Aflevering 1
Algoritmer og datastrukturer
Forår/Sommer 2011

Naja Mottelson (vsj465)
Søren Pilgård (vpb984)

6. maj 2011

Indhold

1 a

```
Hoare-partition(A,p=1,c=)
x = 13
i = -1
j = 13

Step 0:
i = -1, j = 13

     1  2  3  4  5  6  7  8  9 10 11 12
... ___________________________________ ...
:  |13|19| 9| 5|12| 8| 7| 4|11| 2| 6|21|  :
:..|__|__|__|__|__|__|__|__|__|__|__|__|..:
 |->i                              j<----|
i = 1, j = 11
SWAP(A[1], A[11])



Step 1:
i = 1, j = 11

     1  2  3  4  5  6  7  8  9 10 11 12
    ___________________________________
   | 6|19| 9| 5|12| 8| 7| 4|11| 2|13|21|
   |__|__|__|__|__|__|__|__|__|__|__|__|
    |->i                        j<-|

i = 2, j = 10
Swap(A[2], A[10])


Step 2:
i = 2, j = 10

     1  2  3  4  5  6  7  8  9 10 11 12
    ___________________________________
   | 6| 2| 9| 5|12| 8| 7| 4|11|19|13|21|
   |__|__|__|__|__|__|__|__|__|__|__|__|
       |                     j<-|
       |---------------------->i

i = 10, j = 9
return 9
```

2 b

Vi starter med at gennemgå Hoares-Partitioneringsalgoritme ret grundigt.

Vi ser at i og j initialiseres som positioner udenfor arrayret A[p..r] -hhv. som p-1 og r+1. Grundet repeat-until konstruktionen vil de dog blive in/dekrementeret inden der laves opslag i A.

2.1 Første iteration:

Første indre løkke (linje 5-7): Vi ser at j dekrementeres ved indgangen til den første indre løkke. j ligger nu indenfor arrayet A[p..r]. Såfremt A[j] ¡= x stopper den indre løkke (linje 7), ellers fortsætter vi med at at dekrementere j. vi ved at j aldrig bliver mindre end p, da A[p] = x. Den indre løkke vil altså altid stoppe med p ¿= j ¡= r samt A[j] ¡= x. Derudover ved vi at der ikke kan være nogen elementer i A[j+1..r] der er mindre/lig med x.

Anden indre løkke (linje 8-10): Ved indgangen til den anden indre løkke inkrementeres i, så i=p og da A[i] = A[p] = x vil den indre løkke stoppe her.

Efter de indre løkker (linje 11-13) Efter de to indre løkker sammenlignes i og j. Enten har j bevæget sig hele vejen ned igennem arrayet så j=i=p hvorefter vi ved at alle elementer er større end x. Der skal derfor ikke gøres mere da x er det mindste element og ligger forrest. Her vil funktionen terminere.

Hvis j istedet er forskellig fra i, må j nødvendigvis være større end i (j kan på nuværende tidspunkt ikke være mindre end i da A[i] = x). Vi ved så at A[i] = x og at j er stoppet på et element så A[j] ¡= x. Derfor vil A[i] og A[j] blive ombyttet. Efter første iteration gælder derfor at A[j] er et element der er mindre eller lig med x samt at A[i] = x. Derudover ved vi at alle elementer i A[j+1 .. r] er større end x. Mere generelt kan vi sige at alle elementer i A[p..i] er mindre eller lig x samt at alle elementer i A[j .. r] er større eller lig med x. Vi kan konkludere at vi på intet tidspunkt i første iteration vil tilgå et element der ikke findes i A[p..r].

2.2 Efterfølgene iterationer:

Vi ved nu at der i intervallet A[p..i] kun eksiterer elementer der er mindre/lig x samt at der i intervallet A[j..r] kun eksisterer elementer der er større/lig x i hver iteration opretholder vi denne orden samtidig med at vi dekrementerer j og inkrementerer i.

Vi ved vi opretholder denne orden for i hvert trin i den første indre løkke lader vi j glide ned gennem arrayet mod p indtil den finder et element A[j] ¡= x der stopper den j, da den kun bevæger sig hen over elementer der er ¿= x ved vi at alle elementer i A[j+1..r] ¿= x efterfølgende lader vi i gå mod r på samme måde som j. Hvis j møder et element der er ¿= x stoppes der. Vi ved derfor ligesom med j at elementerne i A[p..i-1] ¡= x. Når j og i er stoppet ved vi de skal ombyttes (med mindre i¿j hvormed vi skal stoppe funktionen som set under afslutningen). Efter ombytningen ved vi at A[j] ¿= x og at A[i] ¡= x. Vi har derfor at elementerne i A[p..i] ¡= x og A[j..r] ¿= x, samt at både j og i ligger mellem p og r da de stadig er på vej mod hinanden.

2.3 Afslutning:

På et tidspunkt vil j og i bevæge sig forbi hinanden eller lande på samme element. Vi ved at A[p..i] efter første iteration indeholder mindst et element, samt at elementerne er mindre/lig x. Derfor vil den første indre løkke stoppe hvis j ¡= i da A[j] så vil være ¡= x

Tilsvarende ved vi at A[j..r] indeholder mindst et element samt at elementerne er større/lig x. Derfor vil den anden indre løkke stoppe hvis i ¿= j da A[i] så vil være ¿= x

Hvis bãde j og i stopper på det samme element, så j=i, må elementet være lig med x. Vi ved også algoritmen har traverseret hele arrayet A[p..(ij)..r]. Vi har så at alle elementerne A[p..i-1] ¡= x og at A[j+1..r] ¿=x samt at A[i]=A[j]=x. Vi kan samtidigt konkludere at både i og j ikke har bevæget sig uden for arrayet da de har bevæget sig mod hinanden ind mod midten og stoppet på samme position.

Hvis ikke j og i stopper på samme plads vil de krydse hinanden. Der kan nu ske en af to ting, j krydser i eller i krydser j

Det første scenarie vil forekomme når j rammer i, da vil j stoppe for vi har at A[i] ¡= x. Efter j er stoppet vil i rykke til j+1 hvor i også vil stoppe da A[i] så er ¿= x. Det andet scenarie vil forekomme når j stopper på et tal der er ¡= x, herefter bevæger i sig så forbi j og lander på j+1 hvor det som før vil ende. I begge scenarier har vi at invarianten ikke længere gælder for de nye værdier i og j da de nu har krydset hinanden. Vi ved dog stadig at alle elementer i arrayet A[p..i-1] ¡= x, vi ved også at i = j+1 derfor må der gælde at alle elementer A[p..j] ¡= x. Desuden ved vi at elementerne A[j+1..r] stadig må være ¿= x Funktionen har derfor udført sin opgave og terminere.

Udfra denne dybdegående gennemgang ser vi altså følgende:

- j og i starter udenfor intervallet p til r, men bliver de/inkrementeret som det første i de indre løkker så opslagene i A er gyldige.
- I første trin rykkes j mod p indtil det finder et element der er mindre eller lig med x
- Vi ved at dette altid vil lykkedes da x = A[p]
- i inkrementeres med 1 og A[i] byttes ud med A[j]
- Der ligger nu en værdi i hver ende af arrayet hvor j og i altid vil stoppe.
- Hvis i og j krydser hinanden vil de på et tidspunkt møde en værdi der får dem til at stoppe og funktionen vil terminere da i ¿= j
- Det vistes at den ydre løkke afsluttes enten når i=j eller når i=j + 1
  (I opgave c gennemgås hvordan j altid vil være skarpt mindre end r hvormed j+1 ikke vil falde uden for p til r)

3 c

Først vil vi komme frem til at j vil være skarpt mindre end r når funktionen terminerer. Vi ser først på den første iteration af den yderste løkke. Den første indre løkke vil altid blive kørt mindst to gange: Ved første iteration bliver j dekrementeret til at være lig r.

- Hvis A[j] er større end x vil j fortsætte imod p. j vil derfor blive skarpt mindre end r ved anden iteration af den indre løkke.
- Hvis A[j] er mindre eller lig med x, vil der ske en ombytning af A[i] og A[j] (da i ¡ j). Da den ydre løkke ikke stoppes efter en ombytning vil j blive dekrementeret endnu engang inden funktionen kan terminere j vil derfor være skarpt mindre end r når funktionen er tilendebragt.

Vi ser nu hvorfor j kan være lig p. Dette sker når x værdien er den laveste af værdierne i A[p..r] og starter forrerst, så vil den første indre løkke løbe hele vejen ned til j=p hvor funktionen så terminere uden nogen ombytninger.

Hvis j undervejs mod p støder på en værdi A[j] ¡= x, begynder vi at inkrementere i til A[i] ¿= x, når dette sker ved vi fra 2.2 at elementerne A[p..i] ¡= x. Da ved vi at vi på et tidspunkt vil komme til en afslutning som set i 2.3 hvor j ¿ p

Det er derfor klart at p ¡= j ¡ r

4 c

p ¡= j ¡ r Vi ved at j altid vil være mindre end r når funktionen terminerer. Dette skyldes at den første indre løkke altid vil blive kørt mindst to gange: Ved første iteration bliver j sat til r.

Der kan ske en af to ting, enten vil j være forskelig fra i hvorved der vil ske en ombytning

I første trin: elementet mindre=, så skal der byttes, j != i Der udføres en ekstra it. hvorved j ¡ r

eller j møder et element der er større så fortsætter j mod p og er ¡r

Dernæst skal vi vise j ¡p og at j kan ende i p (=)

5 d

Fra opgave b, sektion 2.1, ved vi at efter første iteration vil det gælde at elementerne i A[p..i] er ¡= x samt at elementerne i A[j..r] er ¿= x

vi ved også fra sektion 2.2 at dette er gældene som j går mod p og i går mod r.

I sektion 2.3 så vi at V̈i ved dog stadig at alle elementer i arrayet A[p..i-1] ¡= x, vi ved også at i = j+1 derfor må der gælde at alle elementer A[p..j] ¡= x. Desuden ved vi at elementerne A[j+1..r] stadig må være ¿= x Funktionen har derfor udført sin opgave og terminere. S̈å efter funktionens kørsel ved vi at elementerne i A[p..j] er mindre eller lig med x-værdien og at elementerne i A[j+1..r] er større eller lig x. Det ses da at alle værdierne i A[p..j] er mindre eller lig værdierne i A[j+1..r]

6 e

```
QUICKSORT (A, p, r)
```

```
if p < r
       j = HOARE-PARTITION (A, p, r)
       QUICKSORT (A, p, j)
       QUICKSORT (A, j + 1, r)
```

7 GT

```
        Lavet af:

    nmmmmn       \||||\
   (( oo ))      | oo |
   (\_ =_/)      \_ -_/
  /))_)(_((\      _||_
    /    \       /    \
   //|  |\\     //|  |\\
  // |  | \\   // |  | \\
  m /    \ m   m  |__|  m
   /~~~~~~~\      | ||
     || ||        | ||
     || ||        | ||
    _|| ||_      _| ||__
   /_/| |\_\    |___|___|
     Naja    og   Søren
```