

Algoritmy komprese dat

Adaptivní Huffmanův kód



Robert G. Gallager
Massachusetts Institute of Technology



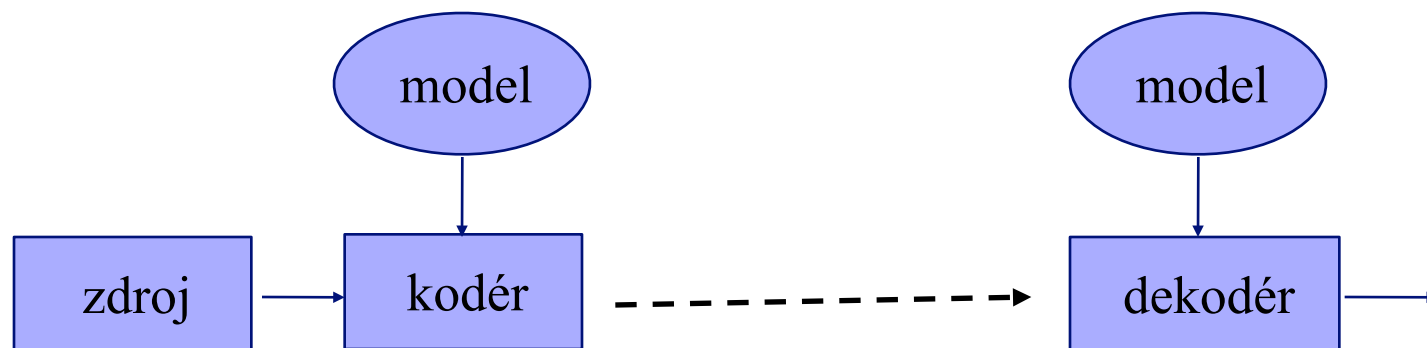
Donald Ervin Knuth
Stanford University

Statické × adaptivní metody

(Statistická) komprese dat

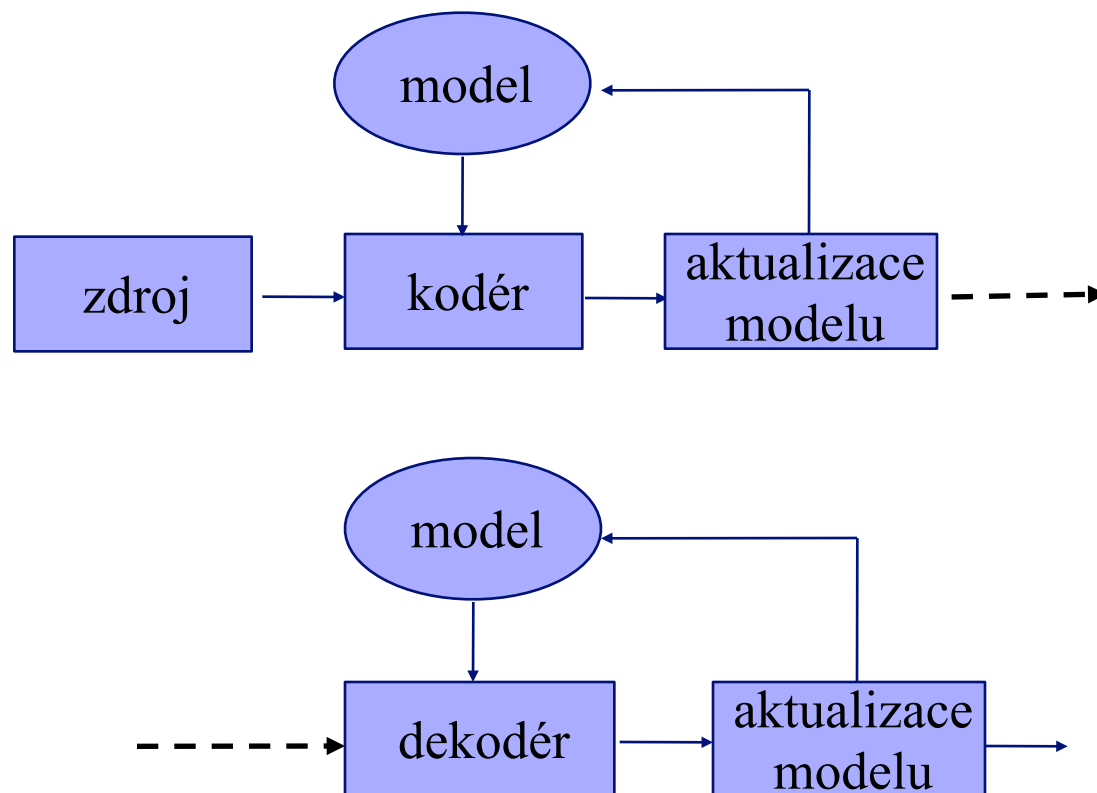
- modelování
- kódování

Statický model



Statické × adaptivní metody

adaptivní model



Adaptivní Huffmanův kód

Algoritmus FGK: Kódování

Inicializuj Huffmanův strom

```
while not EOF do read(znak)  
                    zakóduj znak  
                    aktualizuj strom
```

Algoritmus FGK: Dekódování

Inicializuj Huffmanův strom

```
while not EOF do dekóduj(znak)  
                    write(znak)  
                    aktualizuj strom
```

Adaptivní Huffmanův kód - hrubá síla

Rekonstrukce Huffmanova stromu

- po každé změně četností
- po načtení dalších k znaků
- po změně uspořádání znaků podle četností

Charakterizace Huffmanových stromů


Huffmanův strom - binární strom s nezáporně ohodnocenými vrcholy.

Dva vrcholy binárního stromu, které mají stejného rodiče, nazveme *sourozenci*.

Binární strom s nezáporně ohodnocenými vrcholy má *sourozeneckou vlastnost*, pokud

- \forall rodiče: $\text{ohodnocení}(\text{rodiče}) = \Sigma \text{ohodnocení}(\text{synů})$,
- každý vrchol kromě kořene má sourozence,
- vrcholy lze uspořádat do nerostoucí posloupnosti dle jejich ohodnocení tak, že sourozenci jsou vždy na sousedních místech

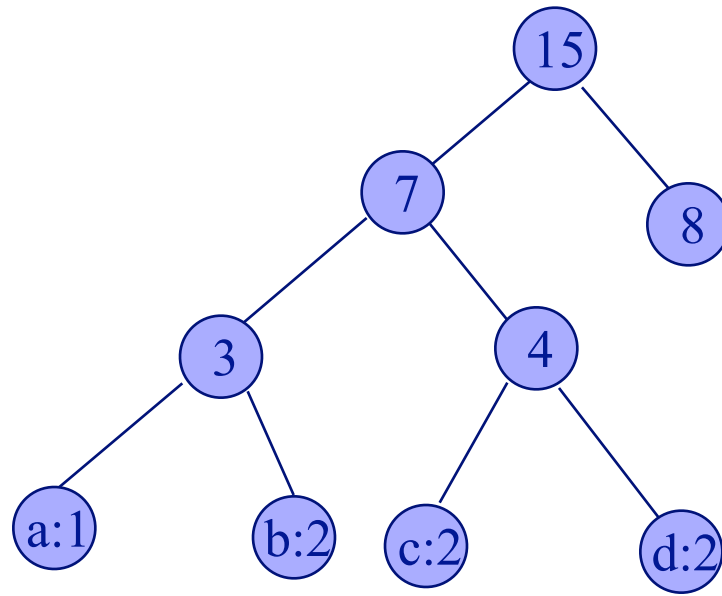
Charakterizace Huffmanových stromů

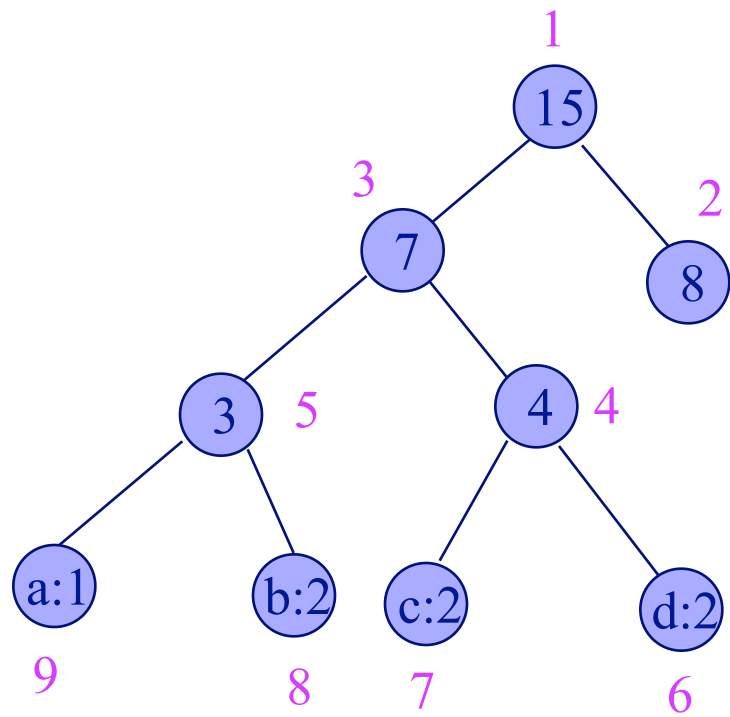
 **Věta** (Faller 1973, Gallagher 1978)

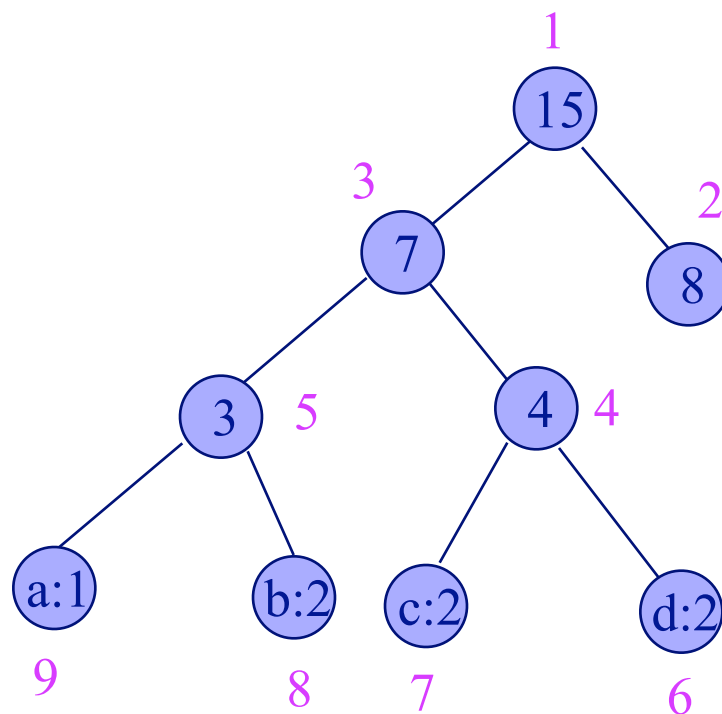
Binární strom s ohodnocenými vrcholy je Huffmanovým stromem právě tehdy, když má sourozeneckou vlastnost.

 Algoritmus FGK

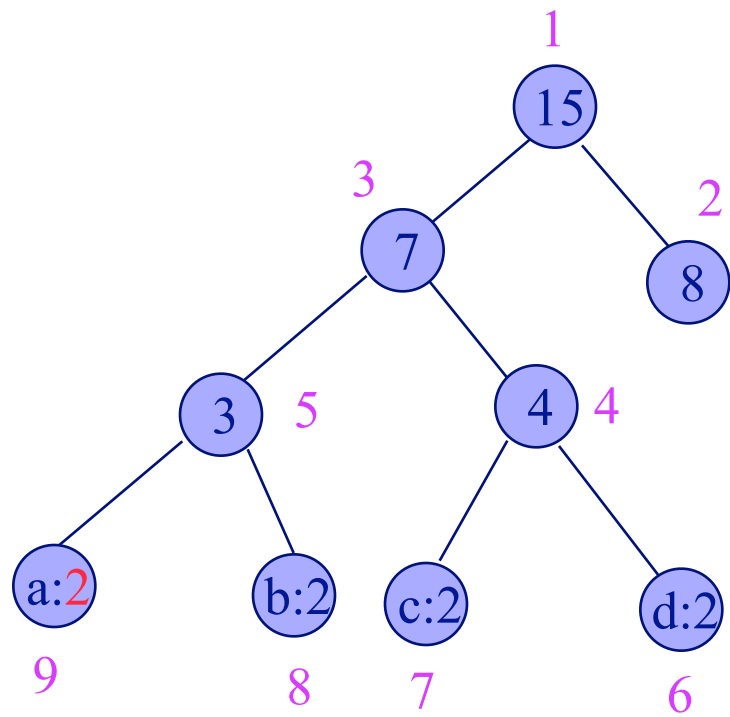
- Faller(1973), Gallagher(1978), Knuth(1985)

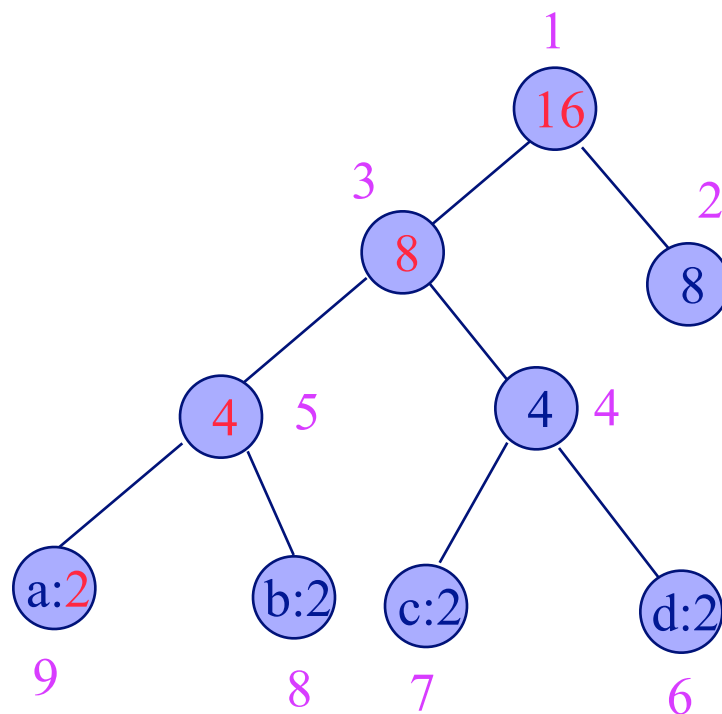




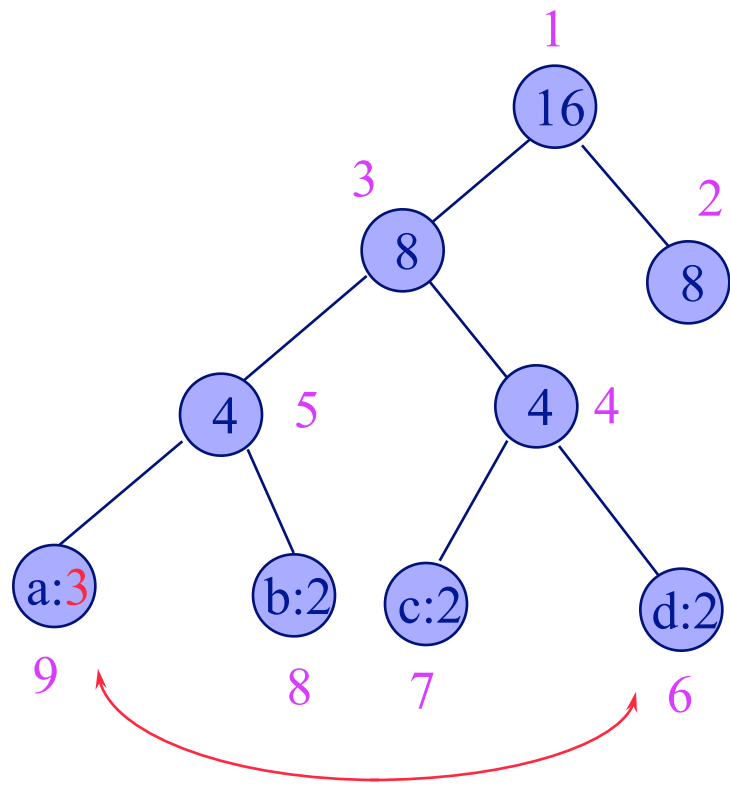


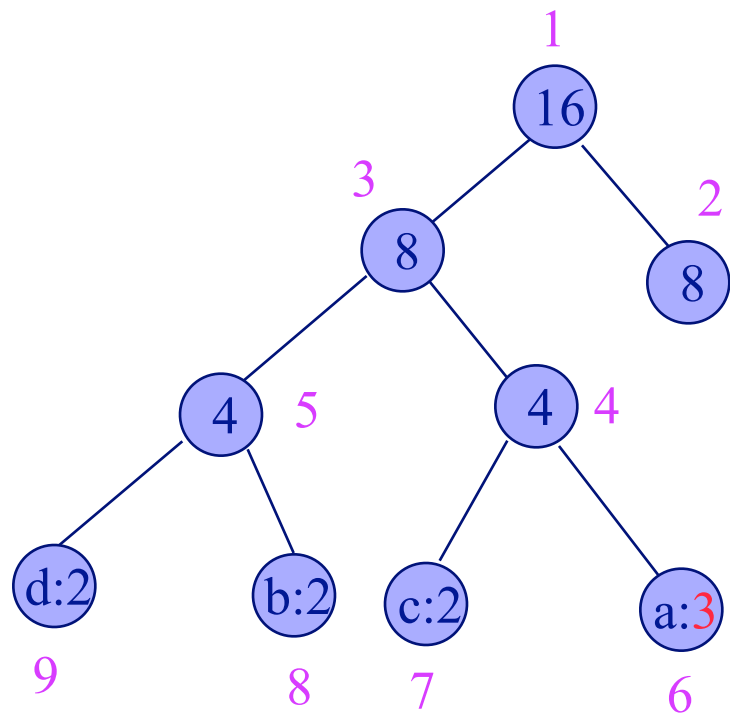
načti znak 'a'

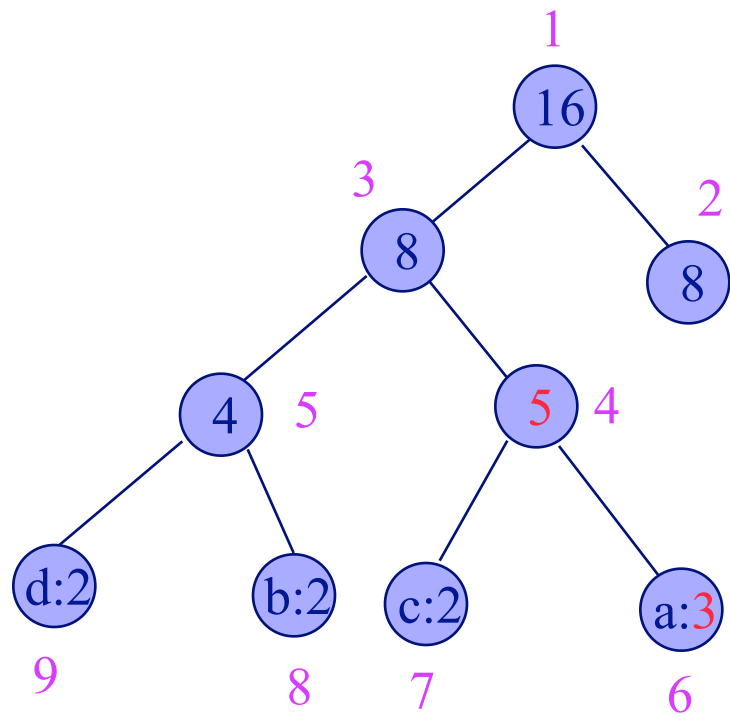


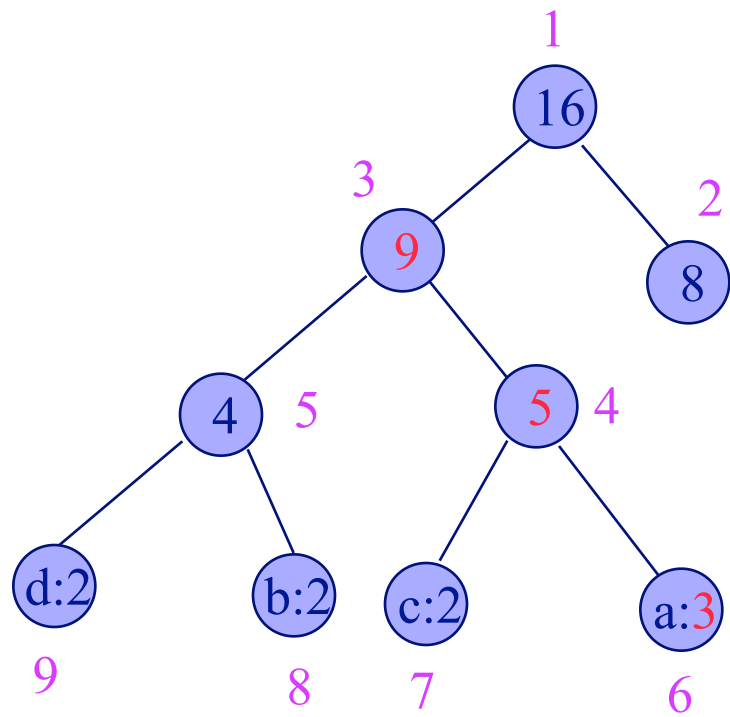


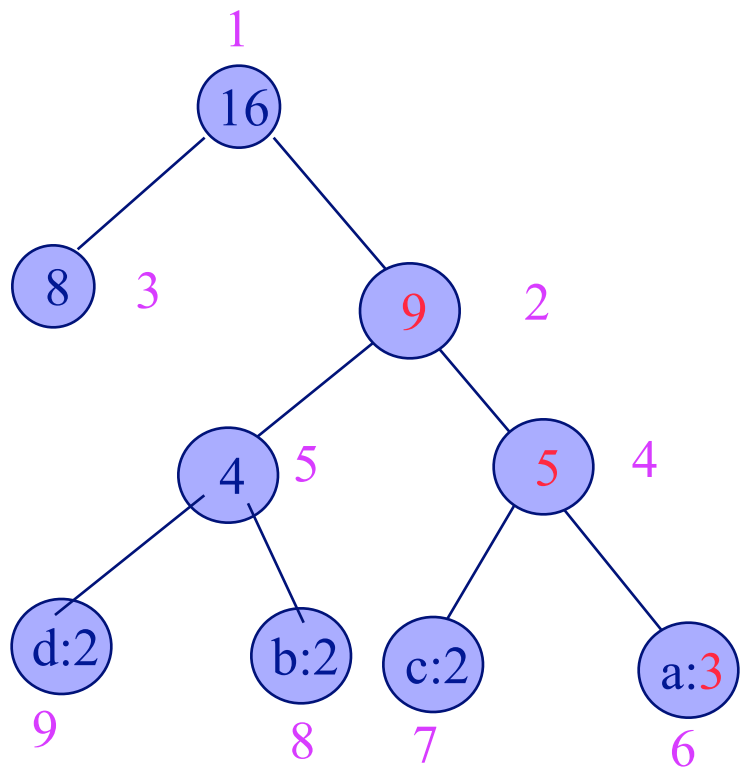
načti znak 'a'

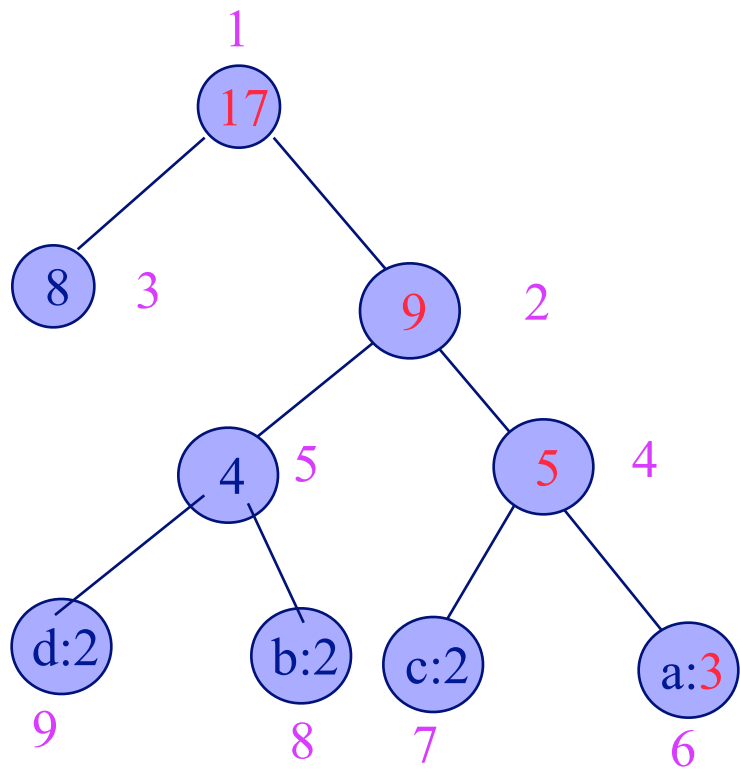












Problém nulových četností

Jak kódovat znaky, které jsou načteny poprvé?

1. řešení: Při počáteční inicializaci jsou do Huffmanova stromu vloženy všechny znaky vstupní abecedy, každý s četností 1.

2. řešení: Při počáteční inicializaci je do Huffmanova stromu vložen spec. znak ESC.

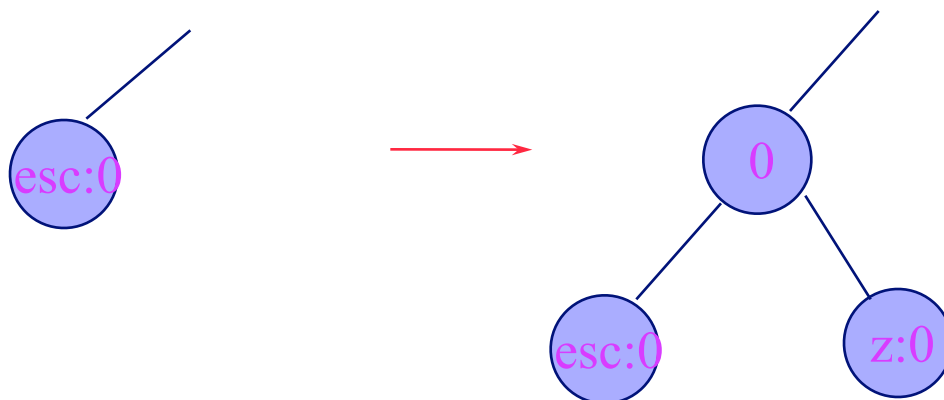
- první výskyt znaku z je kódován jako Huffmanův kód znaku ESC, následovaný znakem z
- poté je do Huffmanova stromu vložen nový list, odpovídající znaku z

Problém znaků, které jsou načteny poprvé

Počáteční inicializace Huffmanova stromu



z je nově načtený znak, který se ve stromě nevyskytuje

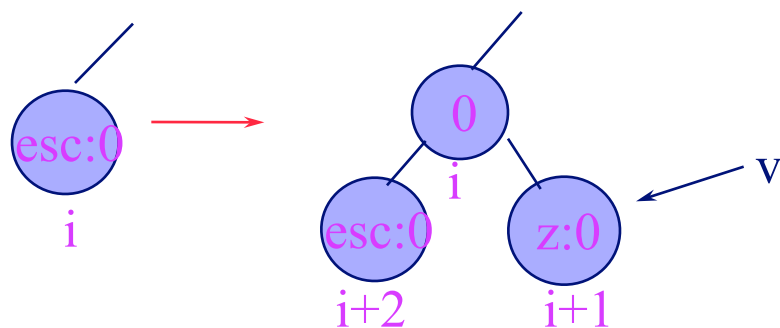


aktualizuj strom

Aktualizace Huffmanova stromu

z je znak načtený na vstupu

if z se ve stromě nevyskytuje
then



else $v :=$ list Huffmanova stromu
se znakem z

Aktualizace Huffmanova stromu

```
if v je sourozenec vrcholu esc
then vyměň v s listem, který má nejnižší pořadí mezi
      vrcholy se stejnou váhou jako v
      v.váha++; v := otec(v)
while v ≠ kořen-stromu
do vyměň v s vrcholem, který má nejnižší pořadí mezi
     vrcholy se stejnou váhou jako v
     (vymění se celé podstromy)
     v.váha++; v := otec(v)
```

je-li z poslední znak,
lze ho uložit přímo do vrcholu esc

FGK:Kódování

```
InicializujHuffmanůvStrom( $T$ )  
repeat read( $znak$ )  
    if první výskyt  $znak$   
    then write(kód(ESC))  
        write( $znak$ )  
    else write(kód( $znak$ ))  
        aktualizuj strom( $T, znak$ )  
until EOF.
```

FGK:Dekódování

```
InicializujHuffmanůvStrom( $T$ )  
 $vrchol :=$  kořen-stromu  
repeat  
while  $vrchol$  není list do read( $bit$ )  
if  $bit=0$  then  $vrchol:=vrchol.levý-syn$   
                  else  $vrchol:=vrchol.pravý-syn$   
if  $vrchol.znak=ESC$  then read( $znak$ )  
else  $znak := vrchol.znak$   
zapiš  $znak$  na výstup  
AktualizujStrom( $T, znak$ )  
until EOF.
```

Vitterův algoritmus

J.S.Vitter (1987)

- pouze 1 výměna při aktualizaci stromu
- FGK nejvýše $l/2$ výměn, kde l = délka právě zapsaného k. s.
- Vitter minimalizuje $\sum_i l_i$ a $\max_i l_i$ (l_i = délka i -tého k. s.)

l_A - průměrná délka kódového slova pro algoritmus A

$$l_V \leq l_H + 1$$

$$l_{FGK} \leq l_H + O(1) \text{ (Milidiu, Laber, Pessoa, 1999)}$$

- 📖 D.E.Knuth: Dynamic Huffman Coding. *J. Algorithms* 6(1985),163-180.
- 📖 J.S.Vitter: Design and analysis of dynamic Huffman codes. *J. ACM* 34(1987),825-845.
- 📖 R.L.Milidiú, E.S.Laber,A.A.Pessoa: Bounding the compression loss of the FGK algorithm. *J. Algorithms* 32(1999),195-211.

Empirické výsledky

E.R.Fiala, D.H.Greene

	SC	TM	NS	CC	BF	SF	RCF	SNI	SCI	BI
FGK	75.1	62.5	59.5	80.4	75.6	63.7	76.7	41.5	85.0	20.5
V	74.9	62.4	59.5	80.2	75.6	63.7	76.6	41.4	85.0	20.5

SC zdrojový kód

TM ASCII (technické memoranda)

NS ASCII (news service)

CC zkompilovaný kód

BF boot file

SF splajnové fonty

RCF bitové mapy fontů kódované RLE

SNI syntetické obrázky

SCI digitalizované barevné fotografie
(8bitů/pixel)

BI digitalizované faxové dokumenty

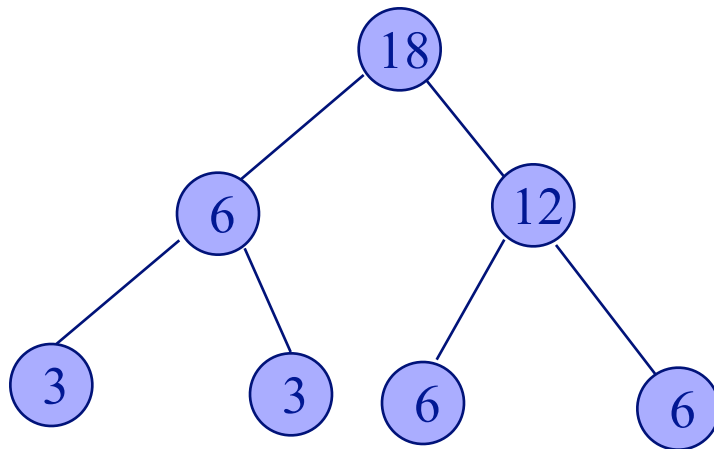
Implementační poznámky

Problém přetečení

- délka nejdelšího kódového slova
- délka souboru (kořen-stromu.váha)

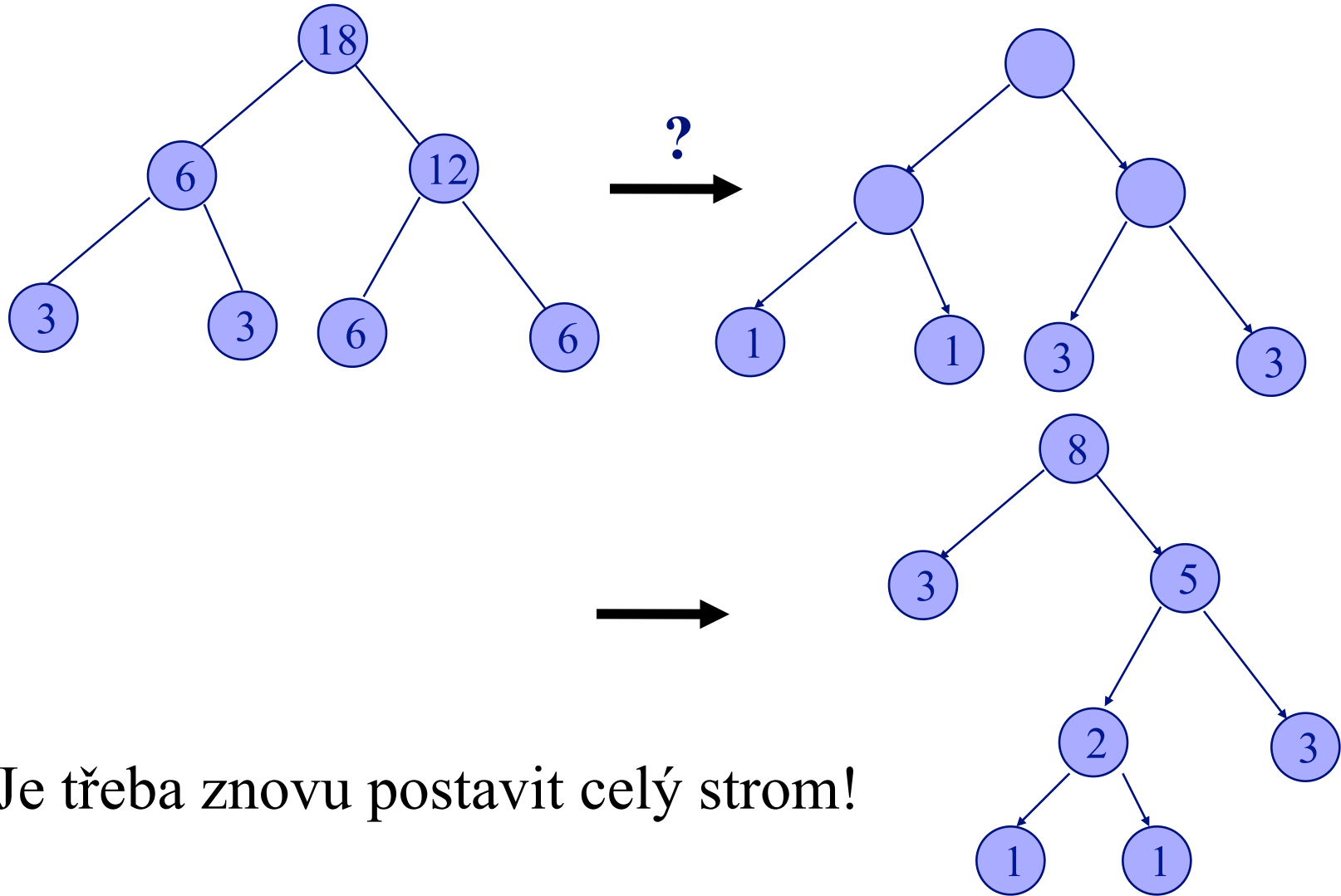
Řešení: vynásobím váhy všech vrcholů koeficientem $r < 1$.

Nevýhoda: nutná reorganizace stromu



$$r = 1/2$$

Reorganizace stromu



Je třeba znovu postavit celý strom!

„Stárnutí“ statistik

každá četnost $\neq 2 \Rightarrow$ starší četnosti mají nižší váhu nežli nové
 \Rightarrow lepší kompresní poměr

každá četnost $\cdot = (1-x)$ v každém kroku, $x \rightarrow 0$

četnost aktuálního znaku $\cdot = (1+x)^t$ v čase t

Příklad: chceme $(1+x)^d = 2$

- $x \approx \ln(1+x)$ pro $x \rightarrow 0$
- $\Rightarrow x \approx (\ln 2)/d$ pro $x \rightarrow 0$

Volba d : stabilní rozložení psti \times časté fluktuace

- $d = 1000$, $1+x = 1.00069$
- celočíselná aritmetika \Rightarrow škálování
- 1.krok: inkrement 10000
- 2.krok: inkrement $\lceil 10000(1+x) \rceil = 10007$
- 3.krok: inkrement $\lceil 10000(1+x)^2 \rceil = 10014$
- po dosažení horní meze všechny četnosti $\neq 2$