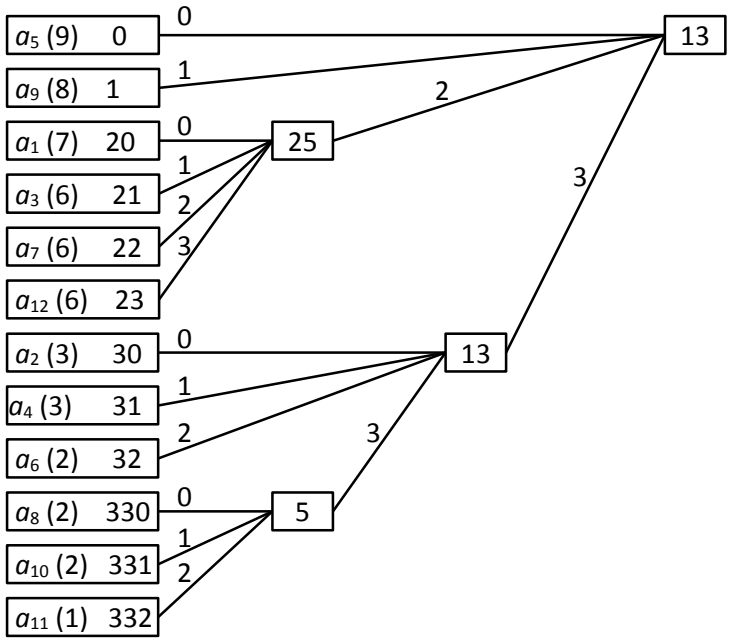
  Fáze zpětné rekonstrukce tedy spočívá v přiřazení nejkratšího n-arního kódu jednotlivým znakům (tj. listům) původní zdrojové abecedy následovně: z každého uzlu, který není listem, vychází n hran k uzlům, jejichž redukcí uzel vzniknul. „Horní“ hraně přiřadíme znak b_1, další znak b_2 atd.. Kódové slovo reprezentující znak původní neredukované abecedy pak tvoří slovo nad abecedou B, které vznikne zřetězením symbolů na cestě od kořene k listu. Výsledkem této fáze je sestrojení substitučního schématu $\begin{matrix} a_1 & \dots & a_r \\ K(a_1) & \dots & K(a_r) \end{matrix}$.

Příklad

Pomocí Huffmanovy konstrukce nalezněte nejkratší čtyřznakový kód zdrojové abecedy $A = \begin{Bmatrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_8 & a_9 & a_{10} & a_{11} & a_{12} \\ {}^{7}/_{55} & {}^{3}/_{55} & {}^{6}/_{55} & {}^{3}/_{55} & {}^{9}/_{55} & {}^{2}/_{55} & {}^{6}/_{55} & {}^{2}/_{55} & {}^{8}/_{55} & {}^{2}/_{55} & {}^{1}/_{55} & {}^{6}/_{55} \end{Bmatrix}$, spočtěte střední délku kódového slova.

Řešení.

Nejprve uspořádáme znaky zdrojové abecedy nerostoucím způsobem dle četnosti výskytu, následně určíme s - počet znaků redukovaných při první redukci. Jelikož $n = 4, r = 12$ a musí platit $(n-1)|(r-s)$, tj. $(3)|(12-s)$, budeme v první fázi redukovat $s = 3$ nejméně četné znaky. V následujících fázích zredukujeme vždy $n = 4$ nejméně četné znaky. Strom znázorňující průběh redukce má následující tvar.



Výsledné substituční schéma má tvar

a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}
20	30	21	31	0	32	22	330	1	331	332	23

.

Pro střední délku kódového slova dostáváme $\bar{d} = {}^{98}/_{55} \cong 1{,}78$

2.3. Aritmetické kódy – metoda DFWLD

V další části se seznámíme s myšlenkou tzv. aritmetických kódů (konkrétně metodou DFWLD), které se řadí k bezeztrátovým kompresním metodám (nultého řádu).

Aritmetické kódy, metoda DFWLD (dyadic fraction with least denominator)

Zdrojová abeceda $A = \begin{Bmatrix} a_1 & \cdots & a_r \\ p_1 & \dots & p_r \end{Bmatrix}$, kde $p_i > 0, \sum_{i=1}^{r} p_i = 1$, navíc předpokládáme, že $p_1 \geq \cdots \geq p_r$.

Dále označme $\boldsymbol{x} = a_{i_1} \dots a_{i_n} \in A^*$ slovo určené k zakódování (kompresi).

Obecný postup aritmetického kódování:

1. Pro jednotlivé prefixy kódovaného slova $\boldsymbol{x}$ postupně konstruujeme posloupnost do sebe vnořených intervalů $\langle 0{,}1) \supset I(a_{i_1}) \supset I(a_{i_1} a_{i_2}) \supset \cdots \supset I(a_{i_1} \dots a_{i_n})$, které jednoznačně reprezentují daný prefix (přesněji intervaly reprezentující všechna slova nad A mající pevnou délku tvoří rozklad intervalu $\langle 0{,}1)$).
2. Z intervalu $I(a_{i_1} \dots a_{i_n})$, který odpovídá kódovanému slovu, vybereme tzv. reprezentanta, tj. číslo $R \in I(a_{i_1} \dots a_{i_n})$, které jednoznačně charakterizuje daný interval.

3. Kód slova $\boldsymbol{x} = a_{i_1} \dots a_{i_n}$ bude tvořit vhodná binární reprezentace čísla R.

Poznámka

- Výše popsaný postup je společný aritmetickým kódům obecně a jednotlivé metody se v podstatě liší pouze způsobem výběru reprezentanta a detaily souvisejícími s jeho binární reprezentací.
- V případě metody DFWLD volíme (jak plyne z názvu - dyadic fraction with least denominator) reprezentanta ve tvaru dyadického zlomku $R = \frac{s}{2^t} \in I(a_{i_1} \dots a_{i_n})$ s nejmenším jmenovatelem.

Detaily DFWLD konstrukce intervalů, výpočtu reprezentanta a jeho binárního zápisu:

i. Konstrukce do sebe vnořených intervalů jednotlivých prefixů $a_{i_1}, a_{i_1}a_{i_2}, \dots, a_{i_1} \dots a_{i_j}, \dots, a_{i_1} \dots a_{i_n}$
Každý interval $I\left(a_{i_1} \dots a_{i_j}\right), j = 1, \dots, n$ je jednoznačně určen svou dolní mezí α_j a délkou l_j, tj. $I\left(a_{i_1} \dots a_{i_j}\right) = \langle \alpha_j, \alpha_j + l_j)$. Při konstrukci postupujeme v podstatě indukcí, tj. na základě α_j a l_j vypočteme α_{j+1} a l_{j+1} následovně
$$\alpha_{j+1} = \alpha_j + l_j \sum_{k<i_{j+1}} p_k \text{ a } l_{j+1} = l_j p_{i_{j+1}},$$
kde $\alpha_0 = 0, l_0 = 1$ (odpovídá základnímu intervalu $\langle 0,1)$).

ii. Výpočet reprezentanta R
Hledáme dyadický zlomek $R = \frac{s}{2^t} \in \langle \alpha_n, \alpha_n + l_n)$ s nejmenším jmenovatelem. Číslo $t \in N^+$ určíme jednoznačně ze zřejmých nerovnic $\frac{1}{2^t} \leq l_n < \frac{1}{2^{t-1}}$. Následně určíme hodnotu $s \in N$ z nerovnic $\alpha_n \leq \frac{s}{2^t} < \alpha_n + l_n$. Těmto nerovnicím vždy vyhovuje alespoň jedna hodnota s, ale nejvýše dvě po sobě jdoucí (v tom případě vždy zvolíme s sudé – zdůvodněte!).

iii. Binární zápis reprezentanta R
Jelikož $R = \frac{s}{2^t} \in \langle 0,1)$, zřejmě $0 \leq s < 2^t$ a tedy R lze zřejmě zapsat ve tvaru $R = (, c_{t-1} \dots c_0)_2$, kde $(c_{t-1} \dots c_0)_2$ je dvojkový zápis přirozeného čísla s pomocí t bitů (v případě potřeby doplníme zleva nuly, např. $(000101)_2$ je dvojkový zápis čísla 5 pomocí šesti bitů).
Kódované slovo $\boldsymbol{x} = a_{i_1} \dots a_{i_n}$ reprezentujeme bitovým řetězcem $c_{t-1} \dots c_0$.

Poznámky

- Zamyslete se nad výše popsanou konstrukcí intervalů jednotlivých prefixů a zdůvodněte skutečnost, že intervaly reprezentující všechna slova délky $n \in N^+$ skutečně tvoří rozklad $\langle 0,1)$. Rozklad zaručuje jednoznačný vztah mezi slovy a intervaly a proto ze znalosti intervalu můžeme jednoznačně rekonstruovat slovo.
- Je třeba si uvědomit, že reprezentant R nezaručuje jednoznačnou rekonstrukci ve smyslu délky rekonstruovaného (dekódovaného, dekomprimovaného) slova. Pro jednoznačnost je nutná ještě znalost délky rekonstruovaného slova (např. $R = 0$, tj. kód 0, reprezentuje libovolně dlouhé slovo obsahující pouze znak a_1).

Příklad

Uvažujte zdrojovou abecedu $A = \begin{Bmatrix} a & b & c & d & e \\ 0{,}3 & 0{,}3 & 0{,}2 & 0{,}1 & 0{,}1 \end{Bmatrix}$. Pomocí metody DFWLD zakódujte slovo $badc$.

Řešení.

Konstrukci intervalů prefixů kódovaného slova lze přehledně zapsat do následující tabulky:

Znak	α_j	l_j	
- - -	0	1	
b	0,3	0,3	$\alpha_1 = 0 + 1 \cdot 0{,}3;\ l_1 = 1 \cdot 0{,}3$
a	0,3	0,09	$\alpha_2 = 0{,}3 + 0{,}3 \cdot 0;\ l_2 = 0{,}3 \cdot 0{,}3$
d	0,372	0,009	$\alpha_3 = 0{,}3 + 0{,}09 \cdot (0{,}3 + 0{,}3 + 0{,}2),\ l_3 = 0{,}09 \cdot 0{,}1$
c	0,3774	0,0018	$\alpha_4 = 0{,}372 + 0{,}009 \cdot (0{,}3 + 0{,}3),\ l_4 = 0{,}009 \cdot 0{,}2$

Nyní stačí určit reprezentanta $R = \frac{s}{2^t} \in \langle 0{,}3774; 0{,}3792)$.
Pro t dostáváme nerovnice $2^{-t} \leq 0{,}0018 < 2^{-t+1}$, tedy $t = 10$.
Pro s máme nerovnice $0{,}3774 \leq \frac{s}{2^{10}} < 0{,}3792$, tedy $s \in \{387{,}388\}$. Jelikož v případě dvou po sobě jdoucích hodnot volíme s sudé, dostáváme $R = \frac{388}{2^{10}} = \frac{97}{2^8} = (,01100001)_2$.
Slovo $badc$ proto zakódujeme na bitový řetězec 01100001.
(Poznamenejme, že ASCII kód by vyžadoval 28 bitů místo 8 bitů.)

Rekonstrukce (dekódování) původního slova probíhá analogicky ke kódování, tj. postupně určujeme intervaly a následně jim odpovídající prefixy, přičemž v každém kroku přidáme na konec již zkonstruovaného prefixu další znak (znalost délky původního slova je nutná proto, abychom věděli, kdy dekódování ukončit). Při rekonstrukci využíváme vlastnosti reprezentanta R, konkrétně

$$\forall j \in \{0, \dots, n-1\}\ \left(\alpha_{j+1} \leq R < \alpha_{j+1} + l_{j+1}\right).$$

Nerovnosti v závorce lze přepsat na tvar

$$\alpha_j + l_j \sum_{k<i_{j+1}} p_k \leq R < \alpha_j + l_j \sum_{k\leq i_{j+1}} p_k,$$

resp.

$$\sum_{k<i_{j+1}} p_k \leq \frac{R-\alpha_j}{l_j} < \sum_{k\leq i_{j+1}} p_k.$$

Z nerovnosti již snadno určíme znak $a_{i_{j+1}}$, který přidáme k již známému prefixu $a_{i_1} \dots a_{i_j}$ (startujeme z prázdného prefixu).

Poznámka
Hodnotu reprezentanta R vypočteme z kódu $c_{t-1} \dots c_0$ dle zřejmého vztahu $R = \sum_{i=0}^{t-1} c_i 2^{i-t}$.

Příklad
Uvažujte zdrojovou abecedu $A = \begin{Bmatrix} a & b & c & d & e & f \\ 0{,}25 & 0{,}25 & 0{,}2 & 0{,}1 & 0{,}1 & 0{,}1 \end{Bmatrix}$. Dekódujte slovo 11000001001001, jestliže délka původního slova byla 5.
Řešení.
Reprezentant má hodnotu $R = 0{,}754456$. Další postup výpočtu je patrný z následující tabulky.

α_j	l_j	$\left(R - \alpha_j\right)/l_j$	Znak
0	1	0,754456	d
0,7	10^{-1}	0,544556	c
0,75	$2 \cdot 10^{-2}$	0,222778	a
0,75	$5 \cdot 10^{-3}$	0,891113	e
0,754	$5 \cdot 10^{-4}$	0,911133	f

Binární řetězec 11000001001001 byl dekódován na text *dcaef*.

Poznámky (dyadické zlomky)

- Nechť $s \in Z, t \in N$. Potom racionální číslo $s/2^t$ nazýváme dyadickým zlomkem. Množina všech dyadických zlomků spolu s operacemi sčítání a násobení tvoří těleso, které je husté v R, tj.
$$\forall x \in R\ \forall \varepsilon > 0\ \exists s \in Z\ \exists t \in N \text{ takové, že } \left|x - s/2^t\right| < \varepsilon.$$
(Jako cvičení ověřte uzavřenost množiny všech dyadických zlomků na sčítání a násobení.)
- K zápisu dyadických zlomků využíváme obvykle dvojkovou soustavu. Konkrétně $s/2^t$ zapisujeme ve tvaru $(d_k \dots d_0, c_t \dots c_0)_2$, kde $(d_k \dots d_0)_2$ je zápis dolní celé části $\left\lfloor s/2^t \right\rfloor$ ve dvojkové soustavě, tj. platí $\left\lfloor s/2^t \right\rfloor = \sum_{i=0}^{k} d_i 2^i$ a $(, c_t \dots c_0)_2$ je zápis lomené části $\left\{ s/2^t \right\}$ ve dvojkové soustavě, tj. platí $\left\{ s/2^t \right\} = \sum_{i=0}^{t} c_i 2^{i-t}$.
- Je-li číslo $\alpha \in (0,1) \cap Q$, lze zjednodušeně popsat konstrukci jeho binárního zápisu následovně (bin_ret bude obsahovat textový řetězec s binární reprezentací čísla α; proměnná $perioda$ nabude hodnoty true v případě zjištění periodického rozvoje):
$bin\_ret \coloneqq$ ",";
repeat $\alpha \coloneqq 2 \cdot \alpha$;
if $\alpha \geq 1$ *then* $bin\_ret \coloneqq bin\_ret$ & "1"; $\alpha \coloneqq \alpha - 1$ *else* $bin\_ret \coloneqq bin\_ret$ & "0";
$\alpha \coloneqq 2 \cdot \alpha$;
until $(\alpha = 0) \vee (perioda)$;

Příklad

a) Sestrojte dyadicky zlomek čísla 5,671875.

b) Určete číslo reprezentované dyadickým zlomkem $(, 011\overline{01})_2$.

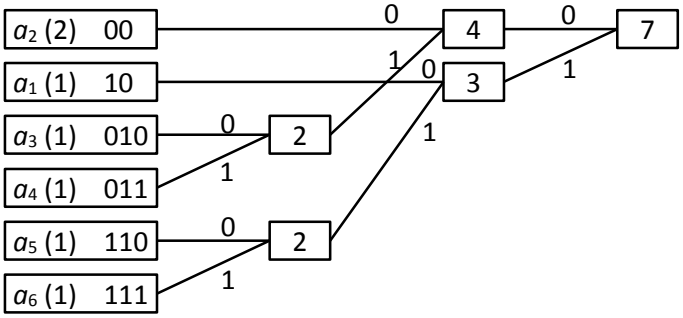
Řešení.

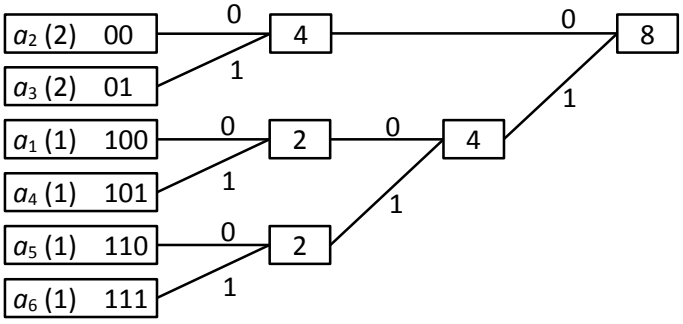
a) $\alpha = 0{,}671875;\ bin\_ret \coloneqq$ ",";
$\alpha \coloneqq 1{,}34375\ (\alpha \coloneqq 2\alpha); bin_{ret} \coloneqq$ ",1"; $\alpha \coloneqq 0{,}34375\ (\alpha \coloneqq \alpha - 1)$;
$\alpha \coloneqq 0{,}6875\ (\alpha \coloneqq 2\alpha); bin_{ret} \coloneqq$ ",10";
$\alpha \coloneqq 1{,}375\ (\alpha \coloneqq 2\alpha); bin_{ret} \coloneqq$ ",101"; $\alpha \coloneqq 0{,}375\ (\alpha \coloneqq \alpha - 1)$;
$\alpha \coloneqq 0{,}75\ (\alpha \coloneqq 2\alpha); bin_{ret} \coloneqq$ ",1010";
$\alpha \coloneqq 1{,}5\ (\alpha \coloneqq 2\alpha); bin_{ret} \coloneqq$ ",10101"; $\alpha \coloneqq 0{,}5\ (\alpha \coloneqq \alpha - 1)$;
$\alpha \coloneqq 1{,}0\ (\alpha \coloneqq 2\alpha); bin_{ret} \coloneqq$ ",101011"; $\alpha \coloneqq 0\ (\alpha \coloneqq \alpha - 1)$;
tedy $5{,}671875 = (101{,}101011)_2$.

b) $(, 011\overline{01})_2 = 2^{-2} + 2^{-3} + 2^{-3} \sum_{n=1}^{\infty} 2^{-2n} = 5/12$

2.4. Adaptivní metody

Zásadní informací ovlivňující účinnost kompresních metod (kompresní poměr) je znalost pravděpodobnosti výskytů znaků zdrojové abecedy. Tyto pravděpodobnosti se obvykle zjišťují a priory a předpokládá se jejich univerzální platnost ve smyslu nezávislosti na právě komprimovaných (kódovaných) datech. Je zřejmé, že takto stanovené četnosti nemusí příliš odpovídat četnostem aktuálně zpracovávaných dat. Jednou z možností, jak tento problém alespoň částečně eliminovat, jsou právě adaptivní metody, které stanovují četnosti průběžně, tj. na základě znalosti již zpracovaných dat.





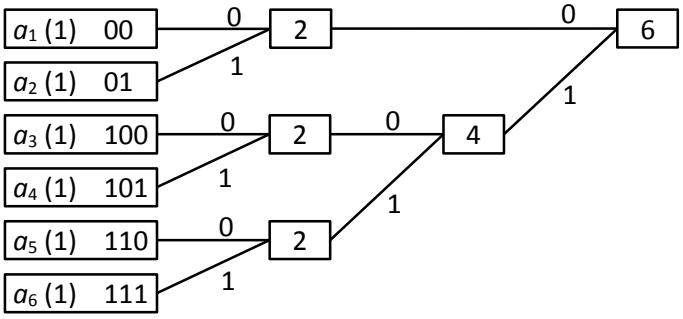
Adaptivní varianta Huffmanovy konstrukce

Nechť $A = \{a_1, \dots, a_r\}$ je zdrojová abeceda, $\boldsymbol{x} = a_{i_1} \dots a_{i_n} \in A^*$ slovo určené k zakódování (kompresi). Zakódování slova délky n proběhne v n na sebe navazujících fázích. Prvním krokem j-té fáze $(j = 1, \dots, n)$ je aktualizace absolutních četností vypočtených z prefixu $a_{i_1} \dots a_{i_{j-1}}$ a následné sestavení nového substitučního schématu (Huffmanovou standardizovanou konstrukcí). Toto substituční schéma použijeme k zakódování znaku a_{i_j}. Poznamenejme, že v první fázi, tj. pro $j = 1$, předpokládáme stejné četnosti všech znaků, tj. rovny 1 (pracujeme s absolutními četnostmi). Postup dokumentuje následující příklad.

Příklad

Uvažujte zdrojovou abecedu $A = \{a_1, a_2, a_3, a_4, a_5, a_6\}$. Pomocí adaptivní Huffmanovy konstrukce zakódujte slovo $a_2a_3a_4a_2a_2a_1$.

Řešení:



Hodnoty uvedené v závorkách jsou absolutní četnosti.

Znak a_2 zakódujeme na 01.

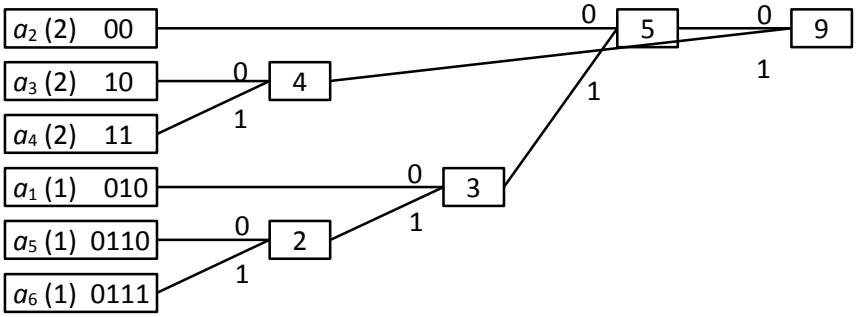
Nyní aktualizujeme četnost znaku a_2, opět seřadíme znaky dle četností a sestavíme nové substituční schéma.

Znak a_3 zakódujeme na 010 a tedy prefix a_2a_3 má kód 01010.

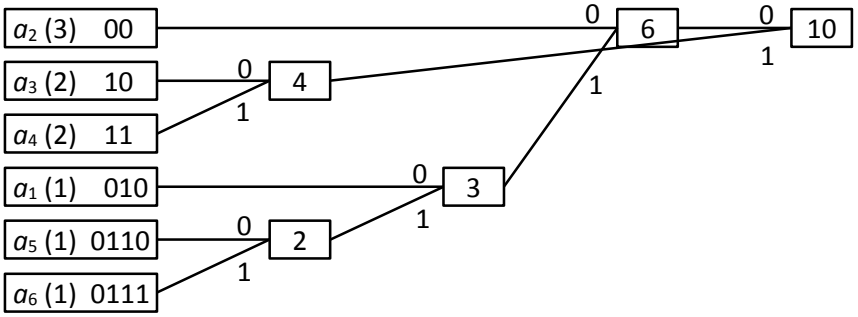
Nyní aktualizujeme četnost znaku a_3, seřadíme znaky dle četností a sestavíme nové substituční schéma.

Znak a_4 zakódujeme na 101 a tedy prefix $a_2a_3a_4$ má kód 01010101.

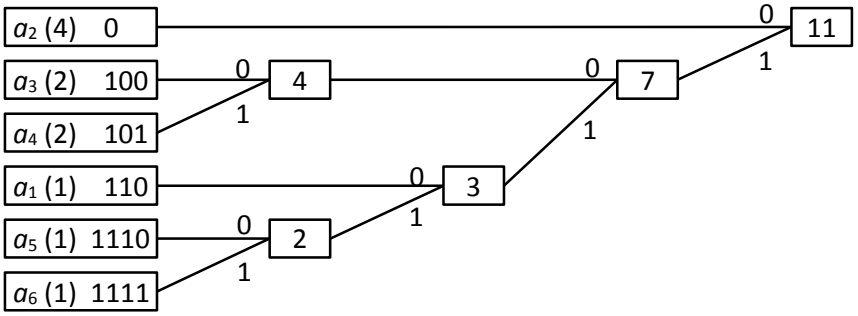
Nyní aktualizujeme četnost znaku a_4, seřadíme znaky dle četností a sestavíme nové substituční schéma.



Znak a_2 kódujeme na 00, prefix $a_2a_3a_4a_2$ má kód 0101010100. Aktualizujeme četnost a_2, seřadíme znaky dle četností, sestavíme nové substituční schéma.



Znak a_2 zakódujeme na 00 a tedy prefix $a_2a_3a_4a_2a_2$ má kód 010101010000. Aktualizujeme četnost a_2, seřadíme znaky dle četností a sestavíme nové substituční schéma.



Znak a_1 zakódujeme na 110.

Znak a_1 zakódujeme na 110.
Zadané slovo $a_2a_3a_4a_2a_2a_1$ má kód 010101010000110.

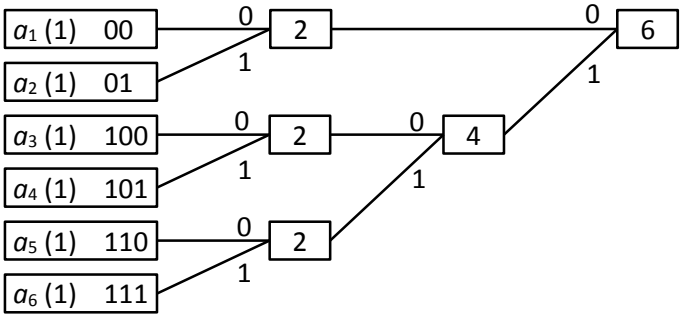
Poznámky

- V případě dekódování (dekomprimace) binárního řetězce vzniklého adaptivní variantou Huffmanovy konstrukce postupujeme analogicky jako při kódování. V úvodu inicializujeme absolutní četnosti všech znaků zdrojové abecedy na 1 a sestavíme substituční schéma, pomocí kterého určíme první znak původního slova. Následně aktualizujeme četnost dekódovaného znaku a sestavíme nové substituční schéma, pomocí kterého určíme další znak. Tento cyklus (aktualizace četnosti → nové substituční schéma → dekódování znaku) opakujeme, dokud není celý řetězec dekódován.
- Víme, že Huffmanova konstrukce není jednoznačná a pokud bychom nezajistili, že při dekódování budou substituční schémata (reprezentována binárními kořenovými stromy) sestavována dle stejných pravidel, nebylo by možné správně dekomprimovat původní text. Z tohoto důvodu budeme používat vždy standardizovanou Huffmanovu konstrukci. Poznamenejme dále, že pokud nově aktualizovaná četnost znaku je rovna četnosti nějakých znaků, zařadíme znak s nově aktualizovanou četností za tyto znaky. Viz konstrukce třetího substitučního schématu v následujícím příkladu - zařazení znaku a_2 s četností 2 za znak a_3.

Příklad

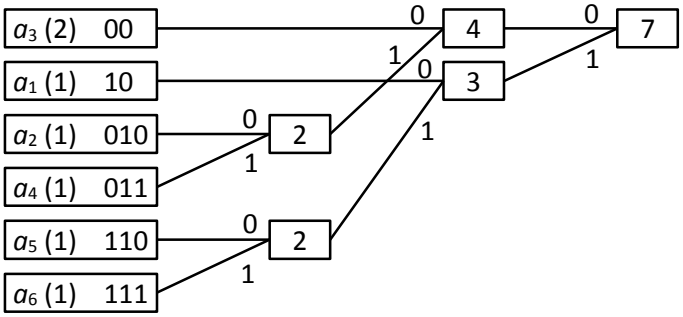
Uvažujte zdrojovou abecedu $A = \{a_1, a_2, a_3, a_4, a_5, a_6\}$. Dekomprimujte (dekódujte) slovo 100010000001111, které vzniklo adaptivní Huffmanovou konstrukcí.

Řešení.



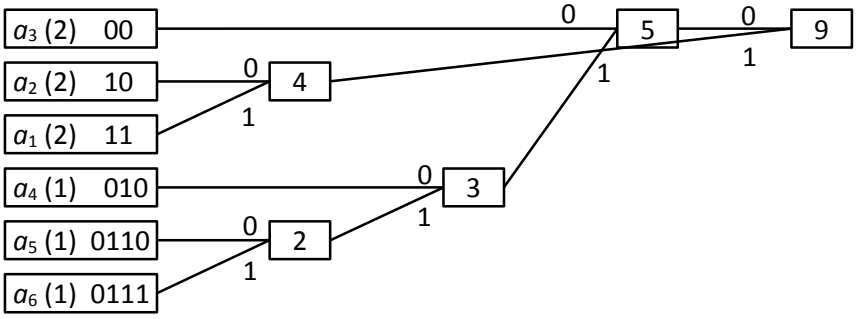
Kódové slovo 100 tvoří prefix dekomprimovaného slova, tudíž první dekódovaný znak je a_3. Zbývá dekomprimovat řetězec 010000001111.

Nyní aktualizujeme četnosti a sestavíme nové substituční schéma.

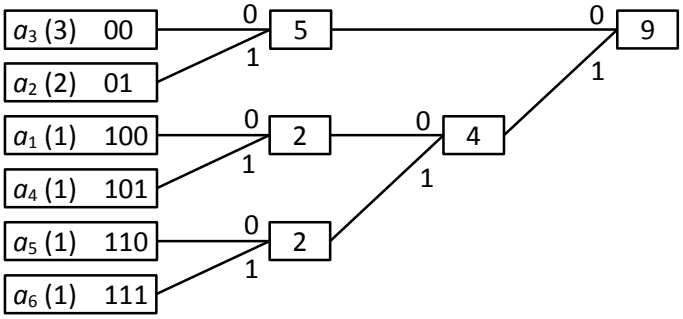


Kódové slovo 010 tvoří prefix, tedy další dekódovaný znak je a_2. Zbývá tak dekomprimovat řetězec 000001111.

Aktualizujeme četnosti a sestavíme další substituční schéma.

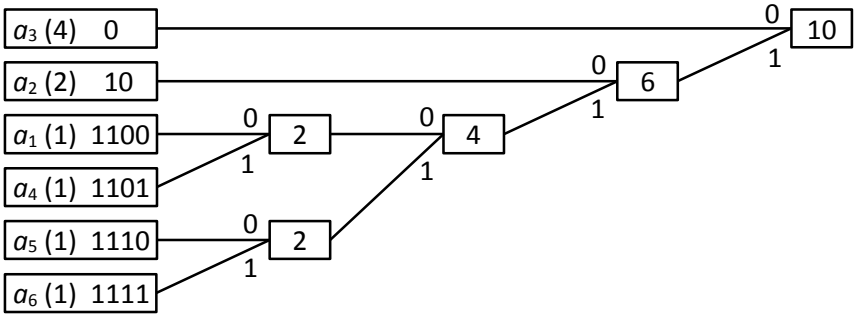


Kódové slovo 00 tvoří prefix, tedy další dekódovaný znak je a_3. Zbývá tak dekomprimovat řetězec 0001111. Aktualizujeme četnosti a sestavíme nové substituční schéma.

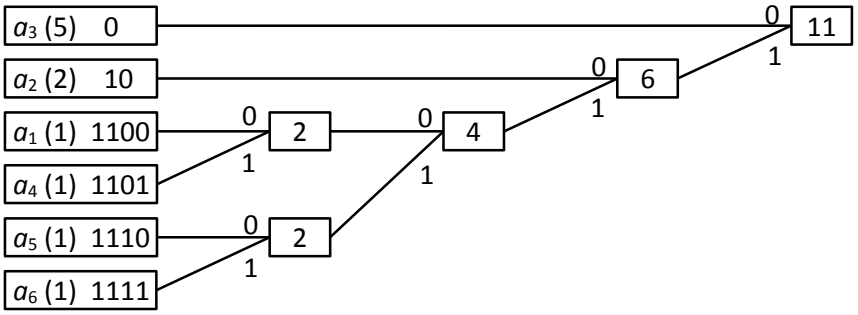


Kódové slovo 00 tvoří prefix, tudíž další dekódovaný znak je a_3. Zbývá tak dekomprimovat řetězec 01111.

Aktualizujeme četnosti a sestavíme další substituční schéma.



Kódové slovo 0 tvoří prefix, tedy další dekódovaný znak je a_3. Zbývá dekomprimovat řetězec 1111.
Aktualizujeme četnosti a sestavíme další substituční schéma.



Kódové slovo 1111 již přímo odpovídá dekódovanému řetězci, tedy další znak je a_6.

Zadaný řetězec 100010000001111 byl dekomprimován na text $a_3a_2a_3a_3a_3a_6$.

Přehled užitého značení

A^*	množina všech (konečných) slov nad (konečnou) abecedou A
N	množina přirozených čísel, tj. $0, 1, \ldots$
N^+	množina kladných přirozených čísel
Z	množina celých čísel
$Q/R/C$	množina racionálních/reálných/komplexních čísel
(a, b)	uspořádaná dvojice prvků a a b
$NSD(a, b)$	největší společný dělitel čísel a, b
S_n	množina všech permutací na n-prvkové množině (symetrická grupa)
Z_p	konečné těleso (úplná soustava zbytků modulo prvočíslo p)
Z_p^n	aritmetický vektorový prostor dimenze n nad konečným tělesem Z_p
$\langle S \rangle$	lineární obal množiny S
K	kód
$K^\perp$	duální kód
G	generující matice kódu
H	kontrolní matice kódu
A^T, A^{-1}	matice transponovaná, inverzní
$\det A$	determinant matice A

Přílohy

a	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*k*	*l*	*m*	*n*	*o*	*p*	*q*	*r*	*s*	*t*	*u*	*v*	*w*	*x*	*y*	*z*
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Tabulka č. 1 - Anglická abeceda a pořadí znaků

Znak	0	1	2	3	4	5	6	7	8	9
ASCII	00110000	00110001	00110010	00110011	00110100	00110101	00110110	00110111	00111000	00111001

Znak	a	b	c	d	e	f	g	h	i	j
ASCII	01100001	01100010	01100011	01100100	01100101	01100110	01100111	01101000	01101001	01101010

Znak	k	l	m	n	o	p	q	r	s	t
ASCII	01101011	01101100	01101101	01101110	01101111	01110000	01110001	01110010	01110011	01110100

Znak	u	v	w	x	y	z
ASCII	01110101	01110110	01110111	01111000	01111001	01111010

Tabulka č. 2 - neúplná ASCII tabulka

a	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*k*	*l*	*m*
8,2	1,5	2,8	4,3	12,7	2,2	2,0	6,1	7,0	0,15	0,77	4,0	2,4

n	*o*	*p*	*q*	*r*	*s*	*t*	*u*	*v*	*w*	*x*	*y*	*z*
6,7	7,5	1,9	0,10	6,0	6,3	9,1	2,8	1,0	2,4	0,15	2,0	

Tabulka č. 3 – četnosti výskytu znaků anglické abecedy (v běžném anglickém textu)

a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**	**k**	**l**	**m**	**n**	**o**	**p**	**q**	**r**	**s**	**t**	**u**	**v**	**w**	**x**	**y**	**z**
B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

Tabulka č. 4 - Vigenèrův čtverec

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
2	2	4	6	8	10	12	14	16	18	20	22	24	0	2	4	6	8	10	12	14	16	18	20	22	24
3	3	6	9	12	15	18	21	24	1	4	7	10	13	16	19	22	25	2	5	8	11	14	17	20	23
4	4	8	12	16	20	24	2	6	10	14	18	22	0	4	8	12	16	20	24	2	6	10	14	18	22
5	5	10	15	20	25	4	9	14	19	24	3	8	13	18	23	2	7	12	17	22	1	6	11	16	21
6	6	12	18	24	4	10	16	22	2	8	14	20	0	6	12	18	24	4	10	16	22	2	8	14	20
7	7	14	21	2	9	16	23	4	11	18	25	6	13	20	1	8	15	22	3	10	17	24	5	12	19
8	8	16	24	6	14	22	4	12	20	2	10	18	0	8	16	24	6	14	22	4	12	20	2	10	18
9	9	18	1	10	19	2	11	20	3	12	21	4	13	22	5	14	23	6	15	24	7	16	25	8	17
10	10	20	4	14	24	8	18	2	12	22	6	16	0	10	20	4	14	24	8	18	2	12	22	6	16
11	11	22	7	18	3	14	25	10	21	6	17	2	13	24	9	20	5	16	1	12	23	8	19	4	15
12	12	24	10	22	8	20	6	18	4	16	2	14	0	12	24	10	22	8	20	6	18	4	16	2	14
13	13	0	13	0	13	0	13	0	13	0	13	0	13	0	13	0	13	0	13	0	13	0	13	0	13
14	14	2	16	4	18	6	20	8	22	10	24	12	0	14	2	16	4	18	6	20	8	22	10	24	12
15	15	4	19	8	23	12	1	16	5	20	9	24	13	2	17	6	21	10	25	14	3	18	7	22	11
16	16	6	22	12	2	18	8	24	14	4	20	10	0	16	6	22	12	2	18	8	24	14	4	20	10
17	17	8	25	16	7	24	15	6	23	14	5	22	13	4	21	12	3	20	11	2	19	10	1	18	9
18	18	10	2	20	12	4	22	14	6	24	16	8	0	18	10	2	20	12	4	22	14	6	24	16	8
19	19	12	5	24	17	10	3	22	15	8	1	20	13	6	25	18	11	4	23	16	9	2	21	14	7
20	20	14	8	2	22	16	10	4	24	18	12	6	0	20	14	8	2	22	16	10	4	24	18	12	6
21	21	16	11	6	1	22	17	12	7	2	23	18	13	8	3	24	19	14	9	4	25	20	15	10	5
22	22	18	14	10	6	2	24	20	16	12	8	4	0	22	18	14	10	6	2	24	20	16	12	8	4
23	23	20	17	14	11	8	5	2	25	22	19	16	13	10	7	4	1	24	21	18	15	12	9	6	3
24	24	22	20	18	16	14	12	10	8	6	4	2	0	24	22	20	18	16	14	12	10	8	6	4	2
25	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

Tabulka č. 5 – Tabulka násobení modulo