Huffman kód

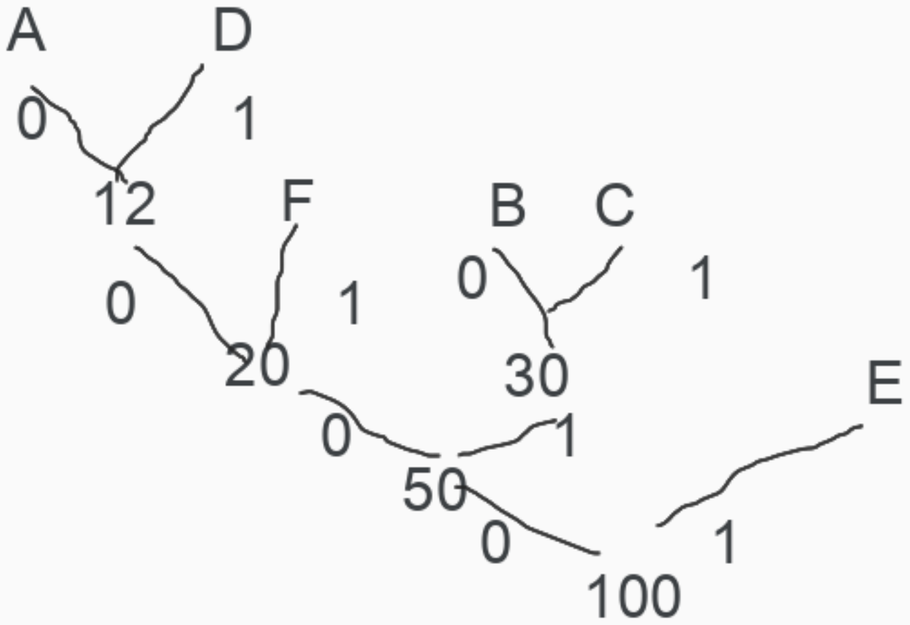
Huffman kód:

Számolás:

Naiv módszer: 6 KAR: 2^3 3 BIT

Huffman: 100*3=300 bit

A	B	C	D	E	F	AD	ADF	BC.	ADFBC.	ADFBCE
5	13	17	7	50	8	12	20	30.	50.	100



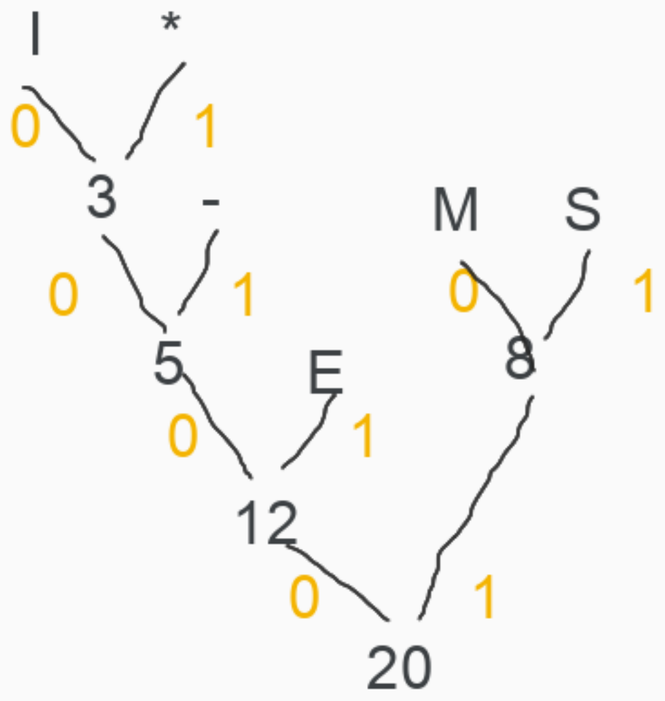
A 0000
B 010
C 011
D 0001
E 1
F 001

4*5=20
3*13=39
3*17=51
4*7=28
1*50=50
3*8=24
össz:212

MESE-EMESE-SEMMISE

Huffman kód

*	M	E	S	-	I	I*	I*-
2	4	7	4	2	1	3	5



*	0001	2*4=8
M	10	8
E	01	14
S	11	8
-	001	6
I	0000	4

NAIV: 3*20=60

HUFFMAN: 48

1010010000

M M E I

LZW

ABA B C A B A B A

OUTPUT: 1 1 2 4 3 4 8

string	Kód	Alternatív kód
A	1	A
B	2	B
C	3	C
AB	4	1B
BA	5	2A
ABC	6	4C
CA	7	3A
ABA	8	4A

LZW

Kódolt szöveg: 1 2 4 3 4 8

A B AB C AB ABA

String	kód
A	1
B	2
C	3
AB	4
BA	5
ABC	6
CA	7
ABA	8

LZW

Output:	3	1	2	5	4	6	9	7	11	2
	C	A	B	AB	CA	BA	BAB	ABC	ABCA	B

String	kód
A	1
B	2
C	3
CA	4
AB	5
BA	6
ABC	7
CAB	8
BAB	9
BABA	10
ABCA	11
ABCAB	12

s c

IN: A A A A A A A A A A

OUT: 1 5 6 7

D

A 1
B 2
C 3
D 4
AA 5
AAA 6
AAAA 7

A B A

AB
BA

LZWcompress($In : \Sigma\langle\rangle$; $Out : \mathbb{N}\langle\rangle$)

- $D : Item\{\}$ // D is the dictionary, initially empty
- $i := 1$ **to** $|\Sigma|$
  - $x : Item(\langle\Sigma_i\rangle, i)$; $D := D \cup \{x\}$
- $code := |\Sigma| + 1$; $Out := \langle\rangle$; $s : \Sigma\langle In_1\rangle$
- $i := 2$ **to** $|In|$
  - $c : \Sigma := In_i$
  - dictionaryContainsString($D, s + c$)
    - $Out := Out$+code(D, s)
    - $s := s + c$
    - $code \leq MAXCODE$
    - $x : Item(s + c, code++)$; $D := D \cup \{x\}$ | SKIP
    - $s := \langle c\rangle$
- $Out := Out$+code(D, s)

IN:	1	2	3	4	5	6	7	8	9	10
OUT:	A	B	C	D	AB	BC	CD	DA	ABB	BCC

A 1
B 2
C 3
D 4
AB 5
BC 6

CD 7
DA 8
ABB 9
BBC 10

CDD 11
DAA 12

ABBB 13

				k				
IN:	1	2	2	3	6	4	8	
OUT:	A	B	B	AB	ABA	BB	BBB	
						s		t

D:

A 1
B 2
AB 3
BB 4
BA 5
ABA 6
ABAB 7
BBB 8

LZWdecompress(In : $\mathbb{N}\langle\rangle$; Out : $\Sigma\langle\rangle$)

D : $Item\{\}$ // D is the dictionary, initially empty

$i := 1$ **to** $|\Sigma|$

  x : $Item(\langle\Sigma_i\rangle, i)$; $D := D \cup \{x\}$

$code := |\Sigma| + 1$ // $code$ is the first unused code

$Out := s := string(D, In_1)$

$i := 2$ **to** $|In|$

  $k := In_i$

  $k < code$ // D contains k

$t := string(D, k)$	$t := s + s_1$
$Out := Out + t$	$Out := Out + t$
x : $Item(s + t_1, code)$ $D := D \cup \{x\}$	x : $Item(t, k)$ // k=code $D := D \cup \{x\}$

  $s := t$; $code++$