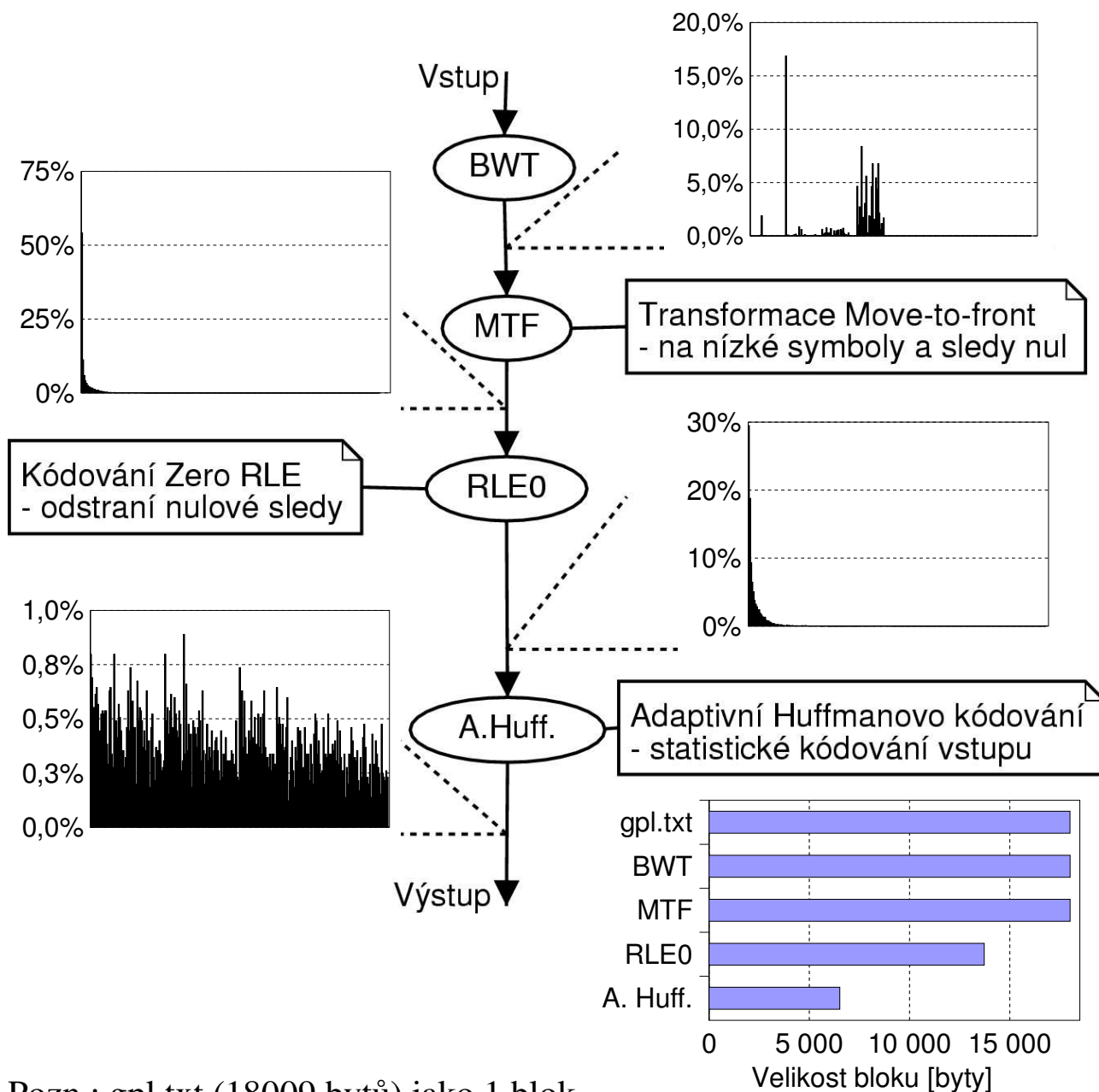


# POUŽITÍ BURROWS-WHEELEROVY TRANSFORMACE

- moderní bezztrátové kompresní algoritmy
- BWT: přeuspořádání symbolů v bloku -> sledy

Blok	mississippi mississippi mississippi
BWT	pppssssssmmmiippiiissssssiiii

- Kompresní algoritmus (BWC)



Pozn.: gpl.txt (18009 bytů) jako 1 blok

# BURROWS-WHEELEROVA TRANSFORMACE

## Kompresní transformace

1. Cyklické rotace bloku o 1 symbol vlevo -> matice  $N \times N$
2. Vzestupné seřazení řádků matice
3. BWT = poslední sloupec ( $L$ ) + index ( $I$ ) původního řádku

	$F$	$L$	
0 KAPKA	4 AKAP	<b>K</b>	0
1 APKAK	1 APKAK	<b>K</b>	1
KAPKA -> 2 PKAKA	-> 3 KAKAP	<b>P</b>	2 -> KKPAA x 3
3 KAKAP	0 <u>KAPKA</u>	<b>A</b>	3
4 AKAPK	2 PKAKA	<b>A</b>	4

Problém: bloky  $\approx 10^6$  symbolů, nejhorší časová složitost  $O(n^2)$

## Dekompresní transformace

- Obnovení  $I$ -tého řádku matice,  $F$  cyklicky následuje za  $L$
- $n$ -tý výskyt symbolu v  $m$ -tém sloupci =  $n$ -tý v  $m \pm 1$

1. Vytvoření  $F$  (první sloupec) seřazením  $L$  (stabilně!) resp. seřazením ukazatelů (transformační vektor  $R$ ), tj. platí:  $F[n] = L[R[n]]$

	0 1 2 <u>3</u> 4	$I$	$I$
KKPAA x 3 ->	KKPAA	$L$	
	34012	$R$	
	( AAK <u>K</u> P	$F$	)

2. Jednoduchý cyklus (v  $R[n]$  je index následníka v  $F$ )

$K_3 \times 1 \rightarrow KA_1 \times 4 \rightarrow KAP_4 \times 2 \rightarrow KAPK_2 \times 0 \rightarrow KAPKA_0$

# TRANSFORMACE MOVE-TO-FRONT

- Seznam symbolů abecedy, frekventované na začátku
- Kódování pozice symbolu + přesun na začátek

Krok	Vstup	Seznam	Výstup
1	<b>K</b> KPAA	<b>K</b> PA	<b>0</b>
2	<b>K</b> PAA	<b>K</b> PA	0 <b>0</b>
3	<b>P</b> AA	K <b>P</b> A	00 <b>1</b>
4	<b>A</b> A	PK <b>A</b>	001 <b>2</b>
5	<b>A</b>	<b>A</b> PK	0012 <b>0</b>

- Charakteristika výstupu (v návaznosti na BWT)
  - omezené spektrum symbolů
  - sled délky  $n$  -> sled  $n-1$  nul

Problém: počáteční konfigurace seznamu -> celá ASCII

## KÓDOVÁNÍ ZERO RLE

- Efektivní kódování pouze jednoho typu sledů (nul)
- Symbol + počet následujících (0 pro žádný)  
tj. max. délka sledu pro ASCII:  $1+(2^8-1) = 256$

Vstup	111 <b>00000</b> 222 <b>0</b> 333
Výstup	111 <b>04</b> 222 <b>00</b> 333

# ADAPTIVNÍ HUFFMANOVO KÓDOVÁNÍ

- Huffmanův strom jen pro právě zpracovanou část vstupu (na začátku je strom „prázdný“)
- Pouze 1 průchod a respektování lokálních změn četností
- Jak (de)kódovat nové symboly?

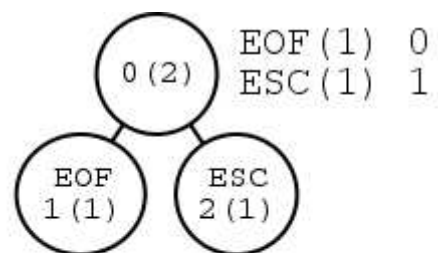
Kód ESCAPE – speciální uzel stromu (ESC), předchází nový symbol

- Jak dekódovat „zarovnaný“ konec vstupu?

Kód END\_OF\_STREAM (uzel EOF)

- Hlavní problém

- efektivní aktualizace (update) při zachování Huffmanova stromu + přetečení ohodnocení kořenu



Vstup: ' m'  
Výstup: ' 1m' + aktualizace stromu

## Model kódování

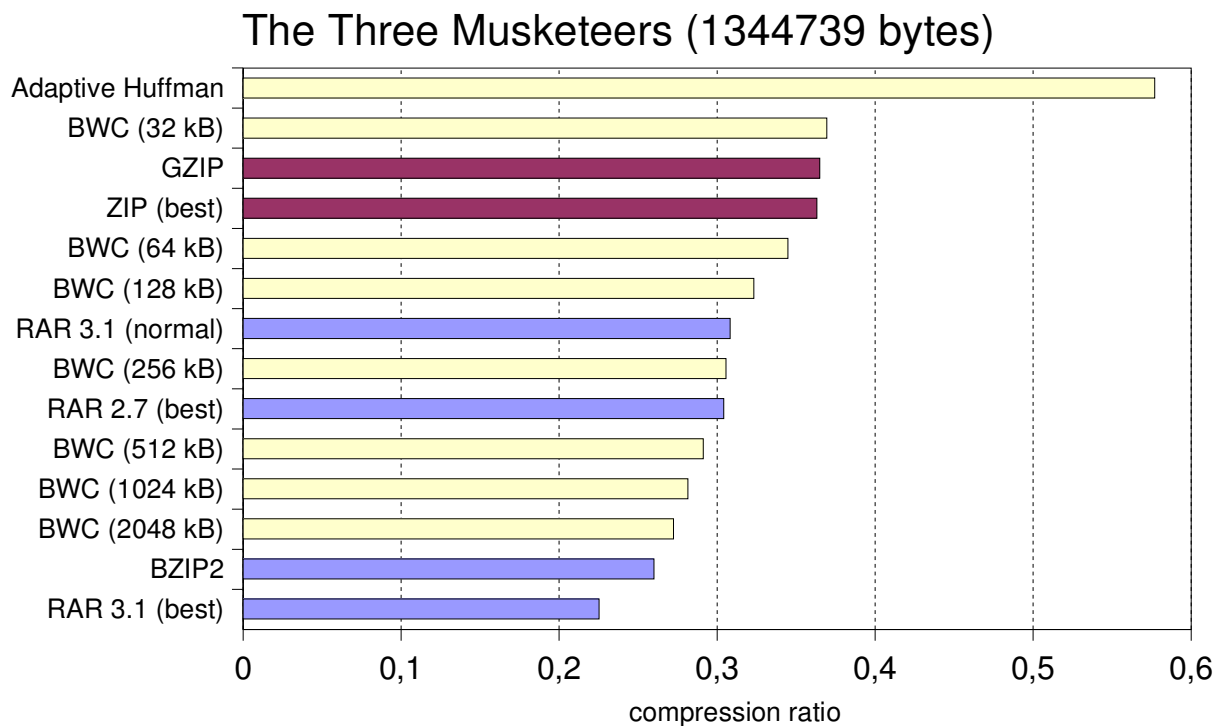
```
init();  
while((symbol = input.get())  
      != END_OF_INPUT) {  
    encode(symbol, output);  
    update(symbol);  
}  
encode(END_OF_STREAM, output);
```

## Model dekódování

```
init();  
while((symbol = decode(input))  
      != END_OF_STREAM) {  
    output.put(symbol);  
    update(symbol);  
}
```

# EXPERIMENTÁLNÍ VÝSLEDKY

- Srovnání kompresních programů (BWC vs. ostatní)



- BWT (BWC, BZIP2) vs. slovníkové metody (GZIP)

