

# Grådighet

Simon Jonassen  
Algoritmer og Datastrukturer  
Høst 2005

# Grådighet

- Optimal delstruktur
  - en optimal løsning av problemet inneholder en optimal løsning av delproblemet.
- *Greedy choice property*
  - en globalt optimal løsning kan oppnås ved å ta en lokalt optimal beslutning.  
(hva skjer hvis vi velger noe annet?)

# Eksempel

- *Fractional Knapsack Problem*

Definisjon: Gitt  $n$  enheter med ulike volum og verdi per enhet, finn den maksimale totalverdien som det er mulig å oppnå ved å pakke flere enheter inn i en sekk med ett gitt volum. Gitt at en kan ta en vilkårlig andel av hver enhet.

# 1. Determiner en optimal delstruktur av problemet:

$$W > w: \quad V_{w,v} = V_i + V_{w-w_i, v-v_i}$$

$$\text{ellers:} \quad V_{w,v} = v_i * W / w_i$$

(der  $i$  er indeks til *det optimale elementet*)

## 2. Konstruer en rekursiv løsning:

RFKP( $v, w, W$ ):

$i :=$  et optimalt element

*if* ( $W < w_i$ ):

*return*  $v_i * W / w_i$

*else* :

*return*  $v_i + \text{RFKP}(v - v_i, w - w_i, W - v_i)$

# 3. Bevis:

- Bevis at for hvert steg i rekursjonen er det grådige valget blant de optimale.
- Vis at det finnes minst et delproblem som er tomt og at dette delproblemet er blant de som blir induisert ved å ta det grådige valget.

# 3. Bevis (forts.)

- Optimal delstruktur
- Greedy-Choice-Property

Husk å sjekke at alle endringene er:

- Uforanderlige senere
- Stemmer med problembetingelsene

4. Evt. konstruer en grådig algoritme som er rekursiv.

//antar at w og v er avt. sortert etter v/w

GRFKP(v,w,W):

if  $w[0] \geq W$ :

return  $v[0] * W / w[0]$

else:

return  $v[0] + \text{GRFKP}(v[1:], w[1:], W - w[0])$

# 5. Konstruer en grådig algoritme som er iterativ

//antar at w og v er avt. sortert etter v/w

GIFKP(w,v,W):

  i=0;

  ret=0;

  while (W>0):

    if w[i]>=W:

      ret+=v[i]\*W/w[i]

      W=0

    else:

      ret+=v[i]

      W-=w[i]

    i+=1

  return ret

## 5. Konstruer en grådig algoritme som er iterativ (forts.)

//antar at w og v er avt. sortert etter v/w

GIFKP\_tweak(w,v,W):

  i=0

  ret=0

  while (W>0):

    ret+=min(W/w[i],1)\*v[i]

    W-=w[i]

    i+=1

  return ret

# Grådighet vs. DP

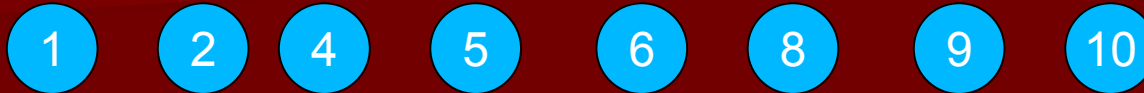
- +1 Er raskere
  - +1 Kan være lettere
  - -10 Virker ikke alltid
- 
- Noe felles?



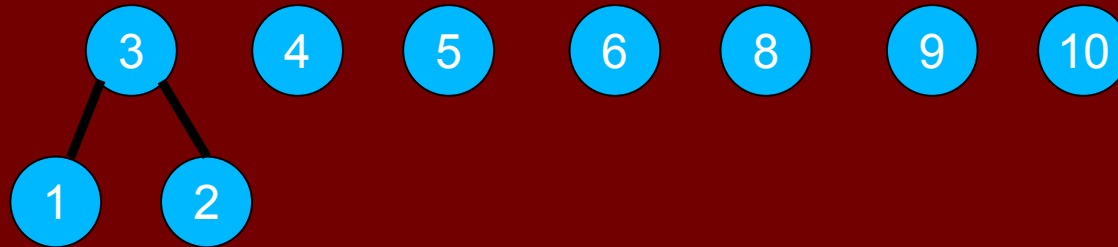
# Huffmankoding

- Prefikskoder
  - ingen gyldig kodelistreng er et prefiks i et annet gyldig kodelistreng.
- 0 10 111 110
- NB! 00 01 10 11 er også prefiks !
- Optimal prefikskoding?
- Et optimalt prefikstre er et fullt binært tre! \*

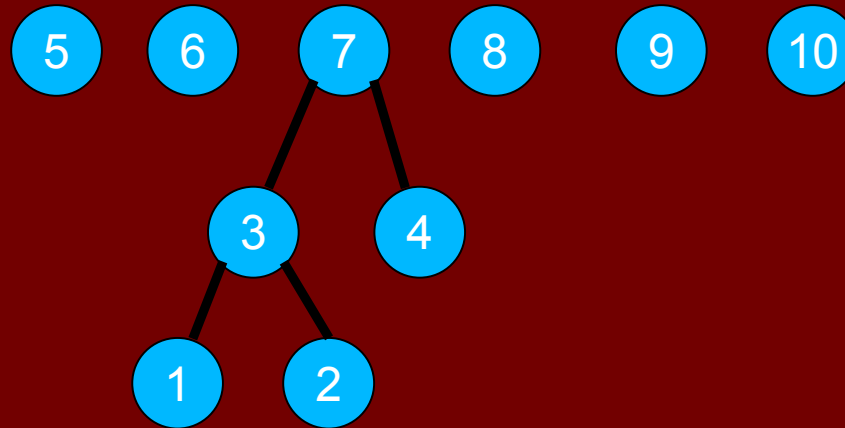
# Eksempel:



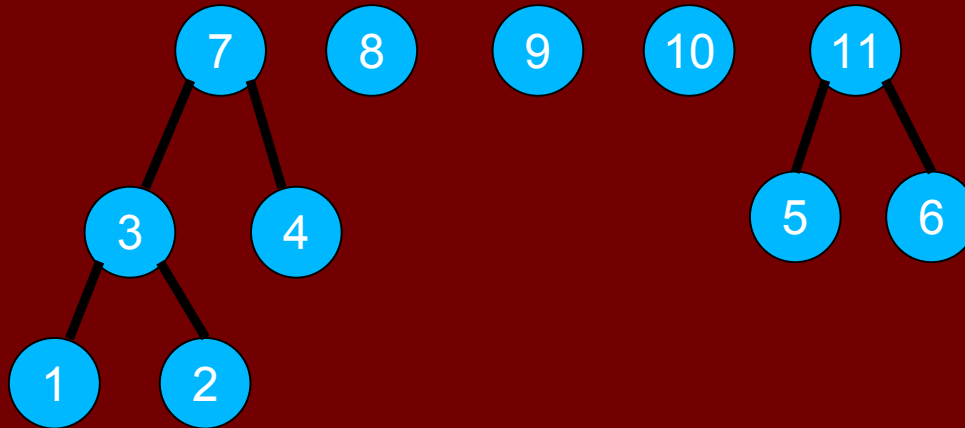
# Eksempel:



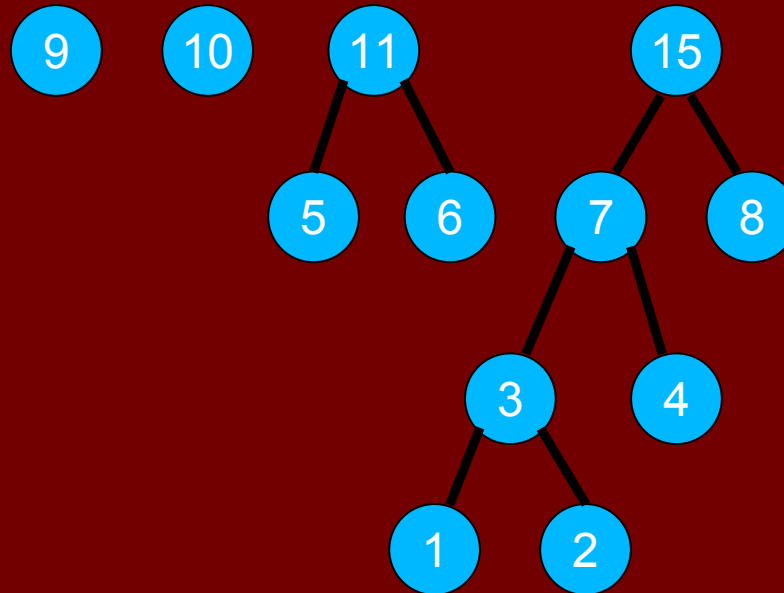
# Eksempel:



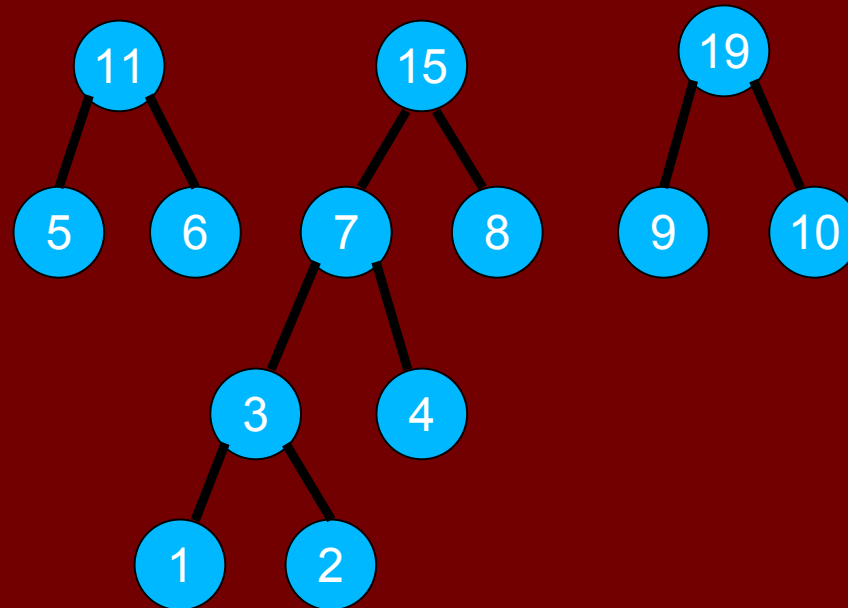
# Eksempel:



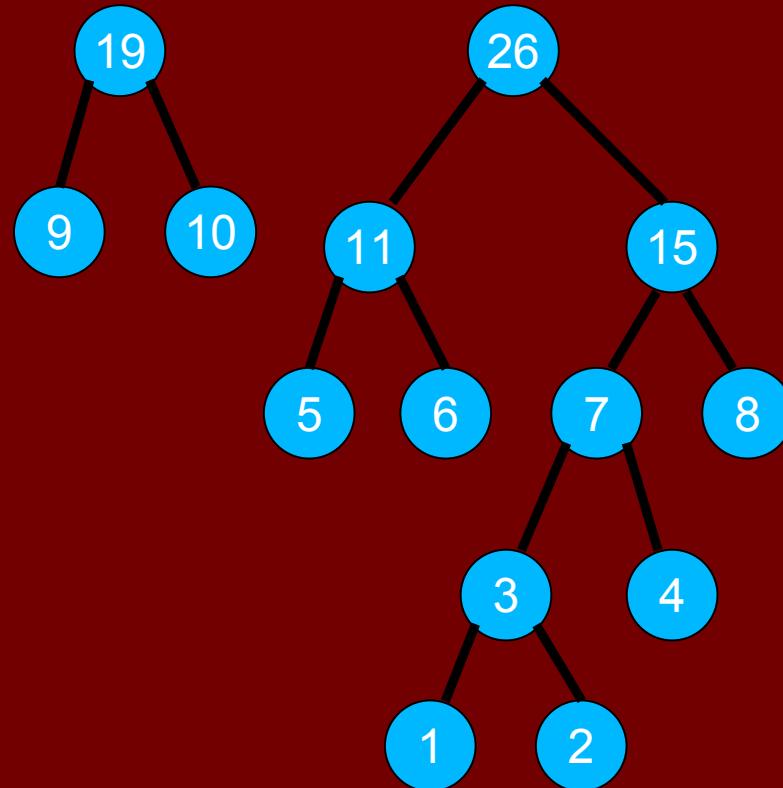
# Eksempel:



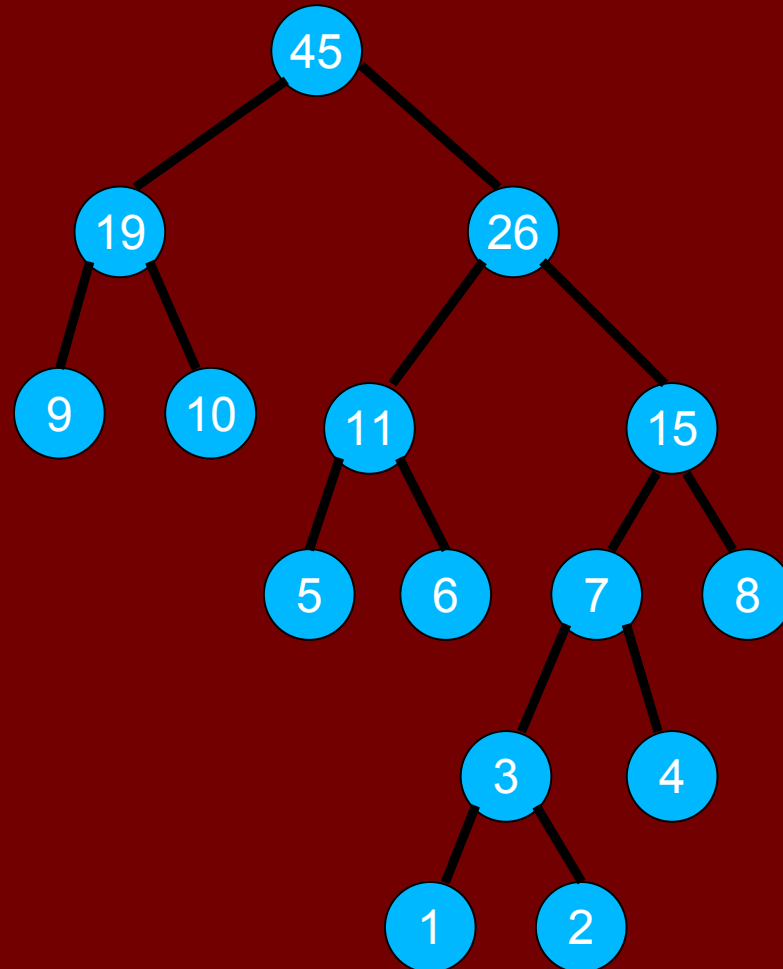
# Eksempel:



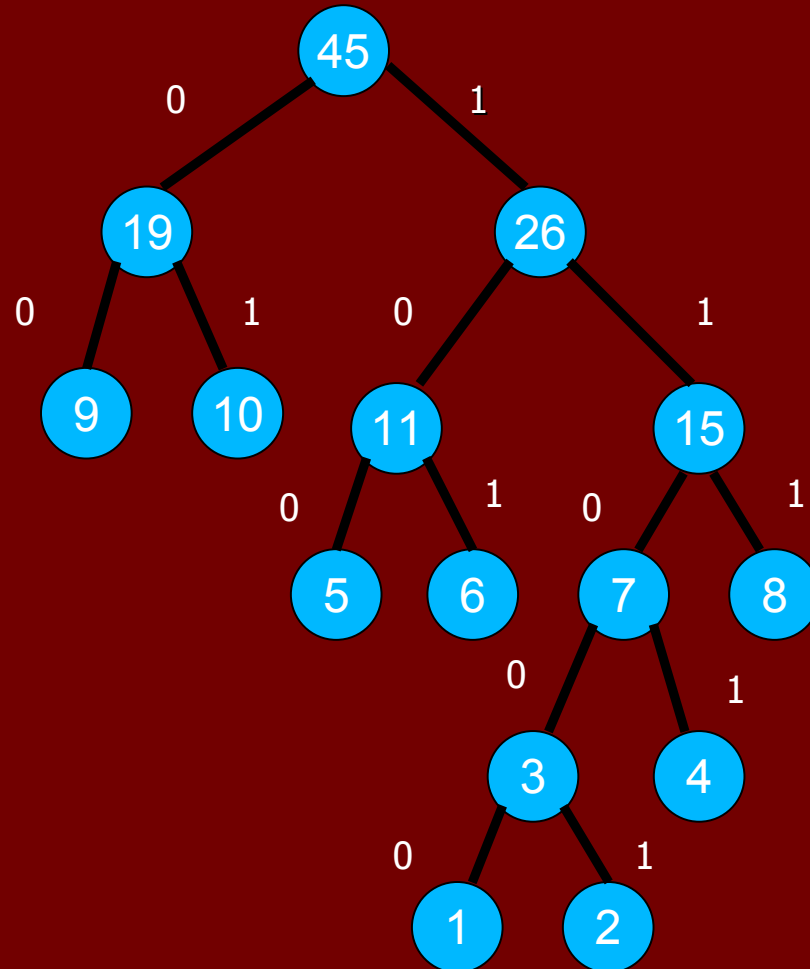
# Eksempel:



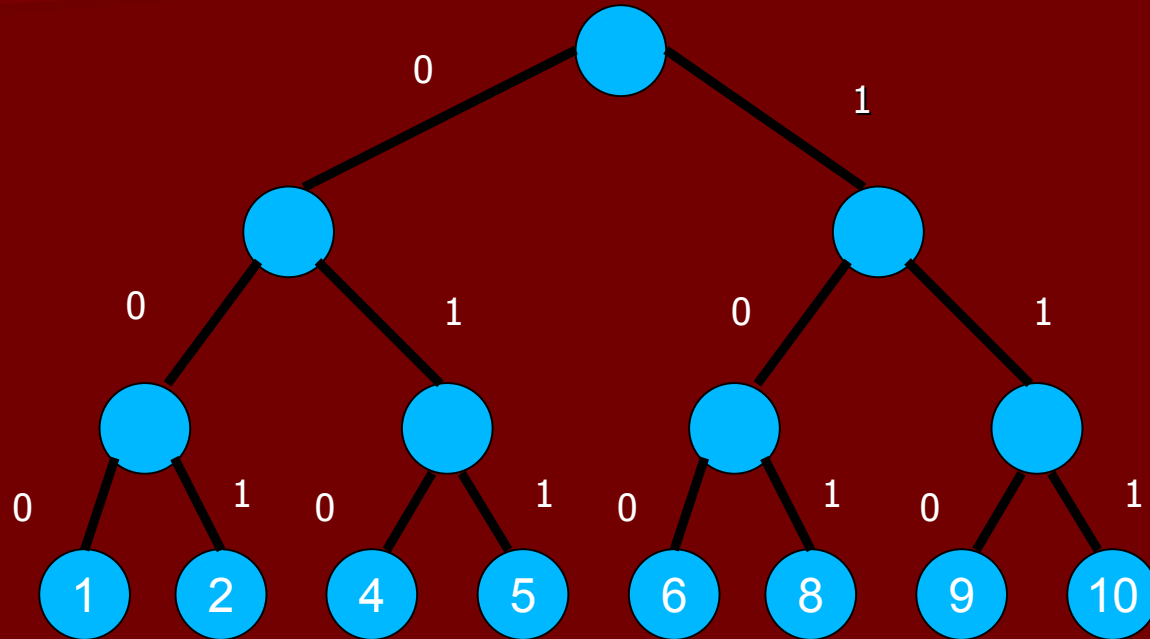
# Eksempel:



# Eksempel:



# Eksempel:



# Eksempel

- $B(\text{Fast})=3*45=135$
- $B(\text{Huffman})=126$
- $B(\text{Fast})-B(\text{Huff})=11$

# Snart er det Jul ;)

## Oppgave 6 (15%)

Student Lurvik skal summere  $n$  positive flyttall,  $x[1] \dots x[n]$ , slik at avrundingsfeilen blir minst mulig. Avhengig av hvordan dette gjøres, kan de ulike tallene delta i et ulikt antall summeringer. For eksempel vil  $x[1]$ ,  $x[2]$ ,  $x[3]$  og  $x[5]$  delta i flere summeringer (tre) enn  $x[6]$  (én) i følgende summeringsrekkefølge:

$$(((x[1] + x[3]) + (x[2] + x[5])) + x[6])$$

Merk her at både tall-rekkefølgen og parentes-settingen velges fritt.

Hvor stor avrundingsfeilen for en sum av to flyttall blir er avhengig av hvor stor summen er (større sum gir større feil). Lurvik innser at det kan være lurt å summere slik at de små flyttallene deltar i flest mulig summeringer (det vil si, de kommer "dypt" i parentes-settingen) mens de store flyttallene deltar i færrest mulig. Hvis  $p(i)$  er "parentes-dybden" til flyttall  $x[i]$  (antall parenteser utenfor elementet) så definerer Lurvik *feilen* til en mulig løsning som

$$p(1) \cdot x[1] + p(2) \cdot x[2] + \dots + p(n) \cdot x[n].$$

Lurvik ønsker nå å finne en parentes-setting som minimaliserer dette feiluttrykket. Skisser en løsning på problemet. Hvilken designmetode bruker du? Vis, med støtte i pensum, at løsningen er korrekt.