45011 Algoritmer og datastrukturer
Løsningsforslag eksamen 12. januar 1993

Oppgave 1

a)
$n_0 = 5$ og $c = 2$
Begrunnelse:
$(n+2)^2 = n^2 + 4n + 4 \leq 2n^2$ for $n \geq n_0 = 5$

b)
Svar (Metode): PARTITION (=Quicksort én runde)
eller Tellesortering (COUNTING SORT).

Kjøretid: $O(n)$

c)

1. 2-3-trær er pr. definisjon balanserte, noe *ordinære* binære søketrær ikke er.
2. Høyde mellom $\log_2 n$ og $\log_3 n$, altså minst like godt som et balansert binært søketre.

d)

1. Sortering "på stedet".
2. Kan "bli ferdig" på $O(n)$ tid, som i 1b), mens Radix bruker $O(\text{antall sifre} \times n)$. ("semi-lineær").

e)
Forskjeller:

- Binær søking er lagret i array, mens søking i binærtre er pekerbasert.
- Binær søking er "alltid balansert", mens søking i binærtre normalt er ubalansert.
- Søking i binærtre er egnet for dynamisk inn & ut, mens binær søking ikke er det.

Likheter:

- Begge metoder er basert på 2-delt splitt & hersk.
- Begge metoder er egnet for effektiv gjenfinning i et ordnet (sortert) materiale.

Oppgave 2

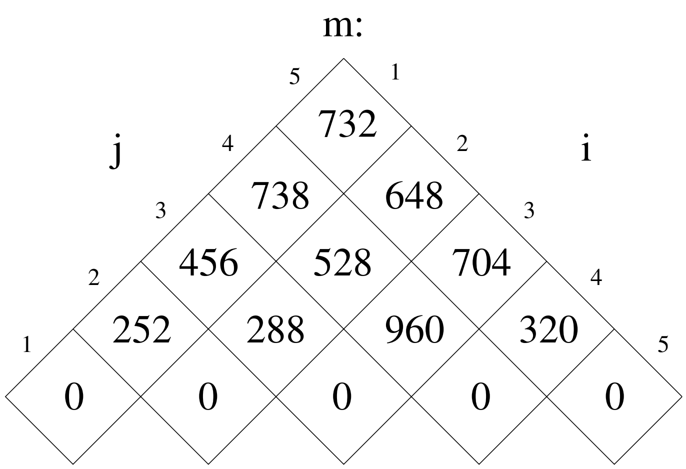
a)

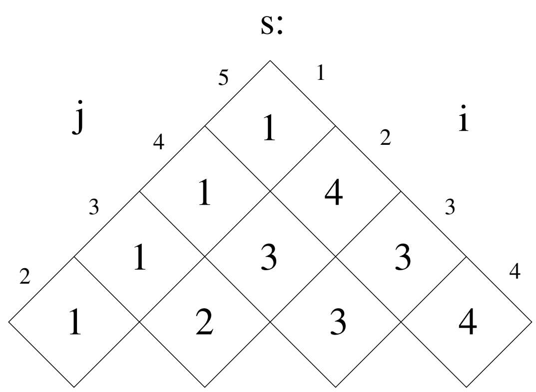
```
Procedure BuildHeap(A,i,n);
  begin
    if i ≤ n then
      BuildHeap(A,2i,n);
      BuildHeap(A,2i+1,n);
      Heapify(A,i);
  end;
```

b)
T(n) = 2T(n/2) + O(log n).

c)
T(n)=Θ(n)
ved “case” nr: 1.

Oppgave 3





Parentessetting: $A_1 \cdot (((A_2 \cdot A_3) \cdot A_4) \cdot A_5)$

Oppgave 4

a)

- Kjør flytalgoritmen én (kun én) gang etter at maksimal flyt er funnet.
- Siden “flytmaks” øker flytens verdi med minst en enhet, vil dette holde.

Merk: Dersom vi har “lett adgang” til snittet som skiller s og t og kanten (u, v) **ikke** er i dette snittet kan ingen høyere flyt oppnås.

b)

- Dersom 0-flyt på (u, v): Ingen forskjell.

  Ellers:

- “ send tilbake” 1 enhet langs flytforøkende vei fra t til v
- Tilsvarende fra u til s.
- Gjør som i a), og kjør flytalgoritmen én gang.

Oppgave 5

a)
$x = \max[S]$ og $y = \min[S]$
Begge operasjonene kan utføres med ett $O(n)$ gjennomløp av S

b)
Sorter S med Mergesort eller Heapsort (**ikke** Quicksort). Disse opererer uansett (worst-case) i $O(n\log n)$ tid. Gå deretter igjennom alle tallene og finn naboelementer x og y som har minst avstand mellom seg. Dette tar $O(n)$ tid.

c)
Definerer middeldistanse:

$$A[s..t]) = \frac{1}{n-1}(\max_{j\in[s..t]} A[j] - \min_{j\in[s..t]} A[j])$$

Ser at dette er høyresiden i likningen gitt i oppgaven for $s = 1$ og $t = n$.
Oppgaven går altså ut på å finne to tall x og y i mengden S som er slik at avstanden mellom tallene er mindre eller lik middeldistansen mellom tallene i S.

function FIND(A, p, q,min$i$,max$i$) : integer;
if $q = p + 1$ **then return** q **else**
  $m := \lceil \frac{p+q}{2} \rceil$;
  median := SELECT(A, p, q, m);
  splitt med PARTITION (som i QUICKSORT) rundt median (= $A[m]$);
  if (median - min$i$)$/(m - p + 1) <$ (max$i$ - median)/$(q - m + 1)$ **then**
    FIND(A, p, m,min$i$,median) **else**
    FIND(A, m, q,median,max$i$);

Kall: $k :=$ FIND($A, 1, n$,funnetmin,funnetmaks);
De søkte verdier er da:
$x = A[k + 1]$ og $y = A[k]$.

Kjøretid:
Prosedyren SELECT returnerer ubetinget det i-te minste elementet i $A[1..n]$ i $O(n)$ tid. PARTITION Bruker $\Theta(n)$ tid. Altså har FIND kjøretid:
$T(n) = \Theta(1) + O(n) + \Theta(n) + T(n/2) = T(n/2) + \Theta(n)$.