Bezstratová kompresia údajov

ÚINF/KMÚ ZS 2019/20
doc. RNDr. Jozef Jirásek, PhD.

Uchovávanie a prenos informácie

- stratová a bezstratová kompresia (komprimácia)
- kompresný pomer = $\frac{\text{veľkosť komprimovaného kódu}}{\text{veľkosť zdrojového kódu}}$ [%]
- kompresný faktor = $\frac{\text{veľkosť zdrojového kódu}}{\text{veľkosť komprimovaného kódu}}$
- veľkosti v porovnateľných jednotkách (bit, bajt)
- nie je možné zostaviť algoritmus, ktorý by pre všetky vstupy mal kompresný pomer < 1

RLE kompresia

- jednoduchá kompresia reťazca bajtov pomocou behov s rovnakými hodnotami
- kódové slová - dvojice (symbol, dĺžka behu)
- aaaabbbbbbbccbbbbaaaa (a,4)(b,7)(c,2)(b,4)(a,4)
- špeciálne dvojice (EOL,0) (EOF,0) (MTI,0,x,y)
  pre n≥3 (n,0,s_1,s_2, ...,s_n)

Rice-Golombove kódy

binárne kódy s nerovnakou dĺžkou kódového slova (na efektívnejšie kódovanie malých čísel)

- parameter m
- číslo n rozložíme na $q = \left\lfloor \frac{n}{m} \right\rfloor$ a $r = n - qm$
- q kódujeme unárnym kódom (q jednotiek)
- pre jednoznačnú dekódovateľnosť vložíme 0
- pre $0 \leq r < 2^{\lceil \log m \rceil} - m$ kódujeme r binárne $\lfloor \log m \rfloor$ bitmi
- pre $2^{\lceil \log m \rceil} - m \leq r \leq m - 1$ kódujeme r ako $r + 2^{\lceil \log m \rceil} - m$ binárne $\lceil \log m \rceil$ bitmi
- (*truncated binary code*)

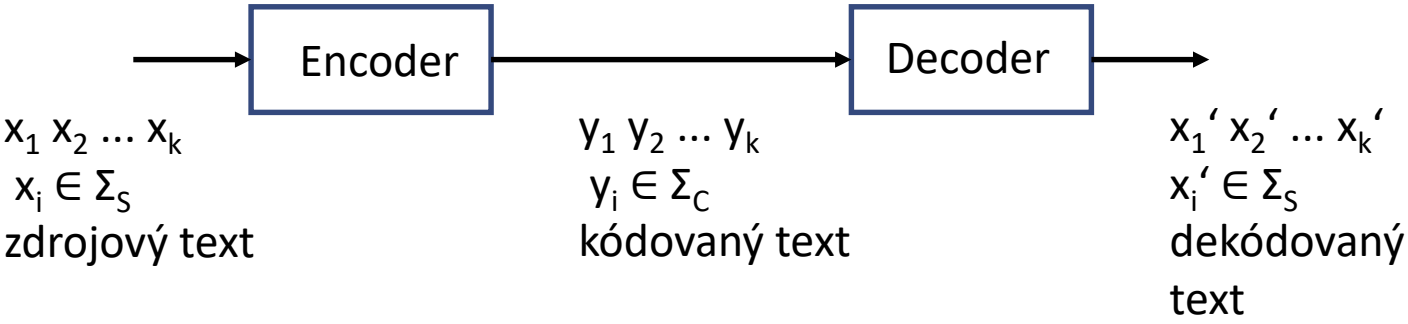
Rice-Golombove kódy - príklad

symbol	m = 1	m = 2	m = 3	m = 4
0	0	0 0	0 0	0 00
1	10	0 1	0 10	0 01
2	110	10 0	0 11	0 10
3	1110	10 1	10 0	0 11
4	11110	110 0	10 10	10 00
5	111110	110 1	10 11	10 01
6	1111110	1110 0	110 0	10 10
…	…	…	…	…

optimálny pre rozdelenie $p(i) = q^i \cdot (1 - q) \quad 0 < q < 1$

keď zvolíme $m = \left\lceil -\frac{1}{\log q} \right\rceil$

Kódovanie zdroja údajov (*Source Coding*)



Σ_S – abeceda zdroja, Σ_C – kódová abeceda

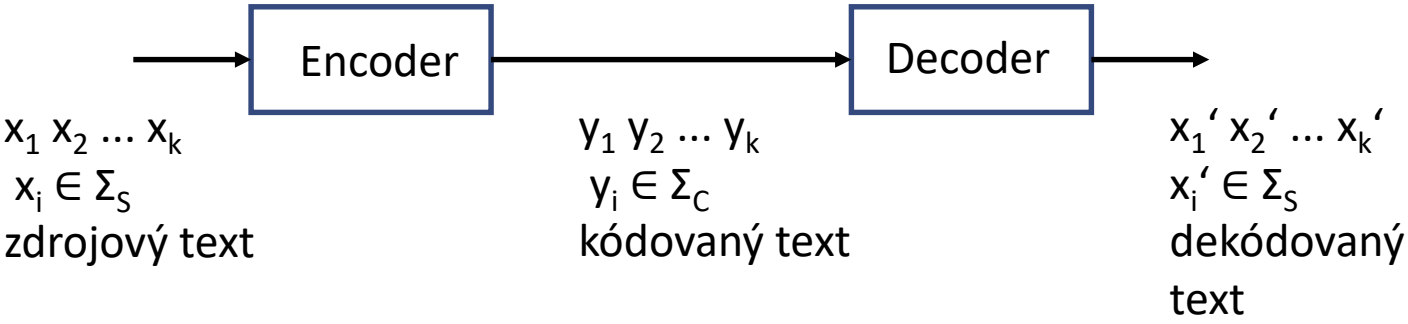
funkcie pre kódovanie a dekódovanie znakov e: $\Sigma_S \rightarrow \Sigma_C$,
d: $\Sigma_C \rightarrow \Sigma_S$ rozšírime na reťazce správ $S \subseteq \Sigma_S^*$ a kódov $C \subseteq \Sigma_C^+$

$E(x_1\ x_2 \ldots x_k) = e(x_1)\ e(x_2) \ldots e(x_k) = y_1\ y_2 \ldots y_k$

$D(y_1\ y_2 \ldots y_k) = d(y_1)\ d(y_2) \ldots d(y_k) = x_1'\ x_2' \ldots x_k' = d(e(x_1))\ d(e(x_2)) \ldots d(e(x_k)) = D(e(x_1)\ e(x_1) \ldots e(x_k)) = D(E(x_1\ x_2 \ldots x_k))$

aby platilo aj $x_1'\ x_2' \ldots x_k' = D(E(x_1\ x_2 \ldots x_k)) = x_1\ x_2 \ldots x_k$

Jednoznačná dekódovateľnosť



$D(y_1\ y_2 \ldots y_k) = d(y_1)\ d(y_2) \ldots d(y_k) = x_1'\ x_2' \ldots x_k'$

ako rozdeliť kódovaný text $y_1\ y_2 \ldots y_n$?

C – množina kódových slov je jednoznačne dekódovateľná (je kódom) – ak neexistuje dvojica textov $y_1\ y_2 \ldots y_k = y_1'\ y_2' \ldots y_r'$

- platí ak kódové slová $y \in C$ sú rovnakej dĺžky – blokový kód
- pre kódy nerovnakej dĺžky (VLC – *variable-length code*) musíme jednoznačnú dekódovateľnosť zabezpečiť inak

Test jednoznačnosti rozdelenia kódu

začínam so zoznamom reťazcov všetkých kódových slov C

- ak existuje v zozname dvojica reťazcov, kde je jeden prefixom druhého, do zoznamu pridám reťazec, ktorý vznikne z druhého reťazca oddelením prefixu (prvého reťazca) – ak tam ešte taký neexistuje
- ak by som mal pridať kódové slovo – kódová množina nie je jednoznačne rozdeliteľná
- ak už neviem nič pridať – kód je jednoznačne dekódovateľný

Sardinas-Patterson algoritmus (polynomiálna zložitosť)

pre množiny reťazcov M, N je množina všetkých reťazcov, ktoré dostaneme z M oddelením prefixu z množiny N jej ľavým kvocientom ... $N^{-1}M = \{ y \mid xy \in M \ \& \ x \in N \}$

- $A_1 = C^{-1}C \setminus \{\varepsilon\}$ - ľavý kvocient C okrem prázdneho slova
- $A_{i+1} = C^{-1}A_i \cup A_i^{-1}C$ - induktívne
- ak A_i obsahuje nejaké slovo z C (resp. prázdne slovo), kódová množina nie je jednonačne rozdeliteľná
- ak $A_i = A_j$ pre $j < i$ – algoritmus je v slučke – kód je jednoznačne dekódovateľný

Binárne VLC kódy a ich dĺžka

$\Sigma_C = \{0,1\}$, $C \subseteq \{0,1\}^*$ – sekvencie 0 a 1

Kraft – McMillan nerovnosť

- C obsahuje n kódových slov s dĺžkami $l_1, l_2, \ldots, l_n$
- ak C je jednoznačne dekódovateľná, potom platí

$$K(C) = \sum_{i=1}^{n} 2^{-l_i} \leq 1$$

- $\left(K(C)\right)^N = \left(\sum_{i=1}^{n} 2^{-l_i}\right)^N = \left(\sum_{i_1=1}^{n} 2^{-l_{i_1}}\right)\left(\sum_{i_2=1}^{n} 2^{-l_{i_2}}\right) \ldots \left(\sum_{i_N=1}^{n} 2^{-l_{i_N}}\right) =$

$$= \sum_{i_1=1}^{n} \sum_{i_2=1}^{n} \ldots \sum_{i_N=1}^{n} 2^{-(l_{i_1}+l_{i_2}+\cdots+l_{i_N})}$$

$l_{i_1} + l_{i_2} + \cdots + l_{i_N}$ sú dĺžky N-tíc kódových slov - aspoň N a najviac Nl, kde $l = max\{l_1, l_2, \ldots l_n\}$; potom $\left(K(C)\right)^N = \sum_{k=N}^{Nl} A_k 2^{-k} \leq \sum_{k=N}^{Nl} 2^k 2^{-k} = Nl - N + 1$

kde A_k je počet kombinácií N kódových slov s celkovou dĺžkou k, čo je najviac 2^k (vzhľadom na jednoznačnú dekódovateľnosť)

Pre $K(C) > 1$ by však ľavá strana nerovnice rástla exponenciálne – čo je spor

Realizácia binárnymi prefixovými kódmi

Pre zadané dĺžky $l_1, l_2, \ldots, l_n$ pre ktoré platí Kraftova nerovnosť $\sum_{i=1}^{n} 2^{-l_i} \leq 1$ vieme vždy nájsť prefixový kód s kódovými slovami dĺžky $l_1, l_2, \ldots, l_n$

- prefixový kód $\forall\, x, y \in C : x$ nie je prefixom y
  (ak $\exists z \in \Sigma_C{}^*$ také, že $xz = y$, potom $z = \varepsilon$)

- binárne kódové stromy
- konštrukcia kódu (Shannon)

Kód zapíšeme do binárneho kódového stromu výšky $l = \max\{l_1, l_2, \ldots, l_n\}$ s 2^l koncovými listami. Postupujeme od najkratšieho k najdlhšiemu kódu. Kódové slovo dĺžky l_i zapíšeme na voľné miesto do i-teho riadku, kde kvôli prefixovej požiadavke ubudne 2^{l-l_i} koncových listov.
Celkom ubudne $\sum_{i=1}^{n} 2^{l-l_i} = 2^l . \sum_{i=1}^{n} 2^{-l_i} \leq 2^l$ (podľa Kraftovej nerovnosti) koncových listov t. j. vystačí na všetky kódové slová.

Štatistická (entropická) kompresia

Binárne kódovanie pre známe rozdelenie pravdepodobnosti výskytov kódových slov s najlepším kompresným pomerom

$p_1, p_2, \ldots, p_n$ ... pravdepodobnosti výskytov znakov abecedy $x_1, x_2, \ldots, x_n$ zdroja Σ_S ($\sum_{i=1}^{n} p_i = 1$)

$l_1, l_2, \ldots, l_n$ ... dĺžky kódových slov (v bitoch)

stredná (priemerná) dĺžka kódového slova $L = \sum_{i=1}^{n} p_i l_i \quad [bit/znak]$

(niekedy tiež informačná hustota – Rate)

hľadáme jednoznačne dekódovateľný kód s takými dĺžkami $l_1, l_2, \ldots, l_n$ aby L bolo minimálne

(znaky s veľkým výskytom budeme kódovať kratšími kódmi)

Huffmanovo kódovanie

rekurzívna konštrukcia binárneho prefixového kódu
s minimálnou strednou dĺžkou kódového slova:

ak $n = 2$ máme len 2 zdrojové symboly, ktoré zakódujeme kódovými slovami 0 a 1

ak $n > 2$ nech x_i a x_j sú zdrojové symboly s najmenšími pravdepodobnosťami p_i a p_j

- spojíme ich do nového symbolu x_{ij} zdrojovej abecedy s pravdepodobnosťou výskytu $p_i + p_j$ a vynecháme symboly x_i a x_j
- pre novú $n - 1$ prvkovú abecedu vytvoríme Huffmanov kód, v ktorom je symbolu x_{ij} priradené kódové slovo w
- pre pôvodné symboly x_i a x_j priradíme kódové slová $w0$ a $w1$

Huffmanovo kódovanie - príklad

pravdep.	0,4	0,2	0,15	0,1	0,1	0,05
symbol	a	b	c	d	e	f
	0,4	0,2	0,15	0,15	0,1	
	a	b	(ef)	c	d	
	0,4	0,25	0,2	0,15		
	a	(cd)	b	(ef)		
	0,4	0,35	0,25			
	a	(b(ef))	(cd)			
	0,6	0,4				
	((b(ef))(cd))	a				

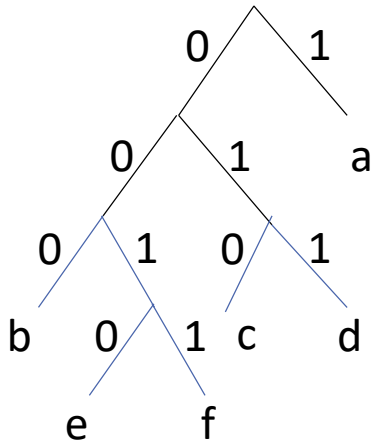
(((b(ef))(cd))a) - výsledok = binárny strom

Huffmanovo kódovanie – binárny strom

(((b(ef))(cd))a)

binárny strom v zátvorkovom zápise
(po ľavej zátvorke – vnorenie o jednu úroveň,
po pravej zátvorke – vynorenie,
po zápise ľavého vrcholu – krok vpravo)

kanonický (jednoznačný) zápis –
vľavo kódujeme 0 a vpravo 1



a	1
b	000
c	010
d	011
e	0010
f	0011

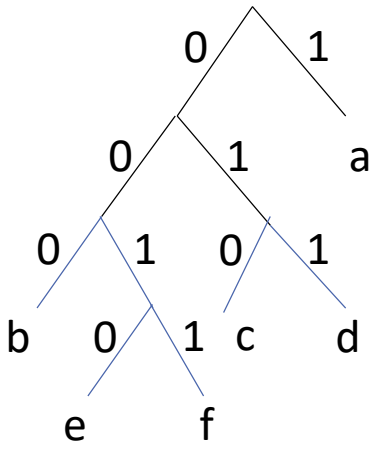
stredná dĺžka kódového slova

$$L = 0{,}4 \cdot 1 + 0{,}2 \cdot 3 + 0{,}15 \cdot 3 + 0{,}1 \cdot 3 + \\ + 0{,}1 \cdot 4 + 0{,}05 \cdot 4 = 2{,}35$$

Huffmanovo kódovanie - príklad

konštrukcia kódu s priebežnou informáciou o pridávaných bitoch od jeho konca

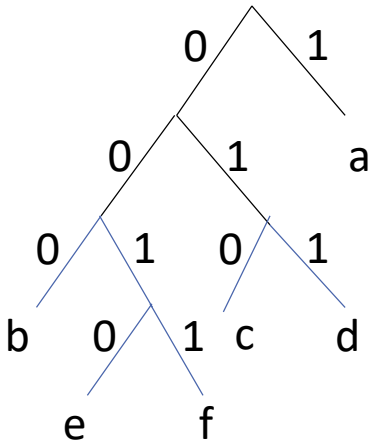
pravdep.	0,4	0,2	0,15	0,1	0,1	0,05
symbol	a	b	c	d	e	f
	0,4	0,2	0,15	0,15	0,1	
	a	b	0e 1f	c	d	
	0,4	0,25	0,2	0,15		
	a	0c 1d	b	0e 1f		
	0,4	0,35	0,25			
	a	0b 10e 11f	0c 1d			
	0,6	0,4				
	00b 010e 011f 10c 11d	a				



výsledný slovník 000b 0010e 0011f 010c 011d 1a

Huffmanovo kódovanie – dekódovanie

kódovaciu tabuľku musí dostať aj príjemca – pre rýchlu implementáciu pomôže reprezentácia vo forme binárneho stromu



a	1
b	000
c	010
d	011
e	0010
f	0011

00010101001100100111
000 1 010 1 0011 0010 011 1
b a c a f e d a
20 bitov/8 znakov

pre ASCII vstup – kompresný pomer $= \frac{20}{8*8} = 31{,}25\%$

pre vstup n znakov s uvažovanou distribúciou pravdepodobnosti bude kompresný pomer $\approx \frac{n*L}{n*8} = \frac{2{,}35}{8} = 29{,}375\%$

Optimálnosť Huffmanovho kódovania

stredná dĺžka kódového slova binárneho HC je najmenšia zo všetkých jednoznačne dekódovateľných binárnych kódov

ak $n = 2$

zdrojové symboly zakódujeme kódovými slovami 0 a 1, žiaden iný jednoznačne dekódovateľný kód nemá kratšie slová

ak sú všetky HC kódy s $n - 1$ vstupnými znakmi optimálne,

nech n-znakový HC C_H vznikol z optimálneho n-1 znakového HC C'_H výmenou kódového slova w za dvojicu w0 a w1. Kód C_H bude zrejme prefixový.

v optimálnom kóde musí mať znak s najdlhším kódovým slovom v binárnom strome suseda – súrodenca (sibling) (inak by som toto kódové slovo mohol skrátiť)

nech x_i a x_j sú zdrojové symboly s najmenšími pravdepodobnosťami p_i a p_j , potom existuje optimálny kód C_O v ktorom sa ich kódové slová líšia len v poslednom bite (v binárnom strome sú súrodencami (siblings)) (môžeme ich bez zmeny strednej dĺžky kódového slova vymeniť za dva symboly s najdlhšími kódovými slovami z predchádzajúceho odstavca)

Optimálnosť Huffmanovho kódovania

ak sú všetky HC kódy s $n-1$ vstupnými znakmi optimálne,

nech n-znakový HC C_H vznikol z optimálneho n-1 znakového HC C'_H výmenou kódového slova w za dvojicu w0 a w1. Kód C_H bude zrejme prefixový.

v optimálnom kóde musí mať znak s najdlhším kódovým slovom v binárnom strome suseda – súrodenca (*sibling*) (inak by som toto kódové slovo mohol skrátiť)

nech x_i a x_j sú zdrojové symboly s najmenšími pravdepodobnosťami p_i a p_j , potom existuje optimálny kód C_O v ktorom sa ich kódové slová líšia len v poslednom bite (v binárnom strome sú súrodencami), môžeme ich bez zmeny strednej dĺžky kódového slova vymeniť za dva symboly s najdlhšími kódovými slovami z predchádzajúceho odstavca)

znaky x_i a x_j s kódovými slovami $u0$ a $u1$ nahradíme jedným znakom x_{ij} s kódovým slovom u a pravdepodobnosťou $p_i + p_j$, ostatné kódové slová necháme podľa C_O. Potom stredná dĺžka kódového slova v novom kóde C'_O (s n-1 znakmi) bude
$L'_O = L_O - p_i l - p_j l + (p_i + p_j)(l-1) = L_O - p_i - p_j$ kde l je dĺžka kódových slov $u0$ a $u1$

pre n-1 znakový optimálny HC C'_H teda bude $L'_H \leq L'_O = L_O - p_i - p_j$ a pretože C_H vznikol z C'_H rozdelením kódového slova w na w0 a w1 pre znaky x_i a x_j, platí
$L_H = L'_H + p_i + p_j \leq L_O - p_i - p_j + p_i + p_j = L_O$

Entropia

zdroj S s pravdepodobnosťou výskytov symbolov abecedy zdroja
$p_1, p_2, \ldots, p_n \quad (p_i > 0)$

miera výnimočnosti (informácia) i-teho symbolu zdroja
$I(x_i) = -\log(p_i) = \log(1/p_i)$

stredná (priemerná) miera výnimočnosti - entropia zdroja

$$H(S) = -\sum_{i=1}^{n} p_i \cdot \log(p_i)$$

miera neusporiadanosti (Shannon 1948)

pre rovnomernú pravdepodobnosť je $-\sum_{i=1}^{n} \frac{1}{n} \log(\frac{1}{n}) = \log n$
(a vždy platí $0 \leq H \leq \log n$)

základ logaritmu určujú jednotky miery informácie
(2 = bity, e = nats, 10 = hartleys)

$\log_2 n$ binárna entropia (počet bitov informácie)

Entropia a vzťah k Huffmanovmu kódovaniu

pre zdroj S s pravdepodobnosťou výskytov znakov abecedy zdroja $p_1, p_2, \ldots, p_n$ a jednoznačne dekódovateľným kódovaním s dĺžkami kódových slov $l_1, l_2, \ldots, l_n$ so strednou dĺžkou L platí $H(S) \leq L$

$$H(S) - L = -\sum_{i=1}^{n} p_i \log(p_i) - \sum_{i=1}^{n} p_i . l_i =$$

$$\sum_{i=1}^{n} p_i \left(-\log(p_i) - \log\left(2^{l_i}\right)\right) = \sum_{i=1}^{n} p_i \log\left(\frac{2^{-l_i}}{p_i}\right) \leq$$

$$\frac{1}{\ln 2} \sum_{i=1}^{n} p_i \left(\frac{2^{-l_i}}{p_i} - 1\right) = \frac{1}{\ln 2}\left(\sum_{i=1}^{n} 2^{-l_i} - 1\right) = \frac{1}{\ln 2}(K - 1) \leq 0$$

a rovnosť nastane práve keď $\forall i \;\; l_i = -\log(p_i)$

dĺžky kódových slov však musia byť celočíselné, ale pre $l_i = \lceil \log(1/p_i) \rceil$ platí

$$H(S) \leq L \leq H(S) + 1$$

Huffmanovo kódovanie - príklad

pre príklad na predchádzajúcich slajdoch

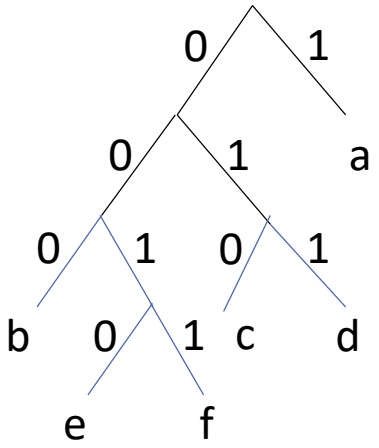
pravdepodobnosť	0,4	0,2	0,15	0,1	0,1	0,05
symbol	a	b	c	d	e	f

výsledný slovník 000b 0010e 0011f 010c 011d 1a

a	1
b	000
c	010
d	011
e	0010
f	0011

stredná dĺžka kódového slova

$$L = 0{,}4 \cdot 1 + 0{,}2 \cdot 3 + 0{,}15 \cdot 3 + + 0{,}1 \cdot 3 + 0{,}1 \cdot 4 + 0{,}05 \cdot 4 = 2{,}35$$



entropia

$$H = 0{,}4 \cdot 1{,}322 + 0{,}2 \cdot 2{,}322 + + 0{,}15 \cdot 2{,}737 + 0{,}1 \cdot 3{,}322 + + 0{,}1 \cdot 3{,}322 + 0{,}05 \cdot 4{,}322 = 2{,}284$$

Rozšírený Huffmanov kód

pre rozdelenie $(0.8, 0.18, 0.02)$ je HC (0,10,11)
$H = 0.816 < 1.2 = L$ je redundancia kódu 0,384 32%

pokiaľ budeme kódovať dvojice vstupných znakov, bude abeceda veľkosti 9 s $L = \frac{1.7228}{2} = 0.8614$ a redundancia len 5%

pre zdroj $S^{(m)}$s m-znakovými vstupnými blokmi je
$H(S^{(m)}) \leq L^{(m)} \leq H(S^{(m)}) + 1$

pričom $H(S^{(m)}) = m\,H(S)$ a $L^{(m)} = m\,L$ a teda

$$H(S) = \frac{H(S^{(m)})}{m} \leq \frac{L^{(m)}}{m} = L \leq \frac{H(S^{(m)}) + 1}{m} \leq H(S) + \frac{1}{m}$$

Source Coding Theorem (Shannon) – N znakov zdroja s entropiou H nie je možné (bez straty informácie) komprimovať na menej ako N.H bitov

Ternárne Huffmanovo kódovanie

kódová abeceda $\Sigma_C = \{0,1,2\}$, $C \subseteq \{0,1,2\}^*$ – sekvencie 0, 1 a 2
spájame tri symboly s najmenšou pravdepodobnosťou do nového symbolu

pravdep.	0,4	0,2	0,15	0,1	0,1	0,05
symbol	a	b	c	d	e	f
	0,4	0,25	0,2	0,15		
	a	0d 1e 2f	b	c		
	0,6	0,4				
	00d 01e 02f 1b 2c	a				

stredná dĺžka kódového slova

výsledný slovník: 000d 001e 002f 01b 02c 1a

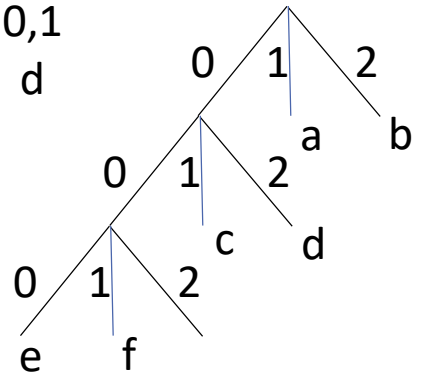
$$L = 0{,}4 \cdot 1 + 0{,}2 \cdot 2 + 0{,}15 \cdot 2 + 0{,}1 \cdot 3 + 0{,}1 \cdot 3 + 0{,}05 \cdot 3 = 1{,}85$$

v poslednom kroku sme spojili len 2 symboly – výsledok nebude optimálny!

Ternárne Huffmanovo kódovanie

aby sme v každom kroku mohli spojiť 3 symboly do jedného, musíme štartovať s (3-1).k +1 symbolmi - pridáme symboly s nulovou pravdepod.

pravdep.	0,4	0,2	0,15	0,1	0,1	0,05	0
symbol	a	b	c	d	e	f	*dummy*
	0,4	0,2	0,15	0,15	0,1		
	a	b	0e 1f	c	d		
	0,4	0,4	0,2				
	00e 01f 1c 2d	a	b				



výsledný slovník 000e 001f 01c 02d 1a 2b

stredná dĺžka kódového slova

$L = 0{,}4 \cdot 1 + 0{,}2 \cdot 1 + 0{,}15 \cdot 2 + + 0{,}1 \cdot 2 + 0{,}1 \cdot 3 + 0{,}05 \cdot 3 = 1{,}55$ „tritov“

$H_3 = \log_3 2 * H_2 = 1{,}441$ entropia v „tritoch“

Tunstallovo kódovanie (blokové entropické)

- 2^k kódových slov dĺžky k bitov
- kódujú sa vstupné reťazce rôznej dĺžky podľa frekvencie výskytu v zdrojovom texte (pre menšie abecedy)
- začína sa s $n = |\Sigma_S|$ kódmi pre znaky abecedy
- postupne opakujeme odstránenie reťazca s najväčšou pravdepodobnosťou výskytu a jeho nahradenie reťazcami, rozšírenými o jeden znak abecedy, pričom sa prepočíta ich pravdepodobnosť výskytu
- po m opakovaniach $n + m(n-1) \leq 2^k$ kódové slová sa uložia do n-árnych stromov (trie)
- rate ~ k/d [bit/symbol] ak d je priemerná výška stromu

Adaptívne Huffmanovo kódovanie

- reagovať na zmeny frekvencie výskytov vstupných znakov zmenami kódovacieho slovníka
- je možné riešiť konštrukciou HC po každom znaku ...
- začne sa rovnomerným kódom a rekonštrukcia podľa aktuálnej frekvencie po zvláštnom znaku ...
- fixný (rovnomerný) kód a priradenie podľa frekvencie výskytov (preusporiadanie priradenia)

Adaptívne Huffmanovo kódovanie

FGK (Faller, Gallager, Knuth)

- zápis binárneho stromu v kanonickom (predpísanom) tvare – jednozačne určí kódový slovník
- váha listov (priebežná početnosť výskytov)
- váha vnútorného vrcholu = súčet váh potomkov
- súrodenecká vlastnosť SP (*sibling property*) – vrcholy je možné jednoznačne očíslovať (od 1 do 2n-1) tak, že

  (a) všetky dvojice 2j-1 a 2j (j ∈ {1..n-1}) majú spoločného rodiča – sú súrodencami (okrem koreňa s číslom 2n-1)

  (b) pre váhy pri tomto usporiadaní platí $w_1 \leq w_2 \leq \ \ldots \leq w_{2n-1}$
- binárny prefixový kód je Huffmanov práve vtedy, keď má odpovedajúci binárny strom súrodeneckú vlastnosť

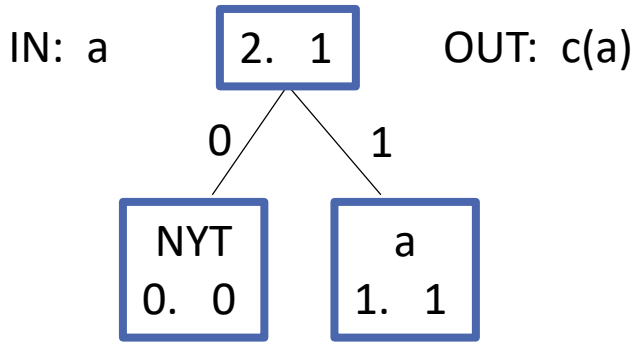
FGK algoritmus

vytvárame binárny strom tak, aby platila súrodenecká vlastnosť

začíname špeciálnym kódom pre znak, ktorý ešte nebol odoslaný NYT (not yet transmitted) s váhou 0

ak kódujeme znak prvýkrát

(a) odošleme aktuálny kód pre NYT a jednoznačný kód znaku (napr. jeho poradie v abecede s blokovým kódom na $\lceil \log n \rceil$ bitov)



(b) vrchol NYT rozvetvíme na dva listy, vľavo ostane pôvodný NYT a vpravo pridáme list pre nový znak s číslom 1 a váhou 1

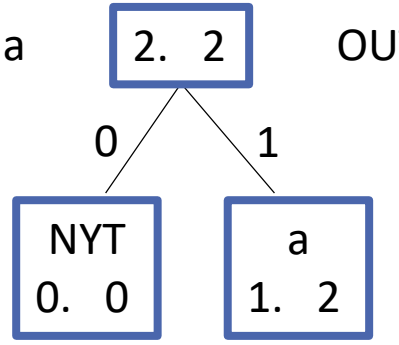
(c) zvýšime ostatné poradové čísla o 2 (pridali sme 2 vrcholy)

(d) aktualizujeme zvyšok stromu od rodiča nového znaku (od vrchola 2 s váhou 0) tak, aby ostala platiť SP

FGK príklad

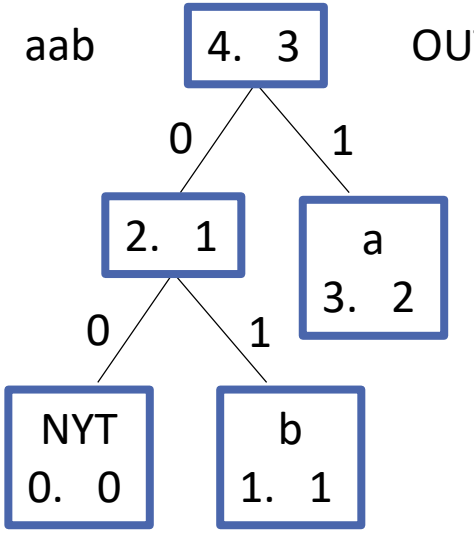
ak kódujeme znak, ktorý už je v strome na pozícii x
(a) odošleme jeho aktuálne kódové slovo
(b) aktualizujeme strom
od vrcholu x tak, aby platila SP
(tu len zvýšime jeho váhu
a váhu jeho rodiča)

IN: aa



OUT: c(a)1

IN: aab



OUT: c(a)10c(b)

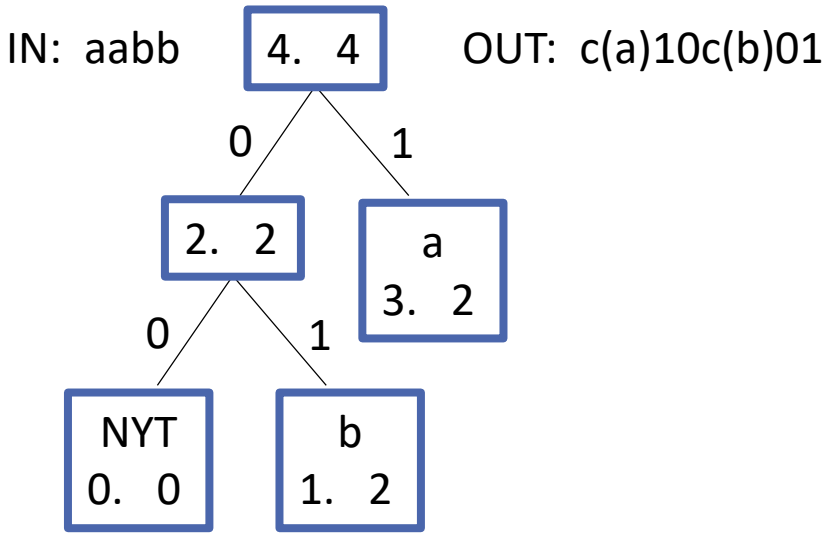
pre ďalší nový znak bude už výstupný kód 0 pre NYT a jeho id kód c(b)

zvýšime ostatné poradové čísla a aktualizujeme strom od vrchola 2., aby spĺňal SP (zvýšime váhu 2. a 4.)

FGK príklad

pre ďalší znak, ktorý nájdeme v strome

(a) odošleme jeho
aktuálne kódové slovo
(b) aktualizujeme strom
(a váhy až po koreň)
od vrcholu znaku tak,
aby platila SP
(teraz len zvýšime váhu
vrcholu 1. 2. a 4.
- váhy ostanú v poradí vzostupne
- susedné dvojice sú v poradí pri sebe)

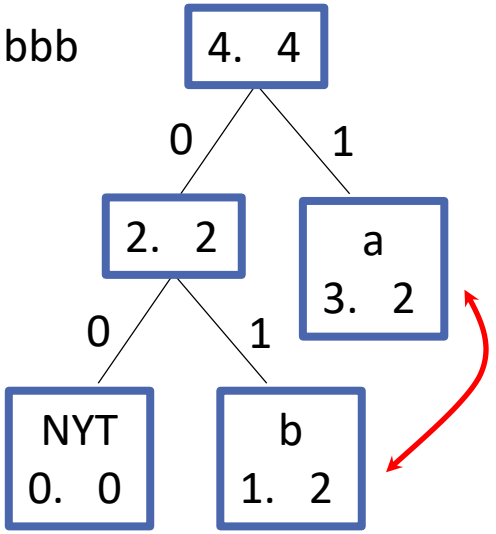


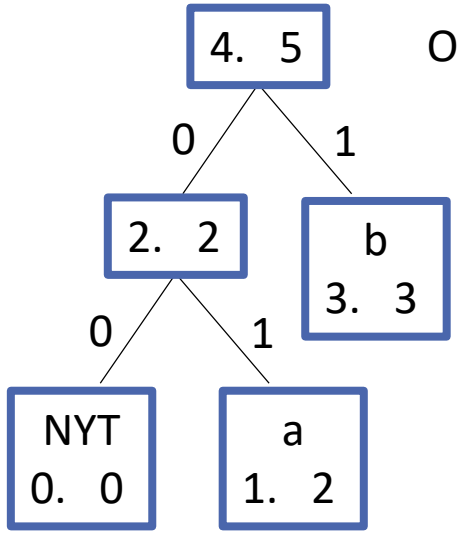
FGK príklad

pri opätovnom výskyte b by teraz aktualizáciou váhy vrchola b došlo k porušeniu poradia váh, preto nájdeme v poradí najvyšší vrchol (okrem rodiča) s touto váhou a vymeníme pozície týchto dvoch vrcholov v strome (aj ich poradové čísla)

potom už môžeme zvyšovať váhy rodičov až ku koreňu (v tomto prípade neporušia SP)

IN: aabbb

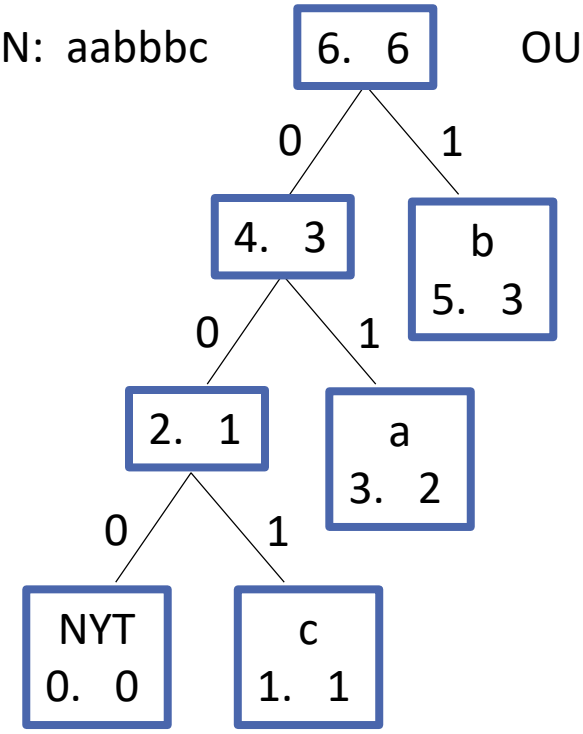




OUT: c(a)10c(b)0101

FGK príklad

IN: aabbbc



OUT: c(a)10c(b)010100c(c)

nový znak c na vstupe znamená

(a) odoslať kód pre NYT a poradový kód znaku c

(b) zmenu NVT vrchola na vnútorný, jeho rozvetvenie na ľavý list NVT (s poradovým číslom 0) a pravý list pre znak c (s číslom 1)

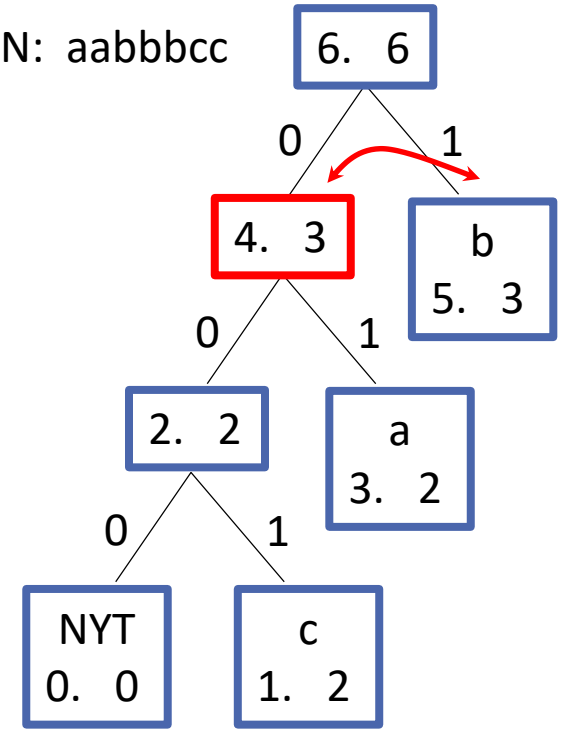
(c) posunutie číslovania vrcholov

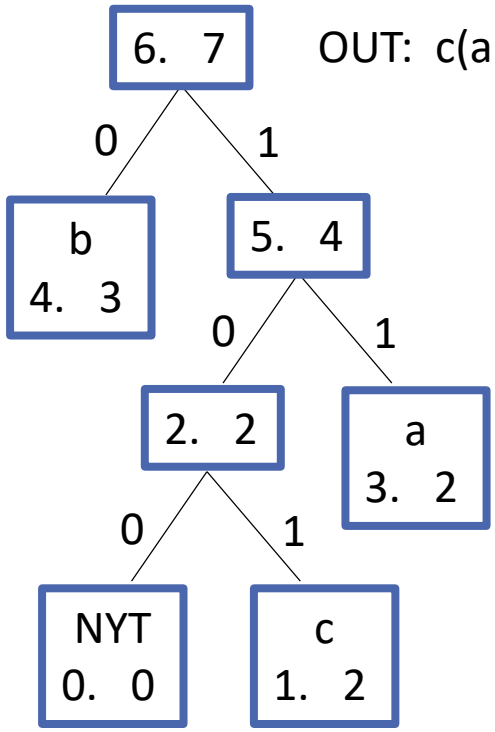
(d) aktualizáciu váh vrcholov od rodiča nového znaku a prípadne zmenu poradia, aby ostala platiť SP (teraz meniť nemusíme)

FGK príklad

ďalší výskyt znaku c na vstupe znamená kód 001 na výstup a úpravu stromu – zvýšime váhu vrcholu 1. a vrcholu 2. Rodič vrcholu 2. (4.) nie je v postupnosti najvyšší s touto váhou, musíme ho vymeniť (aj s podstromami) :

IN: aabbbcc





OUT: c(a)10c(b)010100c(c)001

FGK algoritmus

začíname s vrcholom NYT s váhou 0 a pre každý znak vstupu:

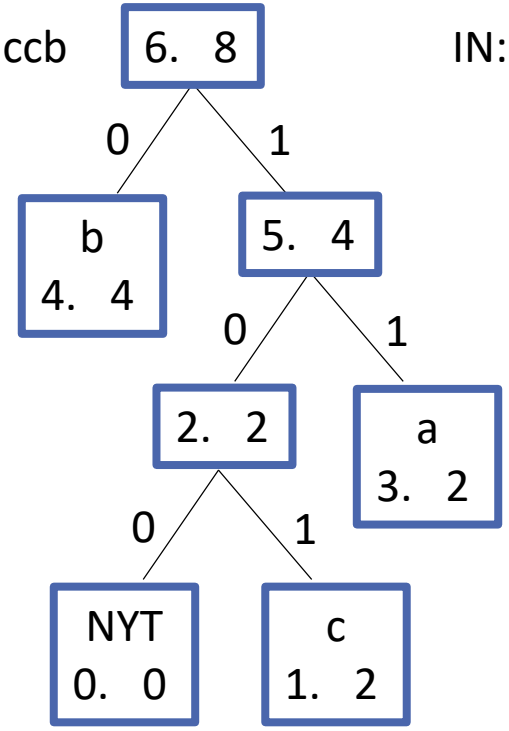
* ak je to nový znak, výstup je kód znaku NYT a poradový kód nového znaku, vrchol NYT sa stane vnútorný, pridáme rozvetvenie na NVT list vľavo (číslo 0, váha 0) a list nového znaku (číslo 1, váha 1) vpravo, číslovanie ostatných vrcholov zväčšíme o 2 a aktualizujeme od vrchola x = 2;
* inak ak znak je v strome na mieste x, výstup je aktuálny kód
  ak susedom listu x je list NYT a v poradí vrcholov existujú listy(!) stromu s rovnakou váhou, vymeníme list x (aj číslovanie) s najvyšším takým listom a ten označíme x
* pokiaľ x nie je koreňom aktualizujeme takto:
  ak po x existujú v poradí vrcholy s rovnakou váhou, vrchol x (a jeho očíslovanie) vymeníme s najvyšším vrcholom s rovnakou váhou (v podstromoch očíslovanie nemeníme) a označíme x nové číslo vrcholu
  zvýšime váhu vrcholu x a ako x označíme jeho rodiča

FGK príklad - pokračovanie

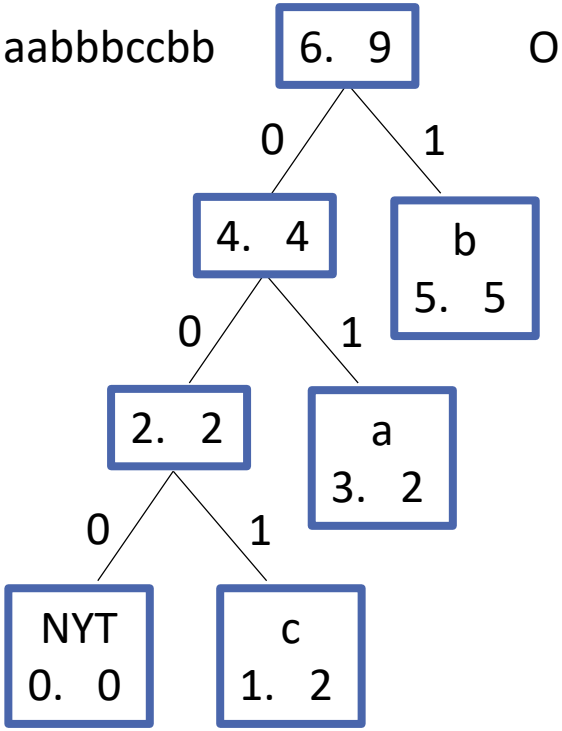
pokračovanie príkladu :

znaky b b vymenia podstromy späť

IN: aabbbccb

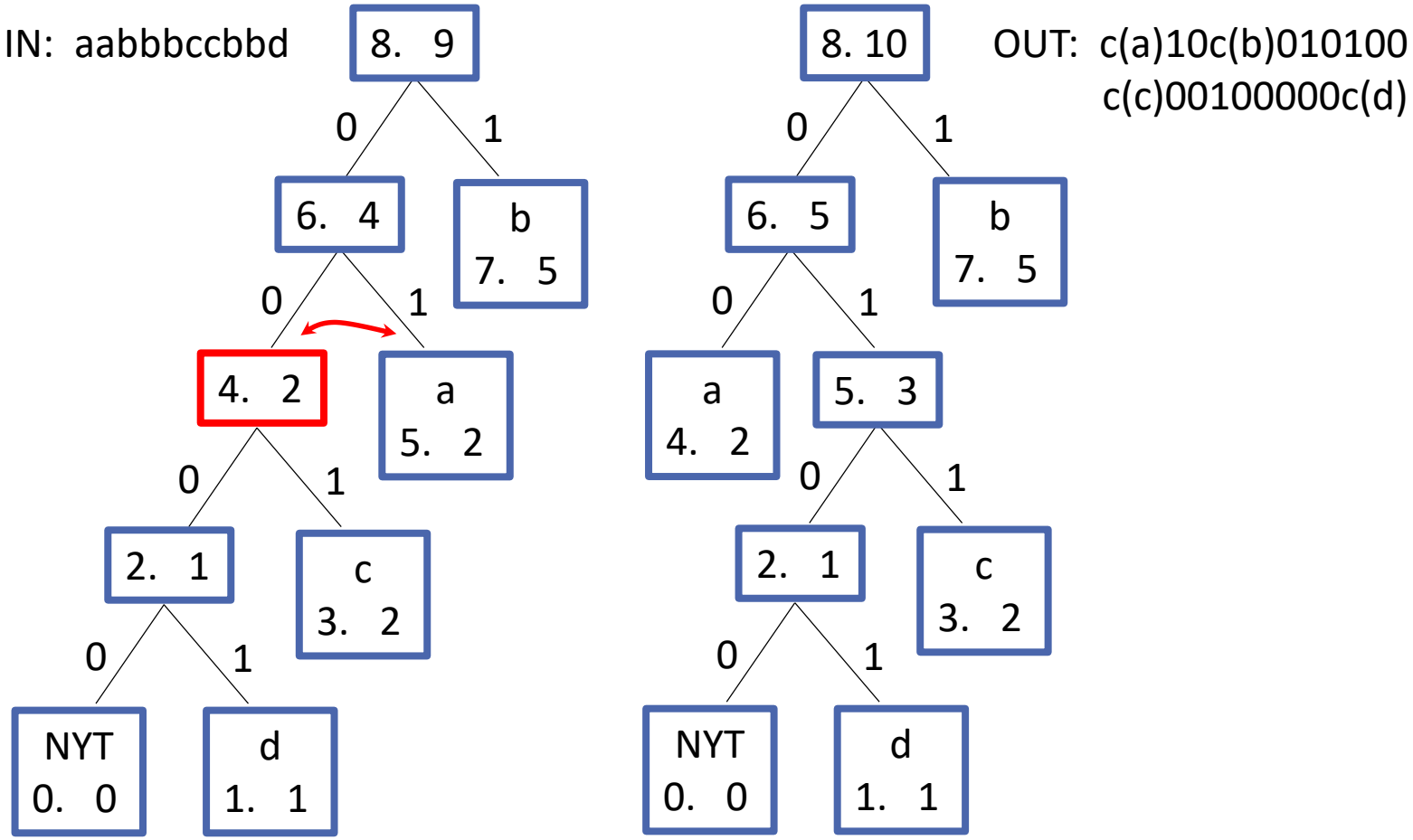


IN: aabbbccbb



OUT: c(a)10c(b)0101
00c(c)00100

FGK príklad - pokračovanie



FGK algoritmus – dekódovanie

dekódovanie tiež začíname s vrcholom NYT s váhou 0
opakujeme prechody stromom podľa kódu od koreňa k listom:

- ak list je NYT (na začiatku implicitne – prázdny kód)
  * dekódujeme nový znak podľa poradového kódu na vstupe
  * vrchol NYT sa stane vnútorný, pridáme jeho rozvetvenie na NVT list vľavo a list nového znaku vpravo
  * upravíme číslovanie ostatných vrcholov (o 2)
  * ďalej aktualizujeme od vrcholu x = 2
- inak dekódujeme znak v liste a označíme ho x
  * ak susedom listu x je list NYT a v poradí vrcholov existujú listy stromu s rovnakou váhou, najvyšší taký list v poradí vymeníme za x (aj s číslovaním) a označíme ho x
- ...

FGK algoritmus – dekódovanie (pokračovanie)

- ...
- pokiaľ x nie je koreňom aktualizujeme takto:
  * ak po x existujú v poradí vrcholy s rovnakou váhou,
    vymeníme vrchol x (a jeho očíslovanie) s najvyšším vrcholom s rovnakou váhou (spolu s podstromami, v podstromoch očíslovanie nemeníme)
    a označíme x nové číslo vrcholu
  * zvýšime váhu vrcholu x a ako x označíme jeho rodiča

poznámky:

- číslovanie možno začať číslom 2n+1 a pridávať stále dve nižšie čísla – nie je potrebné prečíslovávať
- možno začať so stromom s preddefinovanými váhami a všetkými vstupnými znakmi (odpadne práca s NYT vrcholmi)

Adaptívne Huffmanovo kódovanie - Vitter

Vitter (1985)

- implicitné usporiadanie vrcholov – podľa vrstiev od najspodnejšej zľava doprava
- nie je potrebné uchovávať poradie, len štruktúru stromu (po vrstvách od najnižšej úrovne po najvyššiu)
- aktualizovaný vrchol sa vymieňa s najvyšším vrcholom (v topologickom poradí) v bloku vrcholov rovnakej váhy a rovnakého typu (listy resp. vnútorné vrcholy)
- je efektívnejší (vyváženejší strom), možno využiť špeciálne programátorské techniky ...

- zabúdací faktor – ak váha koreňa prekročí určenú hodnotu, všetky váhy sa pomerne zmenšia ...

Obmedzenia Huffmanovho kódovania

- Pre Huffmanov kód platí
  $H(S) \leq L \leq H(S) + p_{max} + 0{,}086$ (Gallagher 1978)
  pri veľkých výkyvoch pravdepodobností dáva zlé výsledky
- pri veľkej pravdepodobnosti výskytu znaku – kód je vždy aspoň jeden bit

napr. znak zdroja s pravdepodobnosťou výskytu 0.999 nesie informáciu $\log(1/0{,}999) = 0{,}00144$ a 1000 takých znakov nesie informáciu 1000 * 0.0014 = 1.44 bitov, no pri HC dostaneme 1000 bitový kód

- v statickej verzii je potrebné prenášať veľké kódovacie tabuľky
-