45011 Algoritmer og datastrukturer
Løsningsforslag eksamen 17. august 1995

Oppgave 1

Å løse denne oppgaven helt presist er vanskelig, men følgende løsning er god nok i eksamenssammenheng, selv om den ikke sjekker f.eks. løsninger av formen $f(n) = \log \log n$. Vi tar utgangspunkt i Masterteoremet og sjekker hvor stor $f(n)$ kan være.

1a) Vi har følgende rekurrensligning: $T(n) = T(n/2) + f(n)$, der $f(n)$ er tidskompleksiteten til `analyserA`. Vi har altså $a = 1$ og $b = 2$ slik at $n^{\log_b a} = 1$. De forskjellige tilfellene i Masterteoremet gir oss:

Tilfelle 1: Løsning T(n) = $\Theta(n^{\log_b a}) = \Theta(1)$ som er raskere enn kravet i oppgaven. Vi prøver derfor tilfelle 2:

Tilfelle 2: Krav $f(n) = \Theta(n^{\log_b a}) = \Theta(1)$ med løsning $T(n) = O(n^{\log_b a} \lg n) = \Theta(\lg n)$. Dermed kan `analyserA` bruke $f(n) = O(1)$ tid. Sjekker om tilfelle 3 gir en bedre løsning:

Tilfelle 3: Krav $f(n) = \Omega(n^{\log_b a+\epsilon}) = \Omega(n^{1+\epsilon})$ med løsning $T(n) = \Theta(f(n)) = \Omega(n^{1+\epsilon})$. Dette er for tregt for kravet i oppgaven.

1b) Vi har følgende rekurrensligning: $T(n) = 2T(n/2)+f(n)$, der $f(n)$ er tidskompleksiteten til `analyserB`. Vi har altså: $a = 2$ og $b = 2$, slik at $n^{\log_b a} = n$. De forskjellige tilfellene i Masterteoremet gir oss (tilfelle 1 som ovenfor):

Tilfelle 2: Krav $f(n) = \Theta(n^{\log_b a}) = \Theta(n)$ med løsning $T(n) = O(n^{\log_b a} \lg n) = \Theta(n \log n)$. Dermed kan `analyserB` bruke $f(n) = O(n)$ tid.

Tilfelle 3: Løsning $T(n) = \Theta(f(n))$ ville gi ønsket løsning dersom $f(n) = \Theta(n \lg n)$, men da er kravet om at $f(n)$ skal være polynomisk større enn $n^{\log_b a} = n$ ikke oppfyllt.

Oppgave 2

Bruk av median som pivot gir $O_w(n \cdot \log n)$ siden median kan finnes i lineær tid. Brukes ikke i praksis p.g.a. urimelig økning av konstant i faktisk kjøretid.

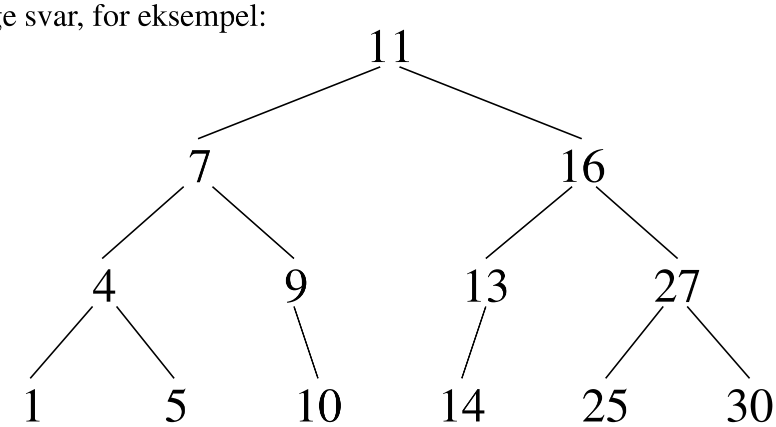
Oppgave 3

Tellesortering : $O(n + k)$; k = verdiområdet
Radixsortering : $O(d(n + k_1))$; d = antall siffer, k_1 = verdiområdet for et siffer

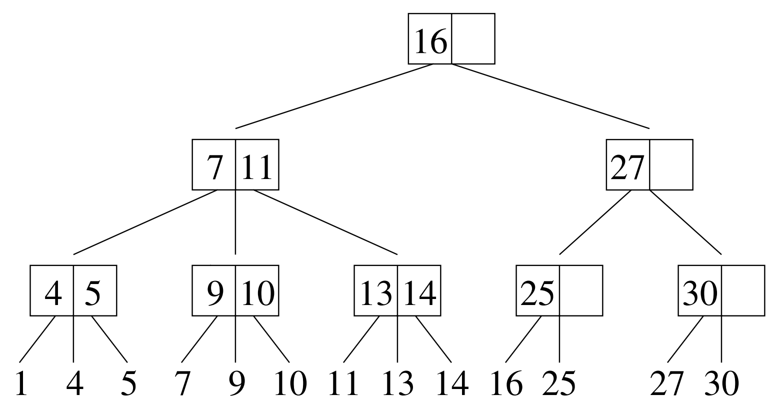
Velger den med lavest kompleksitet. Med $k >> n$ bør radixsortering velges.

Oppgave 4

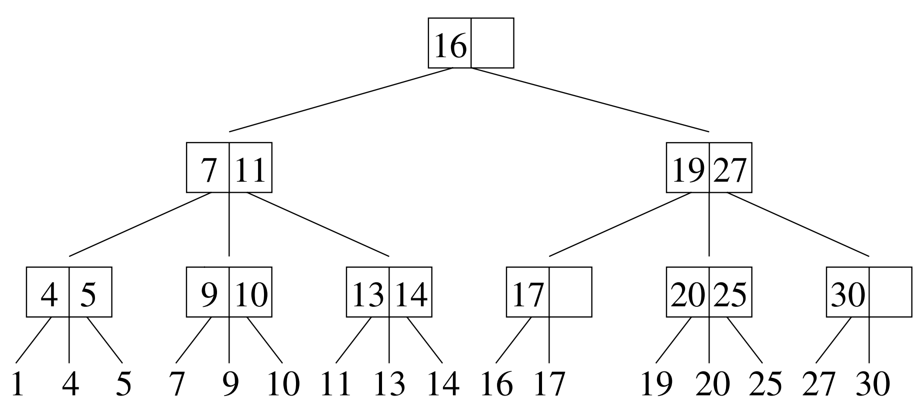
a) Mange riktige svar, for eksempel:



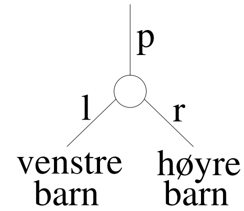
b)



c)



Oppgave 5



- Flyt$_p$ = Min (p,Flyt$_l$+Flyt$_r$) $\quad (*)$
- Definer Flyt$_l = \infty$ hvis venstre barn ikke eksisterer.
  Definer Flyt$_r = \infty$ hvis høyre barn ikke eksisterer.
- Postfikstraversering av treet med $(*)$ gir da svaret i O(n)-tid.
- Ford-Fulkerson kan også benyttes, men gir høyere kjøretid.

Oppgave 6

Lag tabell $E[1..n, 0..100]$ der $E[i, k]$ er den minste feilen for:
$(y[1] - x[1])^2 + (y[2] - x[2])^2 + ... + (y[i] - x[i])^2$ gitt at $y[i] = k$:

for $k = 0$ **to** 100 **do**
  $E[1, k] = (k - x[1])^2$;
for $i = 2$ **to** n **do**
  for $k = 0$ **to** 100 do

$$E[i, k] = (k - x[i])^2 + \min_{l \leq k} E[i - 1, l];$$

$y[n]$ er den k-verdien som minimaliserer $E[n, k]$. Man kan lage en tabell som brukes til tilbakesporing, men dette er ikke vist i koden.

Oppgave 7

Alle kan generaliseres. Rabin-Karp kan gjøres spesielt effektiv ved "separabel" beregning av nøkler.

Oppgave 8

Tallrekken representerer antall enere i binærtall beregnet etter stigende verdier
Rekurrensformel:

$$a_0 = 0;$$

$$a_j = a_{f(j)} + 1, \quad j = 1, 2, 3, ...$$

$$f(j) = j - 2^{\lfloor \log_2 j \rfloor}$$

Flere løsninger finnes, f.eks:

$$a_0 = 0;$$

For j like: $a_j = a_{\frac{j}{2}}$

For j odde: $a_j = a_{\frac{j-1}{2}} + 1$