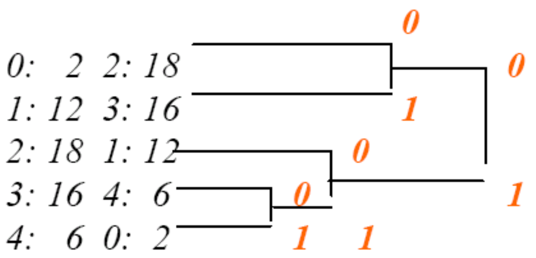
Løsningsforslag, Ukeoppgaver til uke 17 INF2310, våren 2010
Kompresjon og koding – del I

1. Vi har gitt følgende bilde:

1	0	1	2	2	2	3	3	3
1	1	1	2	1	1	3	3	3
1	0	1	1	2	2	2	3	3
2	1	2	2	3	2	3	4	4
2	1	2	3	2	2	3	4	4
2	2	2	3	3	3	4	3	4

a. **Finn Huffman-kodingen av dette bildet. Hvor mange biter blir det per piksel I gjennomsnitt etter koding – hvis vi ser bort fra at vi trenger plass til å lagre kodeboken?**

Vi har følgende forekomster, først i rekkefølge etter symbolverdiene, deretter sortert. Slår sammen først 0 og 4, så dette med 1, så 3 og 2 sammen, så dette med resten, som vist i grafen nedenfor:



Merk: vi ikke har laget frekvenstabell – bare brukt histogrammet.

Huffman-kodene blir da (her er det flere mulige riktige løsninger!):
2: 00
3: 01
1: 10
4: 110
0: 111
Og det gjennomsnittlige antall bits pr piksel etter koding blir
((18+16+12)*2+8*3)/54 = (92+24)/54 = 116/54 =**2.15 bits per piksel.**

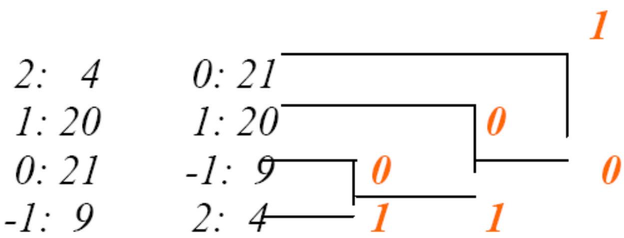
b. **Ved differansetransform tar vi differansen mellom et piksel og dets nabo til venstre. Siden pikslene lengst til venstre i bildet ikke har noen venstre nabo, beholder vi pikselverdien her.**
Finn differanse-transformen av bildet ovenfor.

Etter differansetransform vil bildet se slik ut:

1	-1	1	1	0	0	1	0	0
1	0	0	1	-1	0	2	0	0
1	-1	1	0	-1	0	0	1	0
2	-1	1	0	-1	-1	1	1	0
2	-1	1	1	1	0	1	1	0
2	0	0	1	0	0	1	-1	1

c. Finn så Huffman-koden for det differansetransformerte bildet, slik at du kan beregne det gjennomsnittlige antall bits per piksel for det differansetransformerte bildet.

Vi har følgende forekomster, først i rekkefølge etter symbolverdiene, deretter sortert. Slår sammen først -1 og 2, så dette med 1, så dette med 0, som vist i grafen nedenfor:



Huffman-kodene blir da (her er det igjen flere mulige riktige løsninger!):

0: 1
1: 00
-1: 010
2: 011

Og det gjennomsnittlige antall bits pr piksel etter differansetransform og Huffman-koding blir
*(21+20*2+13*3)/54 = (21+40+39)/54 = 100/54 =**1.85 bits per piksel.***

Altså færre bits pr symbol ved å gjøre differansekoding først og så Huffman, enn med bare Huffman.

d. Entropien til bildet vi startet med er 2.06. Hvorfor ble det gjennomsnittlige antall biter per piksel større enn entropien i deloppgave a, men mindre enn 2.06 i deloppgave c?

Entropien beregnet fra histogrammet til bildet er en nedre grense for hvor kompakt bildet kan kodes, hvis vi bare ser på ett piksel av gangen.

Denne grensen er bare mulig å oppnå hvis alle sannsynlighetene i det normaliserte histogrammet er av typen 1/2k, der k er et heltall.
Dette kravet er ikke oppfylt i det opprinnelige bildet.
Derfor er ikke Huffman-transformen optimal, og vi får et gjennomsnittlig antall bits som er litt større enn entropien.

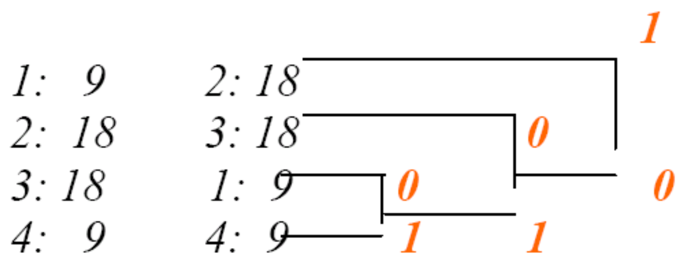
I deloppgave c ser vi ikke bare på ett piksel av gangen, men på differansen mellom to og to piksler. Da kan vi kode mer kompakt enn det som er gitt av entropien for enkelt-piksler.

e. Hvis det bildet du fikk oppgitt i starten av oppgaven var det andre bildet i en bildesekvens, og det første var

1	1	1	2	2	2	3	3	3
1	1	1	2	2	2	3	3	3
1	1	1	2	2	2	3	3	3
2	2	2	3	3	3	4	4	4
2	2	2	3	3	3	4	4	4
2	2	2	3	3	3	4	4	4

Hvilke to bilder ville du da komprimere, og hvor mange biter vil du i gjennomsnitt trenge per piksel for hvert av de to bildene?

Man ville først komprimere det bildet som kommer først (det rett ovenfor).
Her er det forholdsvis greit å finne histogrammet og Huffman-koden: 2: 1; 3: 00; 1: 010; 4: 011



*Det gjennomsnittlige antall bits pr piksel blir (18*1+18*2+9*3+9*3)/54 = 108/54*
=2 bits per piksel.
Og så ville man kode differansebildet, dvs det sist mottatte bildet minus det forrige, piksel for piksel. Det ser slik ut:

0	**1**	0	0	0	0	0	0	0
0	0	0	0	**1**	**1**	0	0	0
0	**1**	0	**1**	0	0	**1**	0	0
0	**1**	0	**1**	0	**1**	**1**	0	0
0	**1**	0	0	**1**	**1**	**1**	0	0
0	0	0	0	0	0	0	**1**	0

Dette er et binært bilde som bare trenger 1 bit per piksel.

2. Anta at det er $G=2^8$ forskjellige gråtone-nivåer i hvert sample, og at når vi sorterer dem etter hvor ofte de forekommer i et bilde, så finner vi i et spesielt tilfellet at sannsynlighetene er ½, ¼, 1/8, 1/16, …, 1/128, 1/256, 1/256.

a. Hvor mange slike bilder kan vi overføre i parallell på en 64 kbits/s linje med Huffman-koding av amplitudene?

Hint 1: Entropien er gitt ved

$$H = -\sum_{i=0}^{G-1} p_i \log_2(p_i)$$

Dessuten: log(teller/nevner) = log(teller) – log(nevner)
Og til slutt: $\log_2(2_n) = n$

Hint 2: Summen ½+2/4+3/8 +4/16+5/32 + … konvergerer raskt mot 2.

Her er det ikke nødvendig å finne Huffman-koden.

*Vi har terpet at en Huffman-koding der alle sannsynlighetene kan skrives som brøker der telleren er 1 og nevneren er en toer-potens – er optimal i den forstand at det gjennomsnittlige antall bits per sample er lik **entropien** til signalet.*

Entropien er her gitt ved
H = - (½ $\log_2(1/2)$ +1/4 $\log_2(1/4)$ + 1/8 $\log_2(1/8)$ + ...) (se "Hint 1")
= ½ + 2/4 + 3/8 + ... = 2 (som angitt i "Hint 2" ovenfor).
Det gjennomsnittlige antall bits per sampel blir altså bare 2 bits/sampel.

Men for å kunne svare på spørsmålet må vi vite hvor mange sampler vi får fra hvert bilde pr tidsenhet.
La oss si at samplingsfrekvensen er 8 kHz.
Da er antall bilder vi kan overføre i parallell:
64 000 bits/s /(8 000 sampler/s * 2 bits/sampel) =4.

3. **Optimal Huffman-koding:**

Finn kodeboken for en Huffman-koding av ”DIGITAL OVERALT!”. Hvorfor kan vi uten å gjøre noen logaritme-beregninger si hva entropien til denne teksten er?

Det sorterte histogrammet og kodeboken blir som vist i tabellen nedenfor

I	*2*	*010*
T	*2*	*011*
A	*2*	*000*
L	*2*	*001*
D	*1*	*1010*
G	*1*	*1011*
	1	*1000*
O	*1*	*1001*
V	*1*	*1110*
E	*1*	*1111*
R	*1*	*1100*
!	*1*	*1101*

Vi har i alt 16 tegn, men bare 12 forskjellige, og hyppighetene er enten 2 (i,t,a,l) eller 1 (d,g, ,o,v,e,r,!).

Så alle de N=12 sannsynlighetene kan skrives som brøker der telleren er 1 og nevneren er en toerpotens (8 eller 16). Altså kan sannsynlighetene for hvert symbol uttrykkes som

$$p(s_i) = \frac{1}{2^k}$$

for heltalls verdier av k.

Da vet vi at det gjennomsnittlige antall biter per symbol er lik entropien.

$$R = \sum_{i=1}^{N} p(s_i) b_i = -\sum_{i=0}^{N} p(s_i) \log_2(p(s_i)) = H$$

*Og R er lett å regne ut: R= (8*3 + 8*4) /16 = 56 /16 =3.5.*

Vi får selvsagt det samme hvis vi faktisk regner ut entropien:

$H = 4(1/2^3) \log_2(2^3) + 8 (1/2^4) \log_2(2^4) = 1.5 + 2 = 3.5$