Tillegg til i går. Tallet 129 kan
lagres som
$129 = 128 + 1 = 10000001$,
altså ved å lagre de binære sifrene.
Men også ved å lagre de desimale
sifrene som tegn, '129', for eksempel
i UTF-8

Informasjonseksplosjon — Kap 7.1 i komp.

1. Bøker. Typisk bok har 300 ord pr. side, ca. 4 tegn pr. ord. Med 500 sider gir det ca. 600 000 tegn i hele boka. Med IsoLatin eller UTF-8 kan vi regne 1 byte pr. tegn, altså til sammen 600000 bytes 600 KB, 0.6 MB.
Med UTF-16 dobles dette til 1.2 MB.

2. Lyd. På en CD er lyden målt 44100 ganger pr. sekund. Hver måling lagres som et heltall med 2 bytes. Siden vi har stereo blir 4 x 44100 bytes pr. sekund, i tilsammen 176 KB pr. sekund. Det gir 10 MB pr. minutt og omtrent 40 MB for en låt. På en CD som inneholder en time musikk har vi da 600 MB med data.

3. Film og bilder.

TV-bilde består av 576 × 720 punkter og i hvert punkt er det 24 bits (3 bytes) med farge info. Et bilde krever derfor 1.2 MB. For å få anstendig kvalitet 25 bilder pr. sekund, så ett sekund gir da omtrent 31 MB, eller 1.9 GB pr. minutt og 112 GB pr. time.

Huffman koding

Grunnleggende ide. Bruk korte koder for tegn som forekommer ofte og lange koder for tegn som forekommer sjelden.

Noen begreper.

En tekst $x = \{x_1, x_2, x_3, \ldots, x_n\}$ er en sekvens av symboler hentet fra et alfabet $\mathcal{A} = \{\alpha_1, \alpha_2, \ldots, \alpha_m\}$

Antall ganger α_i forekommer i x angis med frekvensen $f(\alpha_i)$.

For kompresjon gir vi hvert tegn α_i en kode $c(\alpha_i)$. Og z er den komprimerte teksten der hvert tegn i x er erstattet med sin kode

$$z = \{c(x_1)\ c(x_2)\ c(x_3)\ \ldots\ c(x_n)\}$$

Ex Anta at $x = DBACDBD$, $\mathcal{A} = \{A, B, C, D\}$

$f(A) = 1$, $f(B) = 2$, $f(C) = 1$, $f(D) = 3$.

$c(D) = 0$, $c(B) = 1$, $c(C) = 01$, $c(A) = 10$

Vi erstatter tegnene med koder og får

$z = 011001010$ – 9 bits

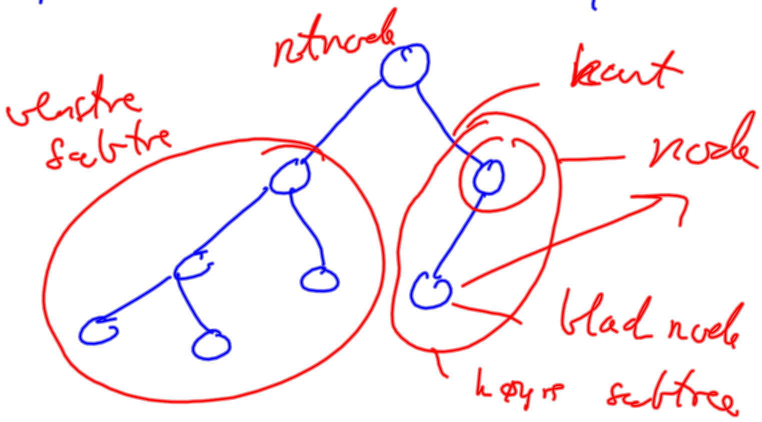
Ex 2. Samme tekst, men kodene

$c(D) = 1$, $c(B) = 01$, $c(C) = 001$, $c(A) = 000$

$z = 1010000011011$ 13 bits

Binære trær.

Et tre består av noder og kanter. For at binært går det to kanter fra hver node.



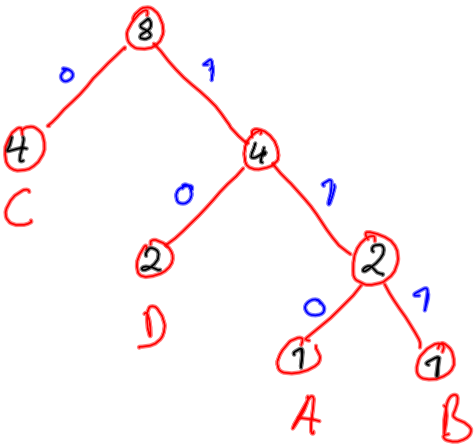
Definisjon av Huffman tre

Et Huffman tre er et binært tre som er assosiert med et alfabet $\{\alpha_i\}_{i=0}^{n}$ med frekvenser $f(\alpha_i)$

1. Hver bladnode er assosiert med nøyaktig en α_i
2. Hver node har en vekt
   (a) Vekten av en bladnode er frekvensen til dens symbol
   (b) Vekten til en annen node er summen av vektene til røttene i nodens subtrær
3. Alle noder som ikke er bladnoder har nøyaktig to barn.
4. Fra Huffman treet kan vi bygge koder.

Ex. $X = CCDACBDC$, $\mathcal{A} = \{A, B, C, D\}$

$f(A) = 1$, $f(B) = 1$, $f(C) = 4$, $f(D) = 2$



$c(C) = 0$, $c(D) = 10$

$c(A) = 110$, $c(B) = 111$

Huffman algoritmen.

Gitt x basert på alfabet $\{\alpha_i\}_{i=1}^{n}$ med frekvenser $f(\alpha_i)$

1. Lag et en-node Huffman tre med hvert av de n symbolene α_i.

2. Gjenta inntil vi bare har ett tre

$$\begin{array}{ccccc} \circ & \circ & \circ\,f(\alpha_3) & \cdots & \circ \\ \alpha_1 & \alpha_2 & \alpha_3 & & \alpha_n \end{array}$$

   (a) Velg to trær T_0 og T_1 med minimal vekt erstatt dem med et nytt tre som har T_0 som venstre subtre og T_1 som høyre subtre.

3. Treet som er igjen til slutt er Huffman treet.

Huffman-algoritmen gir optimal koding blant binær tre algoritmer med prefiks egenskap.