

Kompresa dat

Jan Outrata



KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLMOUCI

přednášky



Statistické metody

= zdrojová slova proměnné délky kódována na kódová slova pevné délky $k \geq \lceil \log_m n \rceil$ kódových symbolů = blokový kód, \sim variable-to-fixed code (n je velikost zdrojové abecedy, m je velikost kódové abecedy)

- chyby v kódových slovech se při dekódování nešíří – robustnost

- požadavky:

- 1 každou (neprázdnou) posloupnost zdrojových symbolů musí být možné vyjádřit jako (případně prefix) zřetězení právě jedné posloupnosti zdrojových slov kódovaných na posloupnost kódových slov – jednoznačná kódovatelnost

- 2 průměrná délka zdrojových slov kódovaných na kódová slova je maximální = optimální kód \rightarrow zdrojové symboly s větší pravděpodobností výskytu tvoří delší zdrojová slova

- 3 je použito maximum kódových slov, ideálně všech m^k – optimalita

- prefixový – pro zdrojová slova kódovaná na kódová slova, jednoznačná kódovatelnost

- průměrná délka kódu: $\frac{k}{\sum_{i=1}^t P(w_i)l(w_i)}$, entropie: $\frac{k}{\sum_{i=1}^t P(w_i)\log_2 P(w_i)}$, t počet zdrojových slov w_i délky $l(w_i)$ s pravděpodobností výskytu $P(w_i)$

- pouze statický a semi-adaptivní model

■ B. P. Tunstall

Input : číslo k

Uses : zdrojová abeceda A , $n = |A|$, pravděpodobnosti $P(A^+)$ výskytu slova z A , velikost m kódové abecedy

Output: $C(T)$

$T \leftarrow A$;

$i \leftarrow 0$;

while $n + (i + 1)(n - 1) \leq m^k$ **do**

$x \leftarrow w \in T, P(w) \geq P(w'), w' \in T$;

$T \leftarrow (T \setminus \{x\}) \cup \{xy \mid y \in A\}$;

$i \leftarrow i + 1$;

$C(T) \leftarrow \{\text{kódová slova délky } k\}$ libovolně;

Příklad

vstup: *barbaraabarboraubaru*, $k = 4$, $A = \{a, b, r, u, o\}$, $p(a) = \frac{7}{20}$, $p(b) = \frac{5}{20}$,
 $p(r) = \frac{5}{20}$, $p(u) = \frac{2}{20}$, $p(o) = \frac{1}{20}$, $p(aa) = p(ab) = p(au) = \frac{1}{19}$, $p(ar) = \frac{4}{19}$, $p(ao) = 0$,
 $p(ba) = \frac{4}{19}$, $p(bb) = p(br) = p(bu) = 0$, $p(bo) = \frac{1}{19}$, $m = 2$

$$T = (((\{a, b, r, u, o\} \setminus \{a\}) \cup \{aa, ab, ar, au, ao\}) \setminus \{b\}) \cup \{ba, bb, br, bu, bo\}$$

výstup: 52 b, 2.6 b/symbol

$$H(T) = -\sum_{w \in T} \frac{k P(w)}{P(w) \log_2 P(w)} \doteq 1.385 \text{ b/symbol}$$

$$\bar{l}(C(T)) = \frac{k}{\sum_{w \in T} P(w) l(w)} \doteq 2.405 \text{ b/symbol}$$

$$\bar{l}(C(T)) - \frac{H(T)}{\log_2 2} \doteq 1.02 \text{ b/symbol} \doteq 74 \%$$

- C. E. Shannon, R. M. Fano
- první pokus o optimální binární prefixový kód – využití distribuční funkce/kumulované pravděpodobnosti (cumulative distribution function) zdroje

Input : čísla a, b

Uses : zdrojová abeceda $A = \{a_1, \dots, a_n\}$, $n \geq 2$, pravděpodobnosti $\{p_1, \dots, p_n\}$, $p_i \geq p_j$ pro $i < j$, výskytu a_i

Output : kód $C(A)$

if $a + 1 = b$ **then**

$C(a_a) \leftarrow \mathbf{0}$;

$C(a_b) \leftarrow \mathbf{1}$;

else

najdi j takové, že $|\sum_{i=a}^j p_i - \sum_{i=j+1}^b p_i|$ je minimální;

$C(A) \leftarrow$ zavolej se rekurzivně s a, j , pokud $a < j$, a s $j + 1, b$, pokud $j + 1 < b$, jinak $C(a_a)$ nebo $C(a_b) \leftarrow$ prázdný řetězec;

$C(a_i) \leftarrow \mathbf{0}C(a_i)$ pro $i = a, \dots, j$;

$C(a_i) \leftarrow \mathbf{1}C(a_i)$ pro $i = j + 1, \dots, b$;

Run with: $1, n$

- optimální při $\sum_{i=a}^j p_i = \sum_k p_k$ a $\sum_{i=j+1}^b p_i = \sum_l p_l$, $k, l \in \{a, \dots, b\}$, $(\cup k) \cap (\cup l) = \emptyset$

Příklad

vstup: *barbaraabarboraubaru*, $A = \{a, b, r, u, o\}$, $p(a) = \frac{7}{20}$, $p(b) = \frac{5}{20}$, $p(r) = \frac{5}{20}$,
 $p(u) = \frac{2}{20}$, $p(o) = \frac{1}{20}$

i	a_i	$p(a_i)$	1,5	1,2	3,5	4,5
1	<i>a</i>	7	0	0		
2	<i>b</i>	5	0	I		
3	<i>r</i>	5	I		0	
4	<i>u</i>	2	I		I	0
5	<i>o</i>	1	I		I	I

výstup: **0I00I00I00I000000I00I00I**

IIII000II00I00I0II0, 43 b, 2.15 b/symbol

$$H(A) = -\sum_{i=1}^5 p(a_i) \log_2 p(a_i) \doteq 2.078 \text{ b/symbol}$$

$$\bar{l}(C(A)) = \sum_{i=1}^5 p(a_i) l(a_i) = 2.15 \text{ b/symbol}$$

$$\bar{l}(C(A)) - \frac{H(A)}{\log_2 2} \doteq 0.072 \text{ b/symbol} \doteq 3.46 \%$$



- David A. Huffman: A method for the construction of minimum-redundancy codes. *Proceedings of the I.R.E.*, pp. 1098–1102, 1952.
- optimální prefixové, vyplývá z vlastností optimálního prefixového kódu (viz Věta dříve) a následujícího Lemma

- David A. Huffman: A method for the construction of minimum-redundancy codes. *Proceedings of the I.R.E.*, pp. 1098–1102, 1952.
- optimální prefixové, vyplývá z vlastností optimálního prefixového kódu (viz Věta dříve) a následujícího Lemma

Lemma

Nechť $A = \{a_1, \dots, a_n\}$ je (zdrojová) abeceda s pravděpodobnostmi p_i výskytu symbolů a_i , $A' = \{a'_1 = a_1, \dots, a'_{n'-1} = a_{n-m'}, a'_{n'} = a_{n-m'+1} \dots a_n\}$, $n' = n - m' + 1 \geq 2$, je (zdrojová) abeceda s pravděpodobnostmi

$p'_1 = p_1, \dots, p'_{n'-1} = p_{n-m'}, p'_{n'} = \sum_{j=n-m'+1}^n p_j$ výskytu symbolů a'_i , kde $m' \in \{2, 3, \dots, m\}$, $m' \equiv n \pmod{m-1}$, $m \geq 2$ a $p_{n-m'+1}, \dots, p_n$ jsou nejmenší, a $B = \{b_1, \dots, b_m\}$ je (kódová) abeceda.

Jestliže $C' : A' \mapsto B^+$ je optimální prefixový kód, pak $C : A \mapsto B^+$, $C(a_i) = C'(a'_i)$, $i < n'$, $C(a_{n-m'+j}) = C'(a'_{n'})b_j$, $j \leq m'$, je také optimální prefixový kód.



Statický a semi-adaptivní model

Input : zdrojová abeceda $A = \{a_1, \dots, a_n\}$, $n \geq 1$, pravděpodobnosti $\{p_1, \dots, p_n\}$ výskytu a_i , příznak *první*

Uses : kódová abeceda $B = \{b_1, \dots, b_m\}$, $m \geq 2$

Output : kód $C(A)$

if $n \leq m$ **then**

$C(a_i) \leftarrow b_i$ pro $i = 1, \dots, n$;

else

if *první* = *true* **then**

$m' \leftarrow (n - 2) \bmod (m - 1) + 2$;

else

$m' = m$

$C(A) \leftarrow$ zavolej se rekurzivně s $A' = (A \setminus \{a_{n-m'+1}, \dots, a_n\}) \cup \{a'_{n'} = a_{n-m'+1} \dots a_n\}$,
 $\{p_1, \dots, p_{n-m'}, p'_{n'} = \sum_{i=n-m'+1}^n p_i\}$, kde $p_{n-m'+1} \geq \dots \geq p_n$ jsou nejmenší a *false*;

$C(a_{n-m'+i}) \leftarrow C(a'_{n'})b_i$ pro $i = 1, \dots, m'$;

Run with: $A = \{a_1, \dots, a_n\}$, $\{p_1, \dots, p_n\}$, *true*

Příklad

vstup: *barbaraabarborauaru*, $A = \{a, b, r, u, o\}$, $p(a) = \frac{7}{20}$, $p(b) = \frac{5}{20}$, $p(r) = \frac{5}{20}$,
 $p(u) = \frac{2}{20}$, $p(o) = \frac{1}{20}$, $m = 2$

$\{a, b, r, u, o\}$, $\{7, 5, 5, 2, 1\}$, $C(u) = C(uo)\mathbf{0}$, $C(o) = C(uo)\mathbf{I}$

$\{a, b, r, uo\}$, $\{7, 5, 5, 3\}$, $C(r) = C(ruo)\mathbf{0}$, $C(uo) = C(ruo)\mathbf{I}$

$\{a, b, ruo\}$, $\{7, 5, 8\}$, $C(a) = C(ab)\mathbf{0}$, $C(b) = C(ab)\mathbf{I}$

$\{ab, ruo\}$, $\{12, 8\}$, $C(ab) = \mathbf{0}$, $C(ruo) = \mathbf{I}$

$C(A)$ stejný jako v příkladu u Shannon-Fanova kódování

$m = 3$

$\{a, b, r, u, o\}$, $\{7, 5, 5, 2, 1\}$, $C(r) = C(ruo)\mathbf{0}$, $C(u) = C(ruo)\mathbf{1}$, $C(o) = C(ruo)\mathbf{2}$

$\{a, b, ruo\}$, $\{7, 5, 8\}$, $C(a) = \mathbf{0}$, $C(b) = \mathbf{1}$, $C(ruo) = \mathbf{2}$

výstup: **1020102000102012220021102021**, 28 znaků, 1.4 znaků/symbol

$H(A) = -\sum_{i=1}^5 p(a_i) \log_2 p(a_i) \doteq 2.078$ b/symbol

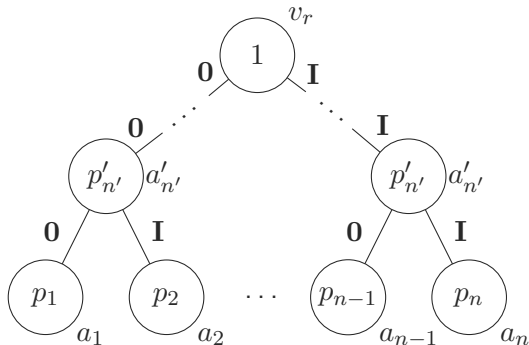
$\bar{l}(C(A)) = \sum_{i=1}^5 p(a_i) l(a_i) = 1.4$ znaků/symbol

$\bar{l}(C(A)) - \frac{H(A)}{\log_2 3} \doteq 0.089$ znaků/symbol $\doteq 6.79\%$

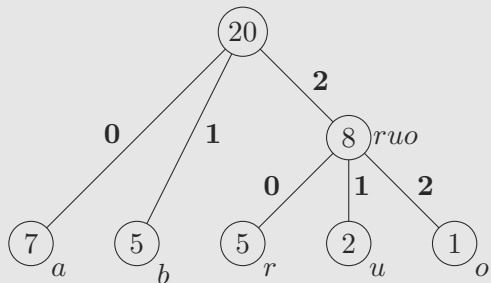
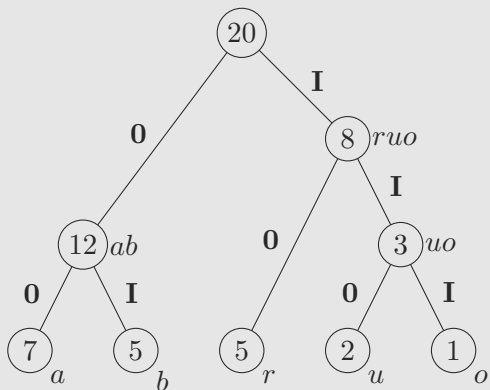
Proč m' a ne m ? Protože při tomto počtu nejdelších kódových slov bude u všech kratších kódových slov a prefixů využito všech m symbolů kódové abecedy a kód má být optimální prefixový.

Huffmanův strom $T_H(A) = \langle V_H(A), E_H(A) \rangle =$ reprezentace Huffmanova kódu $C(A)$ formou m -árního stromu:

- listové uzly $v_l(a_i) \in V_H(A)$ pro symboly $a_i \in A, i = 1, \dots, n$
- vnitřní uzly $v(a'_n) \in V_H(A)$ pro všechny a'_n (ve všech rekurzivních voláních) + kořenový uzel $v_r \in V_H(A)$
- hrany $\langle v(a'_n), v(a_{n-m'+i}) \rangle_{b_i} \in E_H(A)$ a $\langle v_r, v(a'_i) \rangle_{b_i} \in E_H(A)$ označené $b_i \in B, i = 1, \dots, m'$ při $n > m$ a $i = 1, \dots, n$ při $n \leq m$
- $C(a) =$ zřetězení $b \in B$ označujících hrany na cestě stromem z v_r do $v_l(a)$



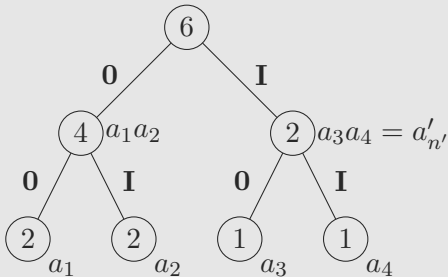
Příklad



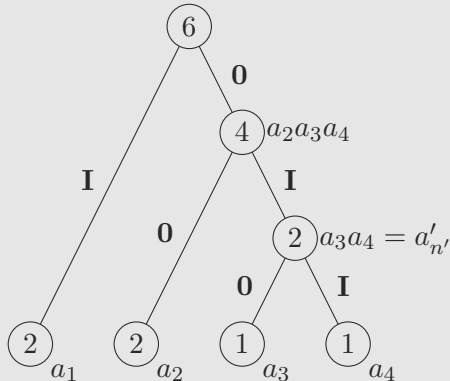
- minimální rozdíly v délkách kódových slov (pro jejich minimální bufferování při pevné rychlosti přenosu/ukládání): zatřídění $a'_{n'}$ před všemi a_j , pro která $p_j = p'_{n'}$, tj. preference a_j před $a'_{n'}$

Příklad

$$A = \{a_1, a_2, a_3, a_4\}, p(a_1) = \frac{2}{6}, p(a_2) = \frac{2}{6}, p(a_3) = \frac{1}{6}, p(a_4) = \frac{1}{6}, m = 2$$

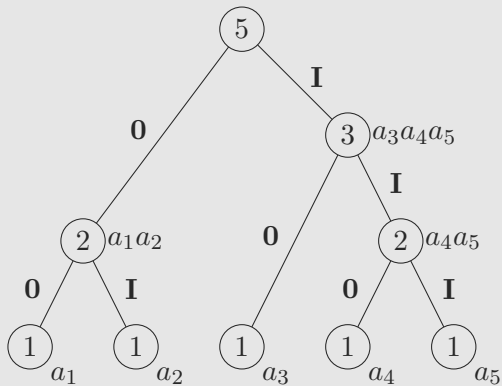
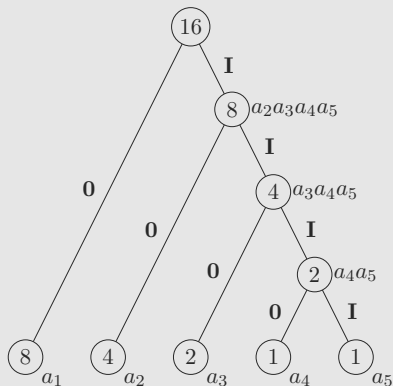


vs.



- $m = 2$ a $p_i > \sum_{j=i+2}^n p_j$, $p_i \geq p_j$ pro $i < j$ (speciálně $p_i = 2^{-i}$, $i = 1, \dots, n-1$ a $p_n = 2^{-(n-1)}$) \rightarrow unární číselný kód $i-1$ pro a_i , $i = 1, \dots, n-1$ a $n-1$ I pro a_n
- $m = 2$ a $p_1 < p_{n-1} + p_n$, $p_i \geq p_j$ pro $i < j$ (speciálně $p_i = \frac{1}{n}$) \rightarrow blokový kód délky $k = \lfloor \log_2 n \rfloor$ pro (např.) a_1, \dots, a_{2^k-1} a délky $k+1$ pro (např.) a_{2^k}, \dots, a_n

Příklad



Adaptivní model – binární kód

- triviálně: znovuvytváření kódu pro každý další symbol na vstupu – výpočetně náročné
- Faller, 1973, Gallager, 1978, Knuth, 1985, Vitter, 1987
- vlastnost Huffmanova stromu (tzv. sibling property): $p_n \leq \dots \leq p_{n-m'+1} \leq p_n \leq \dots \leq p_{n-m'+1}$ následujícího rekurzivního volání – musí stále platit \rightarrow v následujících algoritmech zajištěno pomocí (aktuálního) počtu $n(x)$ výskytů symbolu x a čísla $i(v(x))$, $v(x) \in V_H(A)$
- speciální (escape) symbol ε značící neexistující/první výskyt symbolu

$T_H(A) \leftarrow \langle \{v_l(\varepsilon)\}, \emptyset \rangle$, $C(\varepsilon) \leftarrow$ prázdný řetězec;

$n(\varepsilon) \leftarrow 0$;

$i(v_l(\varepsilon)) \leftarrow 1$;

while načti ze vstupu symbol $a \in A$ **do**

if $v_l(a) \notin V_H(A)$ **then**

 zapiš na výstup $C(\varepsilon)$ a zakódování a ;

else

 zapiš na výstup $C(a)$;

 zavolej následující algoritmus;

Adaptivní model – binární kód

if $v_l(a) \notin V_H(A)$ **then**

$V_H(A) \leftarrow V_H(A) \cup \{v_l(a), v(x = a\varepsilon)\};$

$n(a) \leftarrow 1; n(x) \leftarrow 0;$

$i(v(x)) \leftarrow i(v_l(\varepsilon)); i(v_l(a)) \leftarrow i(v(x)) + 1; i(v_l(\varepsilon)) \leftarrow i(v_l(a)) + 1;$

$E_H(A) \leftarrow (E_H(A) \setminus \{\langle v(w\varepsilon), v_l(\varepsilon) \rangle_b\}) \cup \{\langle v(x), v_l(\varepsilon) \rangle_{\mathbf{I}}, \langle v(x), v_l(a) \rangle_{\mathbf{O}}, \langle v(wx), v(x) \rangle_b\};$

else

$v(x) = v_l(a);$

while $v(x) \neq v_r$ **do**

najdi $v(y)$ takové, že $i(v(y)) \leq i(v(z)), \forall v(z) \in V_H(A), n(y) = n(z) = n(x);$

if $v(y) \neq v(x) \wedge \langle v(y), v(x) \rangle \notin E_H(A)$ **then**

$E_H(A) \leftarrow (E_H(A) \setminus \{\langle v(u = wx), v(x) \rangle_b, \langle v(t = zy), v(y) \rangle_{b'}\}) \cup \{\langle v(u = wy), v(y) \rangle_b, \langle v(t = zx), v(x) \rangle_{b'}\};$

$i \leftarrow i(v(x)); i(v(x)) \leftarrow i(v(y)); i(v(y)) \leftarrow i;$

$n(x) \leftarrow n(x) + 1;$

$v(x) = v(wx), \langle v(wx), v(x) \rangle \in E_H(A);$

$n(x) \leftarrow n(x) + 1;$

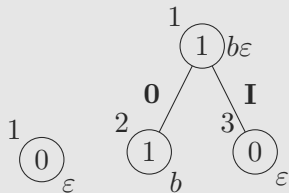
Příklad

1
0
 ϵ

Vstup:

Výstup:

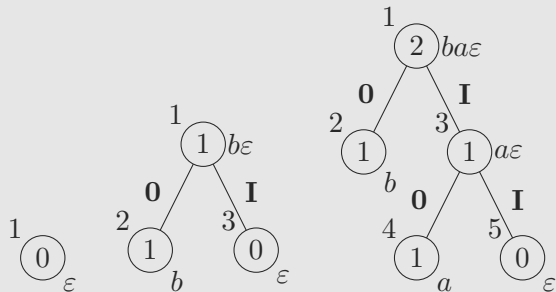
Příklad



Vstup: b

Výstup: b

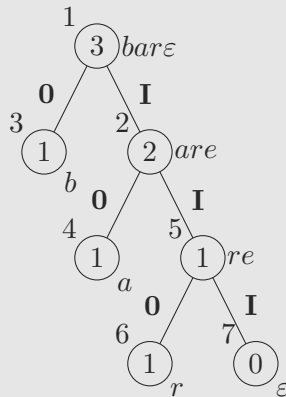
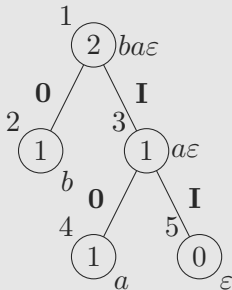
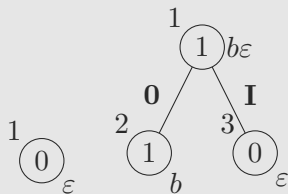
Příklad



Vstup: ba

Výstup: $b\mathbf{I}a$

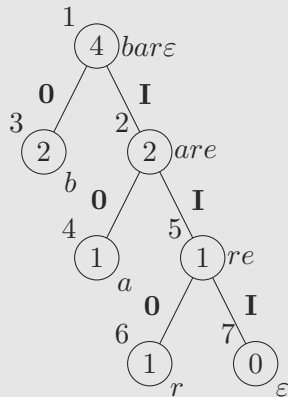
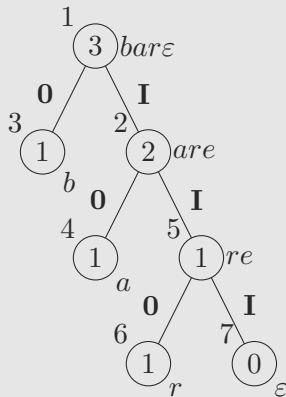
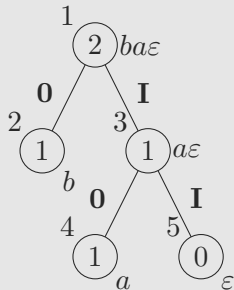
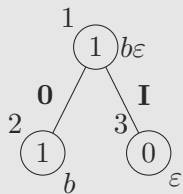
Příklad



Vstup: *bar*

Výstup: *bIaIIr*

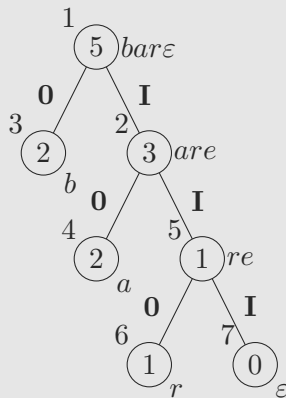
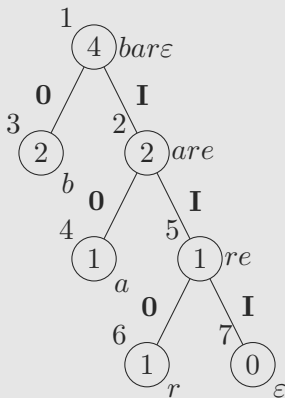
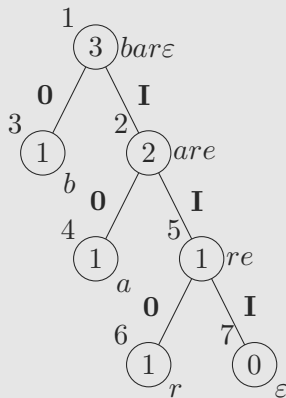
Příklad



Vstup: *barb*

Výstup: *bIaIIr0*

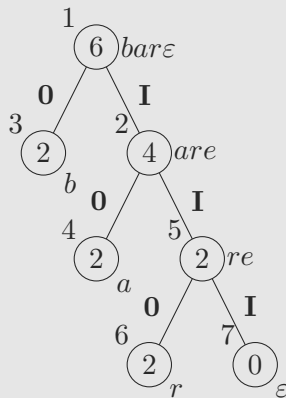
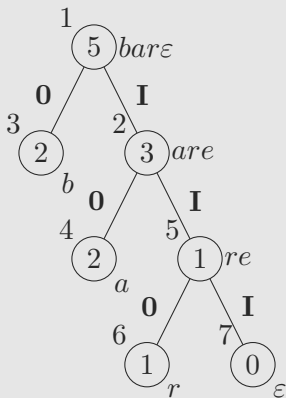
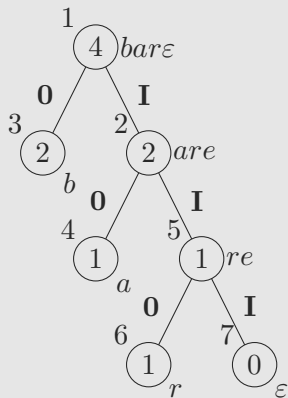
Příklad



Vstup: *barba*

Výstup: ***bIaIIr0I0***

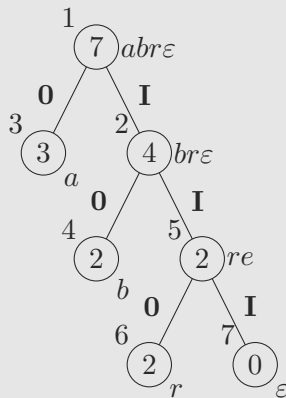
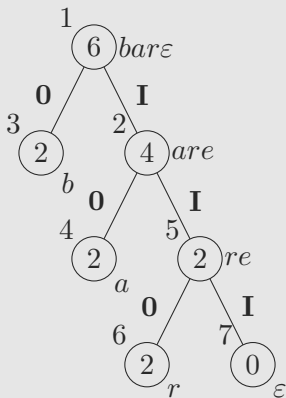
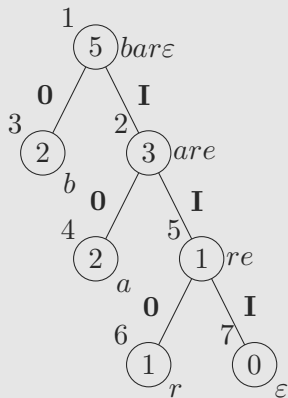
Příklad



Vstup: *barbar*

Výstup: *bIaIIr0I0II0*

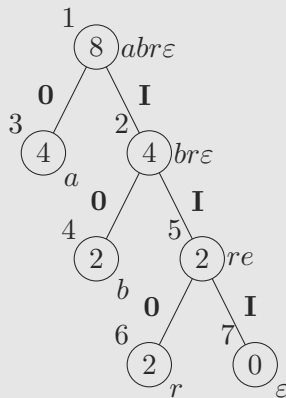
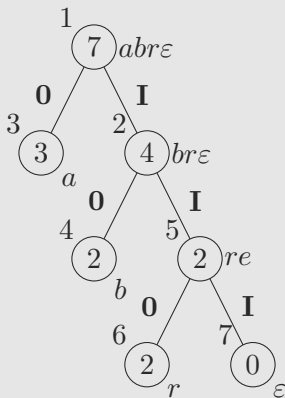
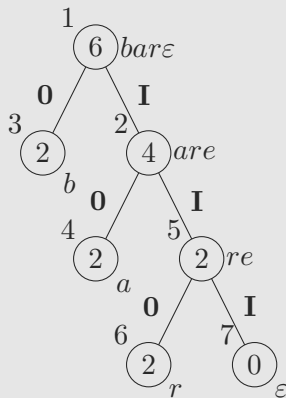
Příklad



Vstup: *barbara*

Výstup: ***bIaIIrOIIOIIIO***

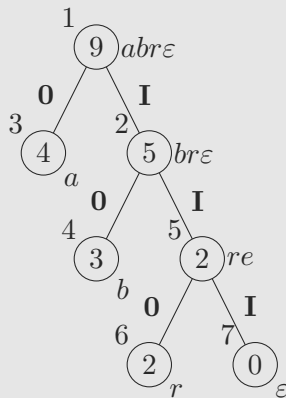
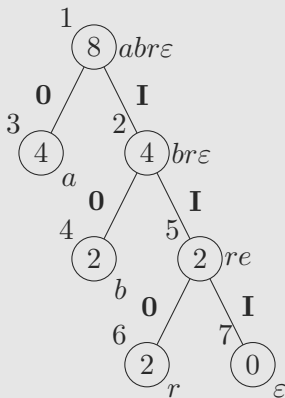
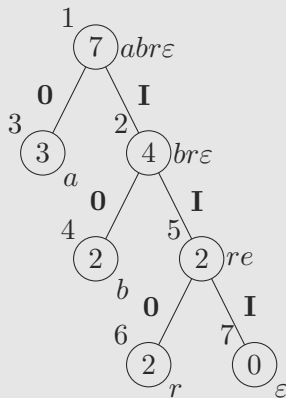
Příklad



Vstup: *barbaraa*

Výstup: ***bIaIIrOIIOIIIOI00***

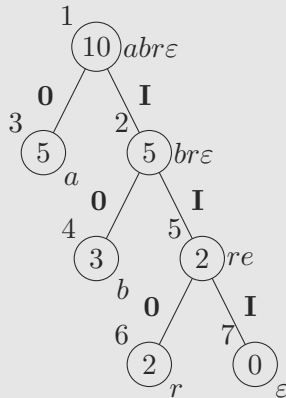
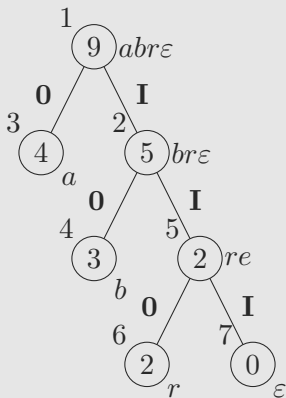
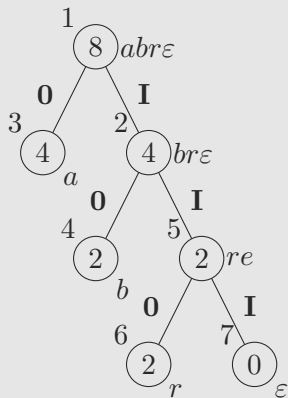
Příklad



Vstup: *barbaraab*

Výstup: ***b1a11r010111010010***

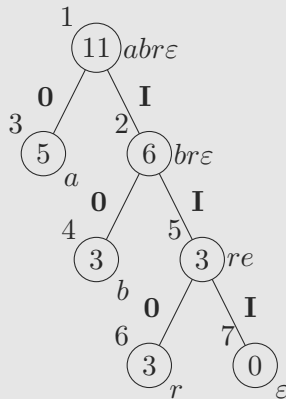
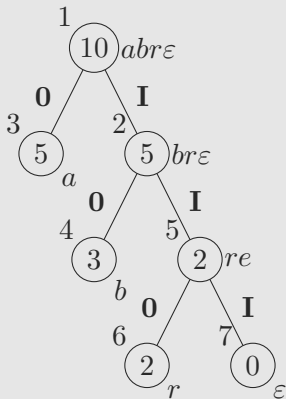
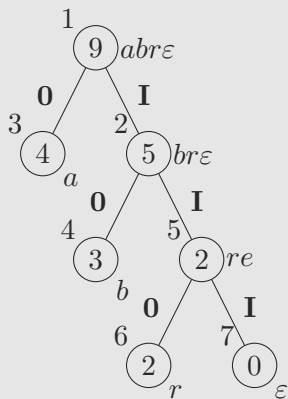
Příklad



Vstup: *barbaraaba*

Výstup: **bIaIIrOIIOIIIOI00I00**

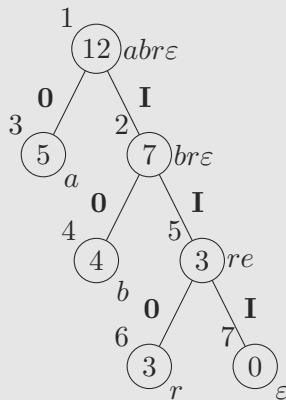
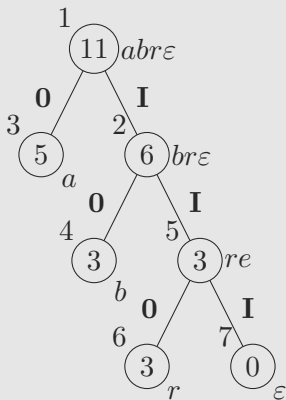
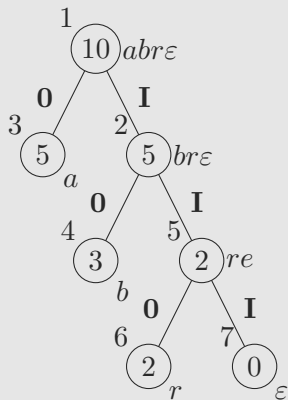
Příklad



Vstup: *barbaraabar*

Výstup: **bIaIIrOIIOIIOOIIOOIIO**

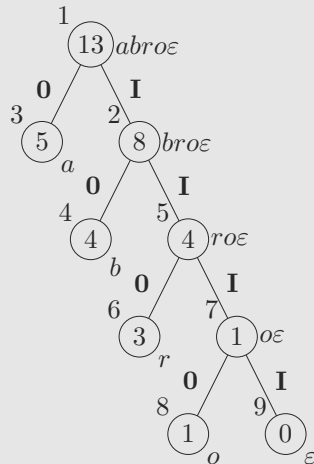
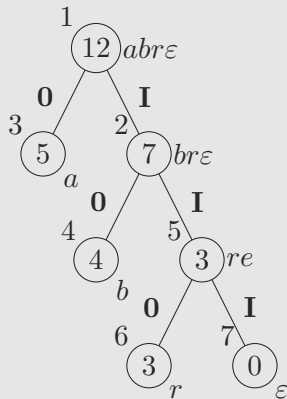
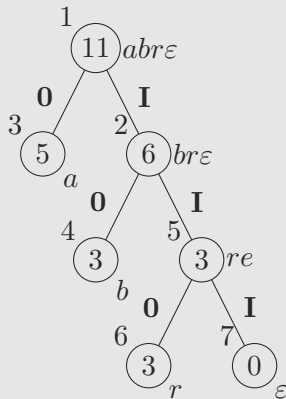
Příklad



Vstup: *barbaraabarb*

Výstup: **bIaIIrOIIOIIIOOIIOOIIIOI**

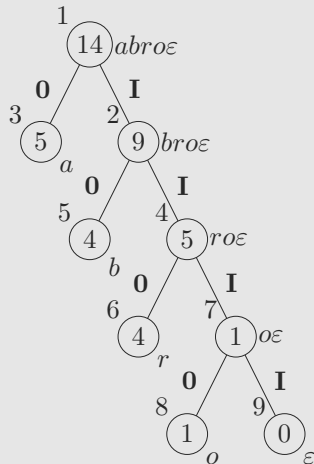
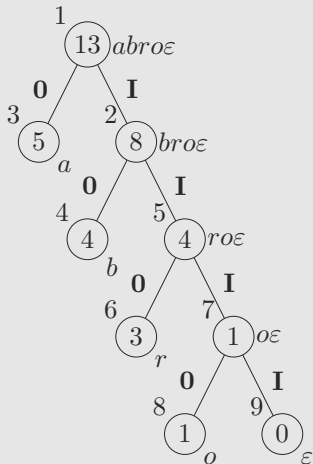
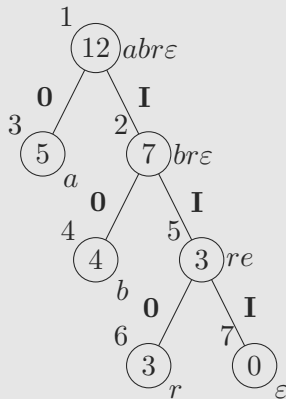
Příklad



Vstup: *barbaraabarbo*

Výstup: *bIaIIrOIIOIIOIIOOIIOOIIOIIOIIIo*

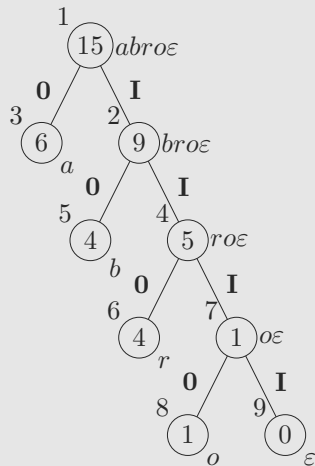
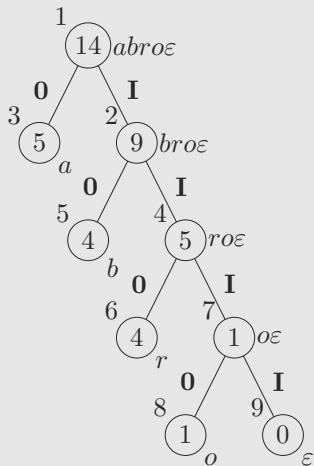
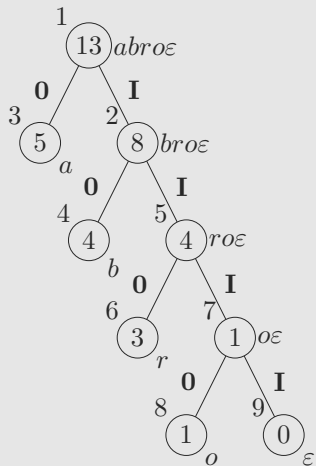
Příklad



Vstup: *barbaraabarbor*

Výstup: *bIaIIrOIIOIIOIIOOIIOOIIOIIOIIIoIIO*

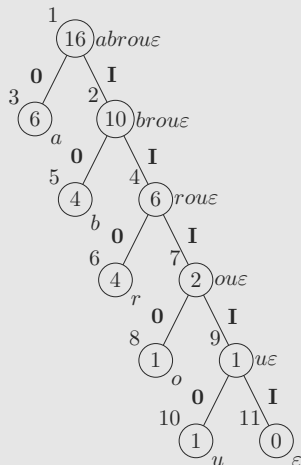
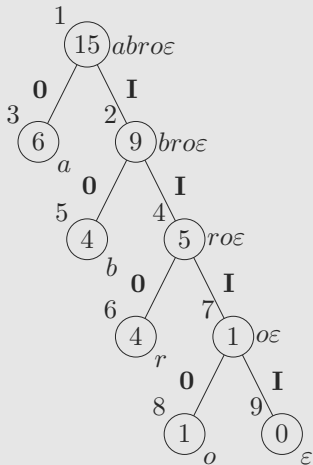
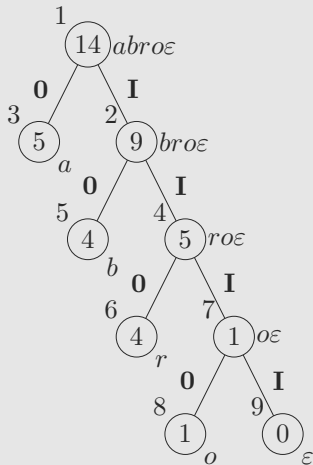
Příklad



Vstup: *barbaraabarbora*

Výstup: **bIaIIrOIIOIIIOOIIIOOIIIOIIOIIIoIIIO**

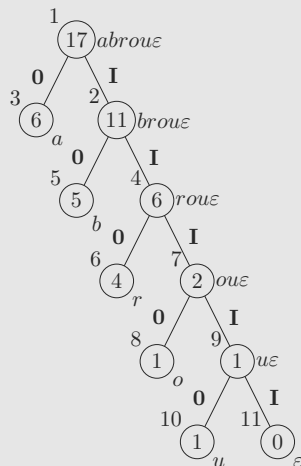
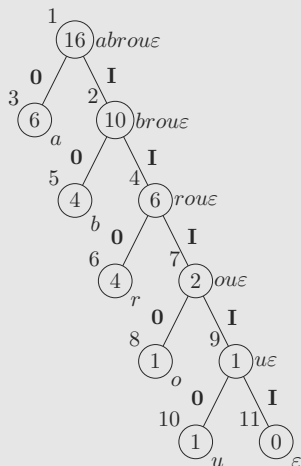
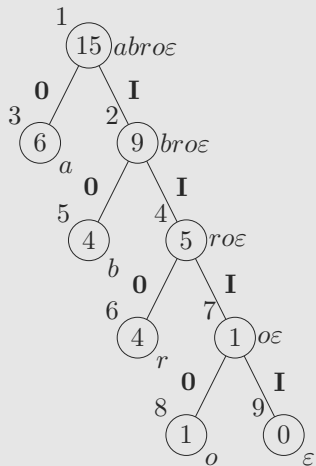
Příklad



Vstup: *barbaraabarborau*

Výstup: *bIaIIrOIIOIIIOOIIIOOIIIOIIOIIIoIIIOOIIIIIu*

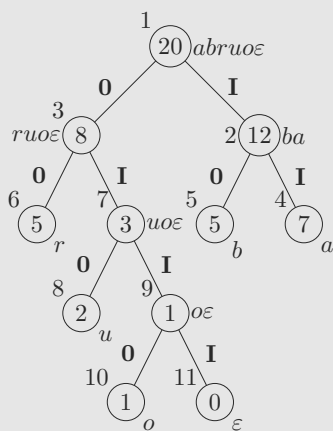
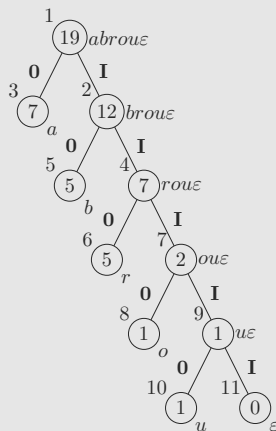
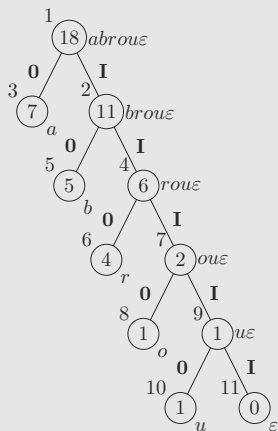
Příklad



Vstup: *barbaraabarboraub*

Výstup: ***bIaIIrOIIOIIIOOIIIOIIOIIIoIIIOOIIIIIuIO***

Příklad



Vstup: *barbaraabarboraubaru*

Výstup: *b1a11r01011010010011010111o11001111u10011011110*, 5 zakódování symbolu + 42 b, 2.1 b/symbol, $\bar{l}(C(A)) = 2.2$ b/symbol, $\bar{l}(C(A)) - \frac{H(A)}{\log_2 2} \doteq 0.122$ b/symbol $\doteq 5.87\%$

Adaptivní model – binární kód

$T_H(A) \leftarrow \langle \{v_l(\varepsilon)\}, \emptyset \rangle;$

$n(\varepsilon) \leftarrow 0;$

$i(v_l(\varepsilon)) \leftarrow 1;$

while není konec vstupu **do**

$v \leftarrow v_r;$

while $v \neq v_l(a)$ **do**

načti ze vstupu symbol $b \in B;$

$v \leftarrow u, \langle v, u \rangle_b \in E_H(A);$

if $a = \varepsilon$ **then**

načti ze vstupu zakódování symbolu $a \in A$ a zapiš na výstup $a;$

else

zapiš na výstup symbol $a \in A;$

zavolej předchozí algoritmus;



Aplikace

- typicky v návaznosti na jiné metody: RLE a MTF, slovníkové, také např. diferenční kódování (obraz, zvuk)



- průměrná délka optimálního prefixového kódu, např. Huffmanova, je minimálně rovna entropii zdroje a nejvýše o 1 větší než entropie (Shannon noiseless coding theorem, viz Věta dříve) – platí těsnější omezení, nejvýše o nejvyšší pravděpodobnost $p_{max} \geq 0.5$ zdrojových symbolů nebo o $p_{max} + 0.086$, $p_{max} < 0.5$
- změna zdrojové abecedy na k -tice (nezávislých) symbolů z původní abecedy A pro přiblížení se entropii ale zvyšuje velikost abecedy, a tím i Huffmanova stromu, na $|A|^k$, např. pro $p_1 = 0.95$, $p_2 = 0.03$, $p_3 = 0.02$ je entropie přibližně 0.335 b/symbol, průměrná délka Huffmanova kódu 1.05 b/symbol, kódu pro 9 dvojic symbolů přibližně 0.611 b/symbol, pro 27 trojic 0.475, pro 81 čtveřic 0.414, pro 243 pětic 0.382, ...!
- ⇒ výhodnější kódovat zdrojová slova než samostatné symboly – ale nevytvářet kód pro všechna slova dané délky, např. Huffmanův!
- kód pouze pro zdrojová slova na vstupu – pořadí ale potřeba jejich abeceda \rightsquigarrow kód pro celý vstup jako jedno zdrojové slovo!
 - vhodné pro malé zdrojové abecedy, např. binární, s velkými rozdíly v pravděpodobnostech symbolů
- = kódování zdrojových slov do čísel z podintervalu $[0, 1)$ kódovaných do binárního kódu

- Pasco, Rissanen, 1976, Rissanen, Langdon, 1979
- využití distribuční funkce/kumulované pravděpodobnosti (cumulative distribution function) $F_X(i) = \sum_{k=1}^i P(X = k)$, $P(X = k) = P(a_k) = p_k$ zdroje (nezávisle s neměnným pravděpodobnostním rozložením se vyskytující) symbolů z abecedy $A = \{a_1, a_2, \dots, a_n\}$ jako náhodných proměnných $X(a_i) = i$, $F_X(0) = 0$

$l \leftarrow 0$; $u \leftarrow 1$;

while načti ze vstupu symbol $a_i \in A$ **do**

$ll \leftarrow l$;

$l \leftarrow l + (u - l)F_X(i - 1)$;

$u \leftarrow ll + (u - ll)F_X(i)$;

// přeškálování $[l, u)$

zapiš na výstup binární reprezentaci (necelé části) čísla z $[l, u)$ s minimem bitů;

Příklad

vstup: *barbaraabarboraubaru*, $A = \{a, b, r, u, o\}$, $p(a = a_1) = \frac{7}{20}$, $p(b = a_2) = \frac{5}{20}$,
 $p(r = a_3) = \frac{5}{20}$, $p(u = a_4) = \frac{2}{20}$, $p(o = a_5) = \frac{1}{20}$
 $F_X(0) = 0$, $F_X(1) = p(a) = \frac{7}{20} = 0.35$, $F_X(2) = p(a) + p(b) = \frac{12}{20} = 0.6$,
 $F_X(3) = p(a) + p(b) + p(r) = \frac{17}{20} = 0.85$, $F_X(4) = p(a) + p(b) + p(r) + p(u) = \frac{19}{20} = 0.95$,
 $F_X(5) = p(a) + p(b) + p(r) + p(u) + p(o) = 1$

vstup	<i>b</i>	<i>a</i>	<i>r</i>	<i>b</i>					
[<i>l</i> , <i>u</i>)	[0.35, 0.6)	[0.35, 0.4375)	[0.4025, 0.424375)	[0.41015625, 0.415625)					
	<i>a</i>		<i>r</i>	<i>a</i>					...
	[0.41015625, 0.4120703125)		[0.4113046875, 0.411783203125)						...

výstup: **OIIIOI00IOIOI** = 0.411376953125

- $C(A^+)$ je prefixový binární kód ze zdrojových slov nad abecedou A , průměrná délka pro slova délky k je $H(A^k) \leq \bar{l}(C(A^k)) < H(A^k) + 1 \Rightarrow$ průměrná délka na symbol z A je $H(A) \leq \bar{l} < H(A) + \frac{1}{k}$
- pro dekódování je nutné znát délku L kódovaného zdrojového slova \rightarrow uložit spolu s komprimovanými daty nebo speciální zdrojový symbol značící konec vstupu

$l \leftarrow 0; u \leftarrow 1;$

$j \leftarrow 0;$

načti ze vstupu binární reprezentaci čísla $x \in [0, 1);$

while $j < L$ **do**

 najdi $i \in \{1, \dots, n\}$ takové, že $F_X(i-1) \leq \frac{x-l}{u-l} < F_X(i);$

 zapiš na výstup symbol $a_i \in A;$

$j \leftarrow j + 1;$

if $j < L$ **then**

$ll \leftarrow l;$

$l \leftarrow l + (u-l)F_X(i-1);$

$u \leftarrow ll + (u-ll)F_X(i);$

 // přeškálování $[l, u)$

- u kódování i dekódování žádoucí průběžný výstup během čtení vstupu, ne až po načtení celého vstupu \rightarrow kód čísla z $[l, u)$ průběžně
- $[l, u)$ se s délkou zdrojového slova zmenšuje, ale uložení necelých čísel je v praxi s omezenou přesností \rightarrow omezení délky slova nebo přeškálování $[l, u)$:

1 $u < 0.5: x \leftarrow 2x, x \in \{l, u\}$

2 $l \geq 0.5: x \leftarrow 2(x - 0.5), x \in \{l, u\}$

3 $l \geq 0.25 \wedge u < 0.75: x \leftarrow 2(x - 0.25), x \in \{l, u\}$

■ případy $\overbrace{3 \dots 31}^{c\text{-krát}} = \overbrace{12 \dots 2}^{c\text{-krát}}$ a $\overbrace{3 \dots 32}^{c\text{-krát}} = \overbrace{21 \dots 1}^{c\text{-krát}}$ \square

```
c ← 0;
// while ...
// přeškálování [l, u):
while u < 0.5 ∨ l ≥ 0.5 ∨ (l ≥ 0.25 ∧ u < 0.75) do
  if u < 0.5 ∨ l ≥ 0.5 then
    if u < 0.5 then // případ 1
      b ← 0;
      d ← 0;
    else // případ 2
      b ← I;
      d ← 0.5;
    zapiš na výstup b;
    while c > 0 do
      zapiš na výstup inverzi b;
      c ← c - 1;
  else // případ 3
    c ← c + 1;
    d ← 0.25;
  l ← 2(l - d);
  u ← 2(u - d);
zapiš na výstup I;
```

Příklad

vstup	b	a	r	b
$[l, u)$	$[0.35, 0.6)$	$[0.2, 0.375)$	$[0.61, 0.6975)$	$[0.5625, 0.65)$
$[l, u)$	$[0.2, 0.7)$	$[0.4, 0.75)$	$[0.22, 0.395), [0.44, 0.79)$	$[0.125, 0.3), [0.25, 0.6), [0, 0.7)$
c	1	0	0	1
výstup		0I	I0	I0
	a	r	a	...
	$[0, 0.245)$	$[0.588, 0.833)$	$[0.176, 0.3475)$...
	$[0, 0.49), [0, 0.98)$	$[0.176, 0.666)$	$[0.352, 0.695), [0.204, 0.89)$...
	0	0	1	...
	0I0	I	0	I

načti ze vstupu $\lceil -\log_2 \frac{p_{min}}{4} \rceil$ bitů binární reprezentace čísla $x \in [0, 1)$, p_{min} nejnižší pravděpodobnost zdrojových symbolů;

```
// while ...
```

```
// přeškálování  $[l, u)$ :
```

```
while  $u < 0.5 \vee l \geq 0.5 \vee (l \geq 0.25 \wedge u < 0.75)$  do
```

```
  if  $u < 0.5 \vee l \geq 0.5$  then
```

```
    if  $u < 0.5$  then
```

```
       $d \leftarrow 0$ ;
```

```
    else
```

```
       $d \leftarrow 0.5$ ;
```

```
  else
```

```
     $d \leftarrow 0.25$ ;
```

```
     $l \leftarrow 2(l - d)$ ;
```

```
     $u \leftarrow 2(u - d)$ ;
```

```
  načti ze vstupu další bit  $b$  anebo  $b \leftarrow 0$ ;
```

```
   $x \leftarrow 2(x - d) + b \cdot 2^{-\lceil -\log_2 \frac{p_{min}}{4} \rceil}$ ;
```

Příklad

$$p_{min} = \frac{1}{20}, \lceil -\log_2 \frac{p_{min}}{4} \rceil = 7$$

vstup: **0110100** = 0.40625 = x

vstup	1	0	10	100
$[l, u)$	[0.35, 0.6)	[0.2, 0.375)	[0.61, 0.6975)	[0.5625, 0.65)
$[l, u)$	[0.2, 0.7)	[0.4, 0.75)	[0.22, 0.395), [0.44, 0.79)	[0.125, 0.3), [0.25, 0.6), [0, 0.7)
x	0.3203125	0.640625	0.2890625, 0.578125	0.1640625, 0.328125, 0.15625
výstup	b	a	r	b
00	0			
	[0, 0.245)	[0.588, 0.833)	[0.176, 0.3475)	
	[0, 0.49), [0, 0.98)	[0.176, 0.666)		
	0.3125, 0.625	0.25		
	a	r	a	

Celočíselná implementace

- zobrazení $[0, 1)$ na $[0, M - 1)$ ($0.25 \mapsto \frac{M}{4}$, $0.5 \mapsto \frac{M}{2}$, $0.75 \mapsto \frac{3M}{4}$), $M \geq 4 \frac{1}{p_{min}}$, M typicky 2^8 , 2^{16} , 2^{32} nebo 2^{64} podle datového typu pro čísla z $[0, M - 1)$
- odhad $F_X(i)$ s frekvencemi/četnostmi $f(a_k) = \frac{n(a_k)}{\sum_{j=1}^{|A|} n(a_j)}$ výskytu symbolu $a_k \in A$ místo pravděpodobností $P(a_k)$
- $l \leftarrow l + \lfloor (u - l + 1)F_X(i - 1) \rfloor$, $u \leftarrow ll + \lfloor (u - ll + 1)F_X(i) \rfloor - 1$, $\frac{x-l+1}{u-l+1}$,
 $u \leftarrow 2(u - d) + 1$, $x \leftarrow 2(x - d) + b -$ kvůli celočíselné aritmetice
- načti ze vstupu $\lceil \log_2 M \rceil$ bitů binární reprezentace čísla $x \in [0, M - 1)$
- při $M = 2^k$ pro nějaké $k \geq 2$:
 - $u < \frac{M}{2} \rightarrow$ nejvýznamnější bit binární reprezentace l i u je **0**
 - $l \geq \frac{M}{2} \rightarrow$ nejvýznamnější bit binární reprezentace l i u je **1**
 - $l \geq \frac{M}{4} \wedge u < \frac{3M}{4} \rightarrow$ druhý nejvýznamnější bit binární reprezentace l je **1** a u je **0**
 - $2x$ a $2(x - \frac{M}{2}) \rightarrow$ bitový posun x doleva o 1 bit, $2(x - \frac{M}{4}) \rightarrow$ navíc inverze (nového) nejvýznamnějšího bitu binární reprezentace

Příklad

$$M = 256 \geq 4 \frac{1}{20}$$

vstup	b	a	r	b
$[l, u)$	[89, 152)	[50, 93)	[177, 220)	[129, 172)
$[l, u)$	[50, 177)	[100, 187), [72, 247)	[98, 185), [68, 243)	[2, 89), [4, 179)
c	1	1	1	0
výstup		0I	I0	I00
a		r	a	...
[4, 64)		[81, 110)	[8, 91)	...
[8, 129)	[162, 221), [68, 187), [8, 247)		[16, 183)	...
0		1	0	...
0		0I	0I	I

Příklad

$$\lceil -\log_2 256 \rceil = 8$$

vstup: **01101000** = 104 = x

vstup	0	10	11	00
$[l, u)$	[89, 152)	[50, 93)	[177, 220)	[129, 172)
$[l, u)$	[50, 177)	[100, 187), [72, 247)	[98, 185), [68, 243)	[2, 89), [4, 179)
x	80	161, 194	133, 139	22, 44
výstup	b	a	r	b
	0	000		
$[4, 64)$		[81, 110)	[8, 91)	
[8, 129)	[162, 221), [68, 187), [8, 247)			
88	176, 96, 64			
a	r	a		

Adaptivní model

- průběžný odhad $F_X(i)$ s frekvencemi/četnostmi $f(a_k) = \frac{n(a_k)}{\sum_{j=1}^{|A|} n(a_j)}$ výskytu symbolu $a_k \in A$ místo pravděpodobností $P(a_k)$
 - inicializace na známý odhad nebo počet $n(a) = 1$ výskytu každého symbolu $a \in A$
 - inkrementace $n(a)$ po (de)kódování symbolu a
- $p_{min} = \frac{1}{|A|}$, u celočíselné implementace nesmí p_{min} klesnout pod $4\frac{1}{M} \rightarrow$ v praxi $n(a_j) \leftarrow \frac{n(a_j)}{2}$ pro $n(a_j) > 1$ při $\sum_{j=1}^{|A|} n(a_j) = \frac{M}{4}$

Aplikace

- v novějších standardech komprese multimediálních dat (WebP, AVIF, AV1, Opus) – ve starších (JPEG, MPEG1–4) kvůli patentům dodnes pouze Huffmanovo

- = modifikované adaptivní binární aritmetické kódování (ABAC, Q kódování, skew kódování), tj. pro binární zdrojovou abecedu s adaptivním modelem
 - $A = u - l$ místo u : $l \leftarrow l + A \cdot F_X(i - 1)$ a $A \leftarrow A \cdot p_i$
 - místo 2 symbolů A , $a_1 = \mathbf{0}$ a $a_2 = \mathbf{1}$, $a_1 =$ více (MPS) a $a_2 =$ méně (LPS) pravděpodobný symbol s průběžně odhadovanými frekvencemi/četnostmi $1 - q$ a q výskytu
 - $l \leftarrow l$ a $A \leftarrow A(1 - q)$ pro MPS
 - $l \leftarrow l + A(1 - q)$ a $A \leftarrow Aq$ pro LPS
 - $\frac{x-l}{A} < 1 - q$ pro MPS a $\geq 1 - q$ pro LPS
 - potlačení násobení (i pro nebinární zdrojové abecedy) – udržování hodnoty A v $[0.75, 1.5)$ a zanedbání násobení A
 - $l \leftarrow l$ a $A \leftarrow A - q$ pro MPS
 - $l \leftarrow l + A - q$ a $A \leftarrow q$ pro LPS
 - $x - l < A - q$ pro MPS a $\geq A - q$ pro LPS
 - přeškálování l, A : $A < 0.75$: $A \leftarrow 2A$, $l \leftarrow 2l$ pro $l < 0.5$ a $l \leftarrow 2(l - 0.5)$ pro $l \geq 0.5$



```
while  $A < 0.75$  do  
  if  $l < 0.5$  then  
     $b \leftarrow 0$ ;  
     $d \leftarrow 0$ ;  
  else  
     $b \leftarrow 1$ ;  
     $d \leftarrow 0.5$ ;  
  zapiš na výstup  $b$ ;  
   $l \leftarrow 2(l - d)$ ;  
   $A \leftarrow 2A$ ;
```

zapiš na výstup $C = 1$;

načti ze vstupu $\lceil -\log_2 \frac{q}{4} \rceil$ bitů binární reprezentace čísla $x \in [0, 1)$;

// while ...

```
while  $A < 0.75$  do  
  if  $l < 0.5$  then  
     $d \leftarrow 0$ ;  
  else  
     $d \leftarrow 0.5$ ;  
   $l \leftarrow 2(l - d)$ ;  
   $A \leftarrow 2A$ ;  
  načti ze vstupu další bit  $b$  anebo  $b \leftarrow 0$ ;  
   $x \leftarrow 2(x - d) + b \cdot 2^{-\lceil -\log_2 \frac{q}{4} \rceil}$ ;
```



- inicializace $q = 0.5$ a aktualizace q v praxi podle tabulky hodnot při přeškálování, ne po (de)kódování každého symbolu
- při častějším výskytu LPS než MPS, $q > A - q$, prohození symbolů včetně aktuálních hodnot l a A , před prvním přeškálováním
- celočíselná implementace: zobrazení $[0, 1.5)$ na $[0, M - 1)$ ($0.25 \mapsto \frac{M}{6}$, $0.5 \mapsto \frac{M}{3}$, $0.75 \mapsto \frac{M}{2}$, $1 \mapsto \frac{2M}{3}$), $M \geq \frac{4}{3}q$, i pro hodnoty q
- použití v novějších standardech komprese videa (H.264, H.265) – kontextově ABAC (CABAC)