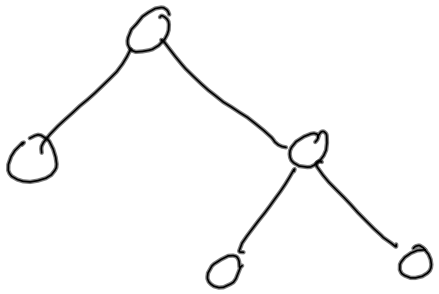
Huffman-koding

Binære trær



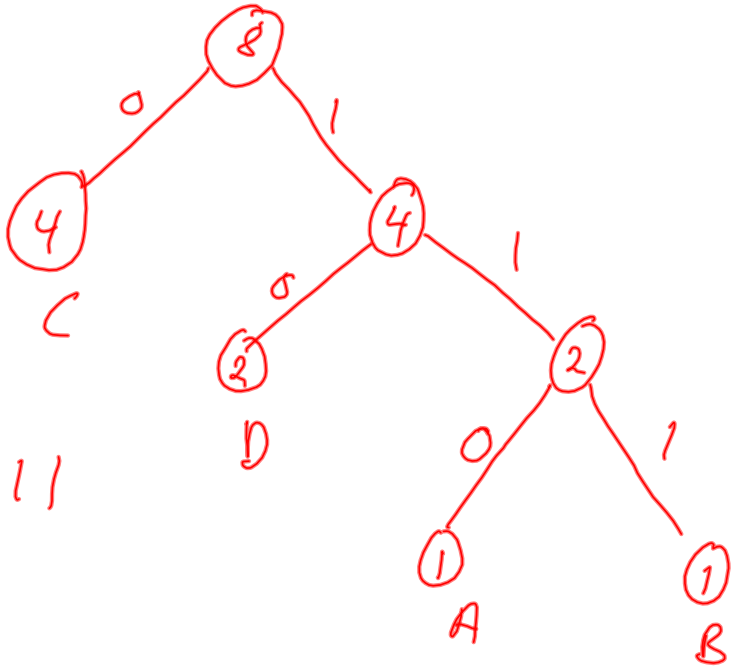
Huffman-tre: Def 7.6

Et binært tre assosiert med et alfabet $A = \{\alpha_i\}_{i=1}^{n}$ med frekvenser $f(\alpha_i)$ – hyppigheten til hvert tegn.

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

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

Huffman-tre:



Fra Huffman-treet kan vi skrive opp kodene

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

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

Algoritme for å lage Huffman-tre

Gitt $A = \{\alpha_i\}_{i=1}^{n}$, tekst x med frekvenser $f(\alpha_i)$, $i = 1, \ldots, n$.

1. Lag en-node Huffmantrær fra hvert av de n symbolene i alfabetet.
2. Repeter inntil bare ett tre:
   a) Velg to trær T_0 og T_1 med minimale vekter og slå de sammen til ett stort tre med T_0 som venstre subtre og T_1 som høgre subtre.
3. Tre som står igjen er et Huffman-tre for teksten x.

Se eksempel 7.9

Observasjoner.

1. Huffman-trær har prefiks-egenskapen
2. Huffmankoding er den beste av alle kodingsteknikker basert på binære trær.

Sannsynlighet istedenfor hyppigheter

Hvis vi har en tekst x med alfabet A og frekvenser $f(\alpha_i)$ blir totalt antall bits i koden

$$B = \sum_{i=1}^{n} f(\alpha_i) \ell(\alpha_i)$$

der $\ell(\alpha_i)$ er lengden av kodeordet til α_i.

B er et uheldig mål på kvaliteten fordi det er avhengig av lengden på teksten.

Bedre: Antall bits pr. tegn (gjennomsnitt)

Hvis teksten har lengde m, bruk:

$$\bar{B} = \frac{B}{m} = \frac{1}{m} \sum_{i=1}^{n} f(\alpha_i) \ell(\alpha_i)$$

$$= \sum_{i=1}^{n} \frac{f(\alpha_i)}{m} \ell(\alpha_i)$$

$$= \sum_{i=1}^{n} P(\alpha_i) \ell(\alpha_i)$$

Shannons teorem. Gitt alfabet $\{\alpha_i\}_{i=1}^{n}$ med sannsynligheter $P(\alpha_i)$. Da er minimalt antall bits pr. tegn

$$H = -\sum_{i=1}^{n} P(\alpha_i) \log_2 P(\alpha_i)$$

H kalles informasjonsentropien.

NB! $x = 2^{\log_2 x}$. Ln på begge sider

$$\ln x = \log_2 x \cdot \ln 2$$

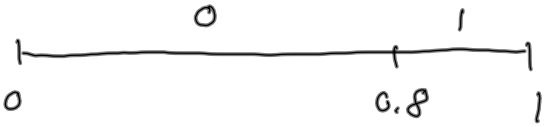
$$\log_2 x = \ln x / \ln 2$$

Aritmetisk koding

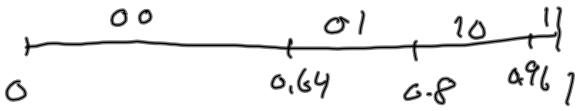
Ide. En tekst assosieres med et intervall i $(0,1)$. Bredden til intervallet svarer sannsynligheten til teksten.

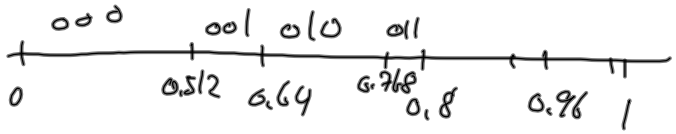
Eks. 7.17. $X = 00100$. $P(0) = 0.8$, $P(1) = 0.2$

Vi allokerer ulike deler av $[0,1]$ til 0 og 1, i forhold til sannsynlighetene



Deler hvert av de to intervallene på samme måte:





Fortsetter: Teksten 00100 svarer til intervallet $[0.512, 0.59392)$

Den aritmetiske koden svarer til et tall på formen $k/2^k$ som ligger i dette intervallet, med minst mulig k. Svar $9/16 = 0.1001_2$.

Vi lagrer 1001.