Eksaminatorie-timer

DM507: Algoritmer og datastrukturer

Løsninger

Uge 8 # 1 2021

Del A

Opgave 1 - Cormen et al. øvelse 7.1-1

Illustrer operationen af `Partition` på arrayet $\langle 13, 19, 9, 5, 12, 8, 7, 4, 21, 2, 6, 11 \rangle$.
Se Figur 1

Opgave 2 - Cormen et al. øvelse 7.1-2

Hvilken værdi q returnerer `Partition`, når alle elementer i arrayet $A[q \dots r]$ har samme værdi? Modificer `Partition`, så $q = \lfloor \frac{p+r}{2} \rfloor$ retuneres når alle elementer i arrayet er ens.

I dette tilfælde vil i inkrementeres hver iteration – i alt foretages $r-q$ iterationer[1]. Da i initialiseres til $i = p - 1$, så følger det

$$i = \underbrace{(p-1)}_{\text{start værdi}} + \overbrace{(r-p)}^{\text{\# iterationer}} = r - 1$$

i slutningen af `Partition`. Da `Partition` returnerer $q = i + 1$, så er $q = (r - 1) + 1 = r$; altså, q har værdien af det sidste indeks.

For hvert element lig med pivot x, skift mellem at placere den i $\leq$- og $\geq$-gruppen. Skifter man mellem $\leq$-gruppen $\rightarrow$ $\geq$-gruppen $\rightarrow \dots$, så returneres $q = \lfloor \frac{p+r}{2} \rfloor$, når alle elementer er ens. Korrektheden af quicksort ændrer sig ikke, da man placerer pivot korrekt og skaber to delproblemer, som kan løses rekursivt (dog kan begge grupper/delproblemer have elementer lig pivot). I nedenstående pseudokode opnås dette skift mellem $\leq$- og $\geq$-gruppen vha. c counteren. Når c er lige ved tjekket, så bruges $\leq$-gruppen.

[1] Der foretages ikke $r - q + 1$ iterationer, da vi stopper når $j = r - 1$ og ikke når $j = r$.

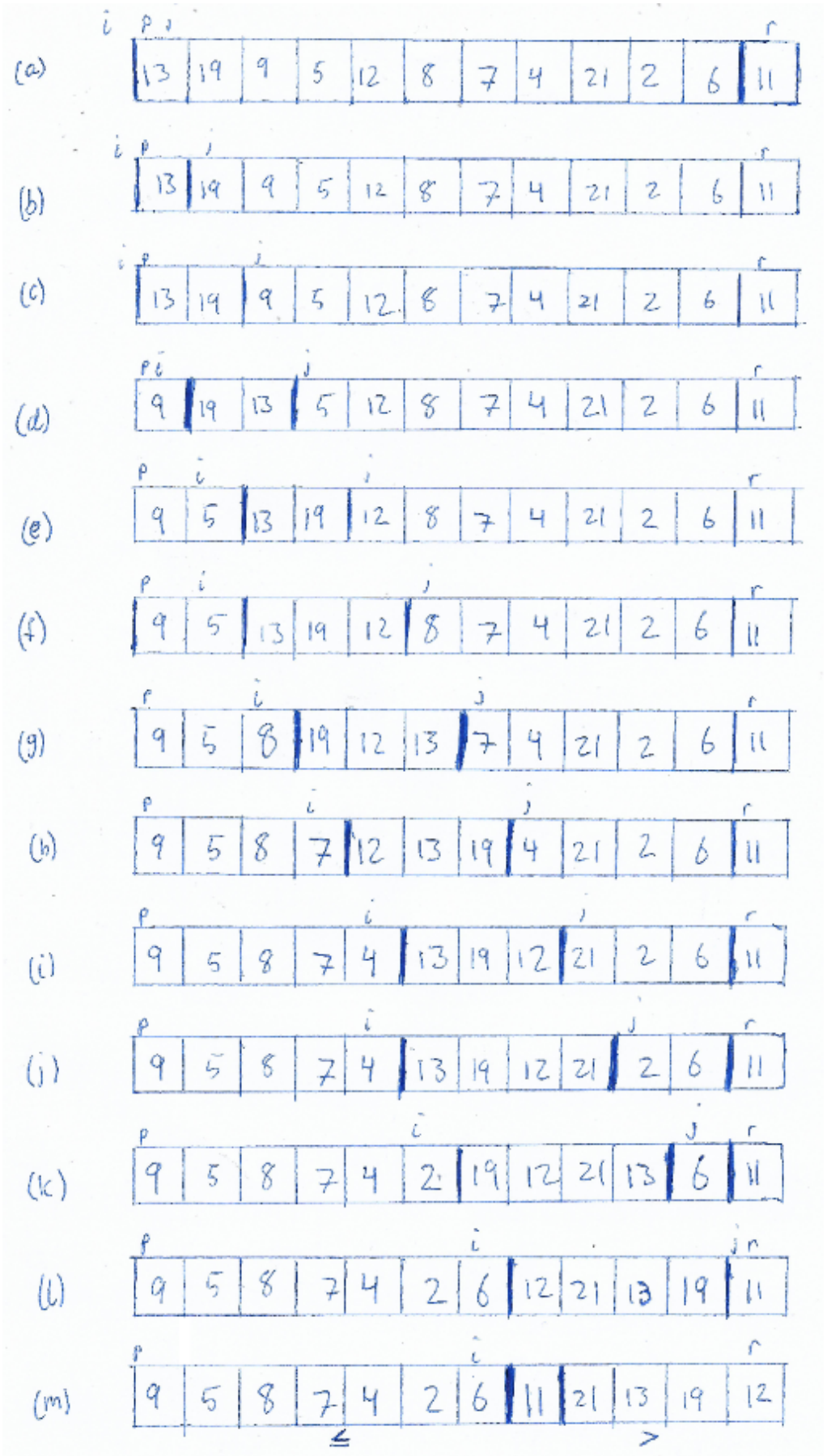


Figure 1: Udførsel af partition.

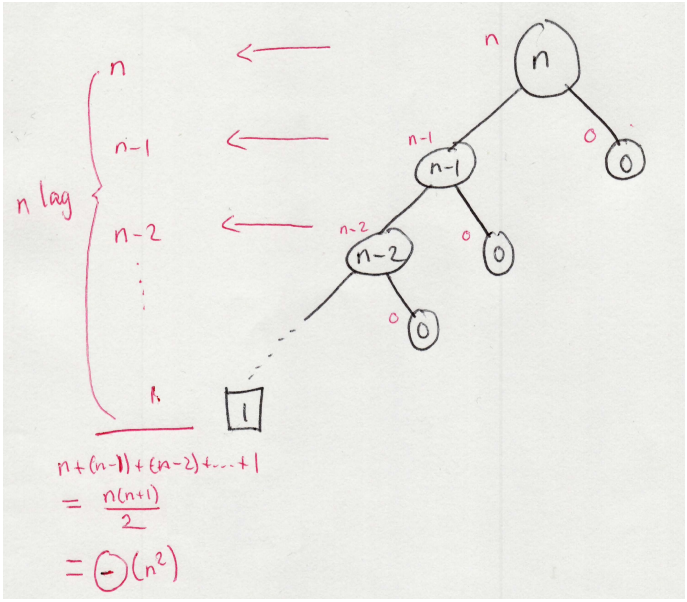
```
Partition'(A, p, r)
  x = A[r]
  i = p - 1
  c = 0
  for j = p to r - 1
    if A[j] < x
      i = i + 1
      exchange A[i] with A[j]
    else if A[j] == x
      c = c + 1
      if c % 2 == 0
        i + 1
        exchange A[i] with A[j]
  exchange A[i+1] with A[r]
  return i + 1
```

Opgave 3 - Cormen et al. øvelse 7.2-2

Hvad er køretiden af Quicksort (antag bogens udgave a partition), når alle elementer i et array A har samme værdi?

Partition retunerer $q = r$ hver gang; altså, der skabes to partitioner af størrelse 0 og $n - 1$. Når alle partitioner er helt ubalancerede, så er køretiden (se figur):

$$n + (n-1) + (n-2) + \cdots + 2 + 1 = \frac{n \cdot (n+1)}{2} = \Theta(n^2)$$



Opgave 4 - Cormen et al. øvelse 7.2-3

Vis at køretiden for Quicksort er $\Theta(n^2)$, når arrayet A indeholder unikke elementer og er sorteret i enten stigende eller faldende orden.

Stigende: Det valgte pivot x – det sidste element i arrayet – vil være det største element (pga. stigende orden). Alle elementer ender i $\leq$-gruppen, og $>$-gruppen er tom. Der er altså et dårligt split – to delproblemer af størrelse $n-1$ og 0. Delproblemet med størrelse $n-1$ er også stigende sorteret; dårligt split gentager sig hver gang. Dette giver en $\Theta(n^2)$ køretid (se evt. Opgave 3).

Faldende: Der opstår to cases:

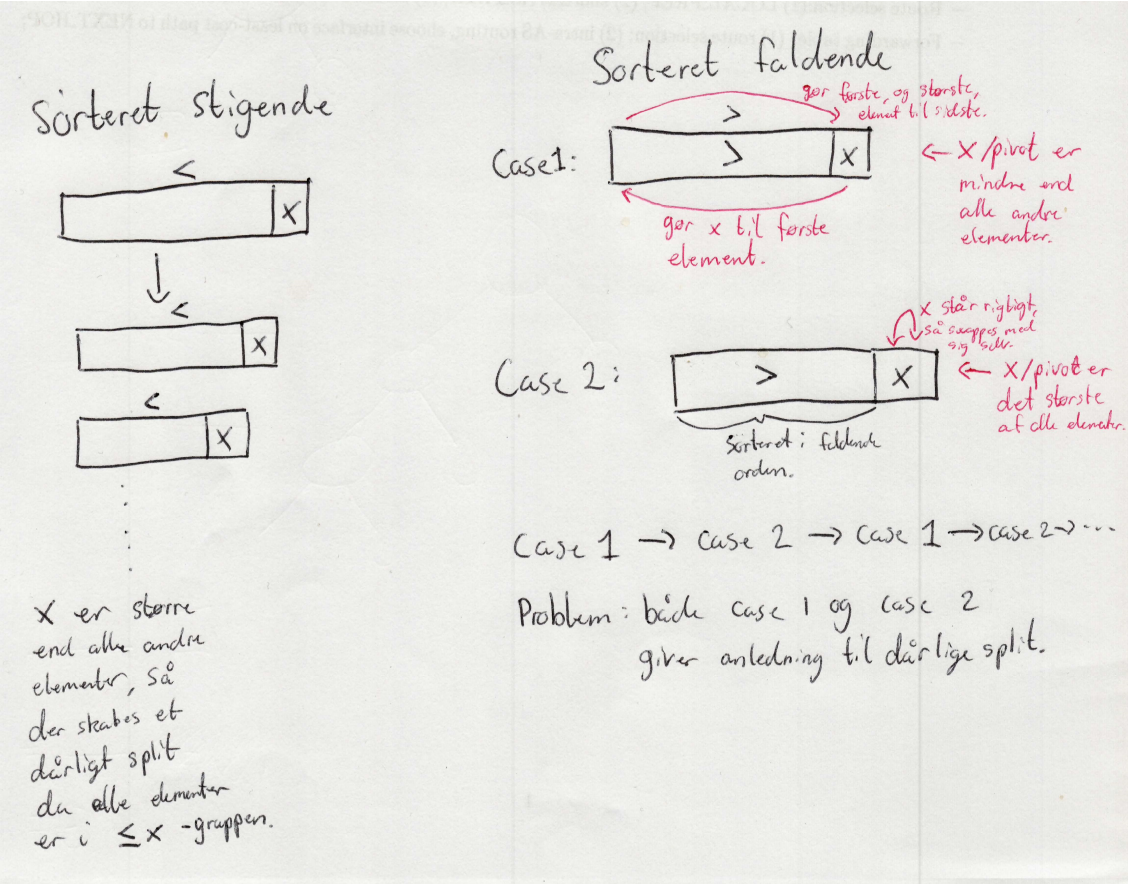
- Case 1 (hele arrayet sorteret i faldende orden): Det valgte pivot x – det sidste element – vil være det mindste element. Alle elementer ender i $>$-gruppen, og $\leq$-gruppen er tom. Der er altså et dårligt split – to delproblemer af størrelse $n-1$ og 0. Pivot x byttes med første og største element i arrayet. Case 2 opstår for $n-1$ delproblem.
- Case 2 (sidste element er det største element, og resten er sorteret i faldende orden): Det valgte pivot x - det sidste element - vil være det største element. Alle elementer ender i $\leq$-gruppen, og $>$-gruppen er tom. Der er altså et dårligt split – to delproblemer af størrelse $n-1$ og 0. Pivot x byttes med sig selv. Case 1 opstår for $n-1$ delproblem.

Når input arrayet er sorteret i faldende orden skiftes der mellem Case 1 og Case 2:

$$\text{Case 1} \rightarrow \text{Case 2} \rightarrow \text{Case 1} \rightarrow \text{Case 2} \rightarrow \ldots$$

Der er altså et dårligt split i både Case 1 og Case 2 (kun dårlige split foretages). Dette giver en $\Theta(n^2)$ køretid (se evt. Opgave 3).

Note: Se evt. en skitse nedenunder (teksten skrevet med dårlig håndskrift siger ikke mere end teksten overover)



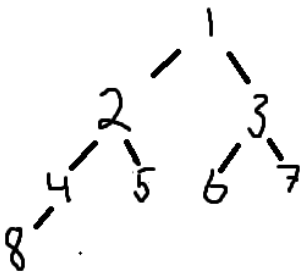
Opgave 5 - Cormen et al. øvelse 6.1-5

Er et sorteret (ikke-falende) array en min-heap?

Ja. Antag til modstrid et sorteret array ikke er en min-heap. I så fald eksisterer der et barn med en forældre, som har en større key. Da forældren er på et lavere indeks, så må værdien dog være højst barnets key, da arrayet er sorteret ↯. Et sorteret array må være en min-heap.

Betragt evt. følgende:

$$\underbrace{\langle 1, 2, 3, 4, 5, 6, 7, 8 \rangle}_{\text{binært træ heap fortolkning}}$$



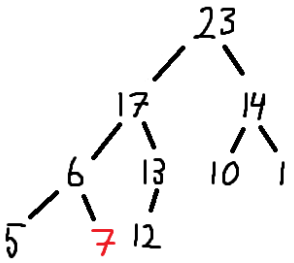
Note: En min-heap er ikke nødvendigvis et sorteret array:

$$\text{sorteret array (ikke-falende)} \Rightarrow \text{min-heap} \not\equiv \text{min-heap} \Rightarrow \text{sorteret array (ikke-falende)}$$

Opgave 6 - Cormen et al. øvelse 6.1-6

Er følgende en max-heap?

$$\underbrace{\langle 23, 17, 14, 6, 13, 10, 1, 5, 7, 12 \rangle}_{\text{binary tree heap interpretation}}$$



Nej - det er ikke en max-heap.

Note: Byt rundt på 6 and 7 for at opnå en max-heap. Eller kald `Max-Heapify([...])` på index 4.

1 Del B

Opgave 1 - Eksamen juni 2008, opgave 1 b

Angiv best-case og worst-case køretid samt køretiden for sorteret input for Insertion Sort, Merge Sort og Quicksort.

	Best-case	Worst-case	Sorteret input
Insertion sort	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n)$
Merge sort	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n \log n)$
Quicksort	$\Theta(n \log n)$	$\Theta(n^2)$	$\Theta(n^2)$

- Insertion sort: Husk tilbage til tidligere opgaver. Best-case er når den er sorteret (stigende), og her er den hurtigt færdig, da der er ingen inversioner at rette. Worst-case er input sorteret i faldende orden, da der her er det maksimale antal inversioner, og køretiden bliver $\Theta(n^2)$.
- Mergesort: Uanset input, så laver vi samme arbejde; altså er køretiden $\Theta(n \log n)$.
- Quicksort: Fra slides ved vi worst-case er $\Theta(n^2)$, og vi har argumenteret for køretiden for sorteret input i tidligere opgaver. Best-case ved vi fra slides er $\Theta(n \log n)$.

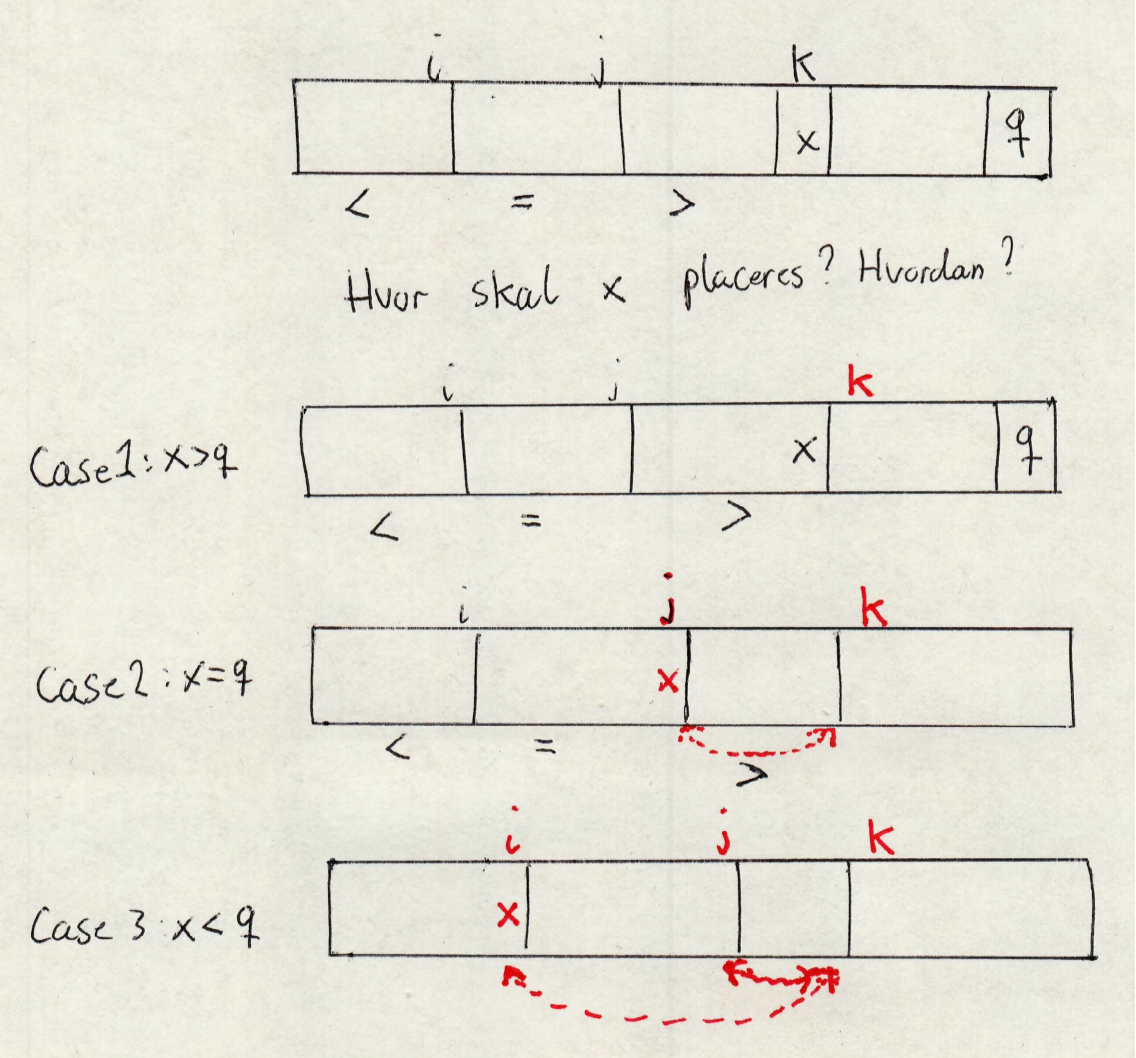
Opgave 2 - Cormen et al. problem 7-2, spørgsmål b

Modificér `Partition` proceduren, så den returnerer to indekser q og t, hvor $p \leq q \leq t \leq r$, sådan at

- alle elementer i $A[q..t]$ er ens
- hvert element af $A[p..q-1]$ er mindre end $A[q]$
- hvert element af $A[t+1..r]$ er større end $A[q]$.

Ligesom `Partition`, så skal `Partition'` bruge $\Theta(r-p)$ tid.

Se nedenstående pseudokode (og evt. skitsen):




```
Partition'(A, q, r)
  x = A[r]
  i = p - 1
  j = p - 1
  for k = p to r - 1
    if A[k] < x // Case 3
      i = i + 1
      j = j + 1
      exchange A[i] with A[k]
      exchange A[j] witj A[k] // now A[k] has value which initially was A[i]
    else if A[k] == x // Case 2
      j = j + 1
      exchange A[j] witj A[k]
    // Case 1: Increment k; otherwise do nothing
  exchange A[j+1] with A[r]
  return (i + 1, j)
```

Svar derefter igen på de to sidste øvelser:

- Opgave 3 - Cormen et al. øvelse 7.2-2 (alle elementer er ens): Første partition samler alle elementer i subarrayet $A[q..t]$; de to arrays $A[1..q-1]$ og $A[t+1..r]$ er tomme. Altså, der skabes to delproblemer af størrelse 0. Køretiden er nu $\Theta(n)$, da partition kun behøves blive

kørt én gang.

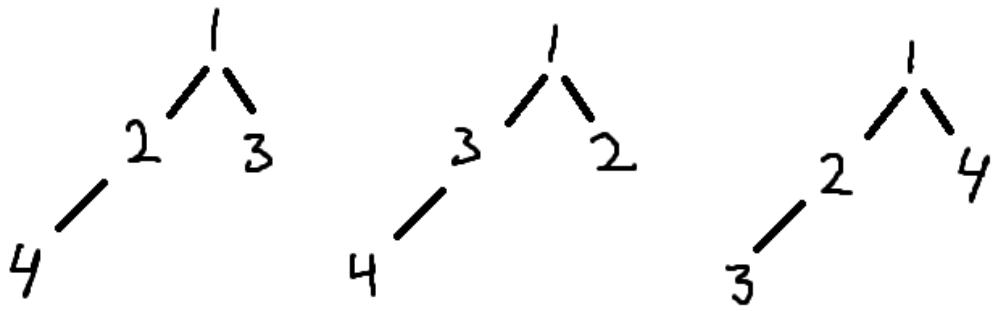
- Opgave 4 - Cormen et al. øvelse 7.2-3:
  - Stigende sorteret: $\Theta(n^2)$ køretid - samme problem som tidligere med dårlige split.
  - Faldende sorteret: $\Theta(n^2)$ køretid - samme problem som tidligere med dårlige split.

Opgave 3 - Eksamen juni 2008, opgave 4 a

Tegn alle mulige binære min-heaps, som indeholder fire knuder med prioriteterne 1, 2, 3 og 4. Observér følgende:

- Det er en heap, så det bineære træ skal have heap facon.
- Det er en min-heap, så det mindste element skal være i roden (dvs. 1).
- Endvidere, så kan 4 ikke have nogen børn, da den er større end 1, 2 og 3.

Følgende kombinationer er gyldige min-heaps:



Note: Andre kombinationer eksisterer, men disse ville gå imod heap-facon egenskaben (se Rolfs slides under "Heap facon").

Opgave 4 - Cormen et al. øvelse 6.1-4

Hvor i en max-heap kan det mindste element være, når det antages alle elementer er unikke.

Det mindste element vil være i et blad. Antag til modstrid, at det mindste element ikke er i et blad. Da det er en max-heap (med unikke elementer), så skal der være max-heap orden, men i så fald kan det mindste element ikke have et barn, da max-heap ordnen vil være brudt ↯. Altså, så skal det mindste element være i et blad.

Ekstra: Kan det være i hvilket som helst blad? Ja! For alle blade i en max-heap af størrelse n, så findes der en udgave, hvor det mindste element er på et givent blad. Du kan altid bytte det mindste element ud med et andet blad, og rette eventuelle problemer vha. `Max-heapify`. Du kan også indsætte elementer i et array og sortere arrayet i faldende orden. Nu kan du bytte alle blade med hinanden, da forældren til elementerne efter byttet altid vil være større pga. de er sorteret.