

Introduktion til algoritmik og datastrukturer

IT-højskolen i København

13. juni 2001

Dette eksamenssæt består af 3 opgaver med i alt 13 delopgaver. De 13 delopgaver vægtes ens i bedømmelsen. Du har i alt 4 timer til din rådighed. Husk at angive sidetal, navn og cpr.-nummer på alle sider i din besvarelse. Eksamenssættet består af 7 nummererede sider.

CLR refererer til "Introduction to Algorithms" af Cormen, Leiserson og Rivest, 18. tryk, 1997.

I besvarelser, hvor der skal angives effektive algoritmer, lægges der i bedømmelsen vægt på den beskrevne løsnings asymptotiske tidskompleksitet. I spørgsmål, hvor der skal angives tidskompleksitet, skal denne udtrykkes i O -notation med mindst mulig vækstrate.

Opgave 1

Denne opgave handler om O -bestemmelse.

a) Hvad er $O(\log n) + O(1) + O(n)$?

Lad der være givet en hob, som understøtter følgende operationer BUILD-HEAP, EXTRACT-MIN samt DECREASE-KEY.

b) Antag at hoben angivet ovenfor understøtter EXTRACT-MIN i tiden $O(\log \log n)$, DECREASE-KEY i $O(1)$ tid samt BUILD-HEAP i lineær tid. Angiv tidskompleksiteten for Dijkstra's algoritme i en graf med m kanter og n knuder såfremt denne hob bruges.

Betragt følgende metode :

$F1(n)$

1 **if** $n > 2$

2 **then return** $(F1(n - 1) + F1(n - 2)) \pmod n$

3 **else return** 2

c) Forklar hvorledes $F1$ kan implementeres, så den har tidskompleksiteten $O(n)$.

Betragt følgende tre metoder :

F2(n)

```
1 if  $n > 0$ 
2   then F2( $\lfloor n/2 \rfloor$ )
3     F2( $\lfloor n/2 \rfloor$ )
```

F3(n)

```
1 if  $n > 0$ 
2   then for  $i \leftarrow 1$  to  $n$ 
3     do  $j \leftarrow 1$ 
4       F3( $\lfloor n/2 \rfloor$ )
5       F3( $\lfloor n/2 \rfloor$ )
```

F4(n)

```
1 if  $n > 0$ 
2   then for  $i \leftarrow 1$  to  $n$ 
3     do  $j \leftarrow 1$ 
4       F4( $\lfloor n/2 \rfloor$ )
```

d) Antag at ovenstående procedurer kun bliver kaldt med et heltal n . For hver af de fire ovenstående procedurer skal du angive tidskompleksiteten udtrykt i n .

Opgave 2

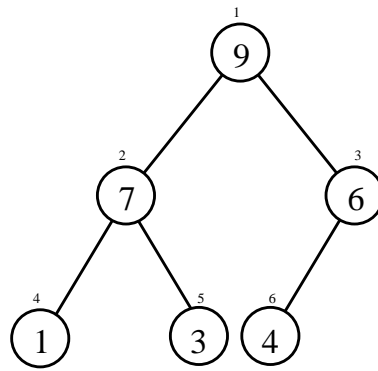
Denne opgave handler om hobe og amortisering. En hob er defineret som en datastruktur, der understøtter operationerne EXTRACT-MAX og INSERT.

INSERT(A, k) : Indsætter værdien k i hoben A .

EXTRACT-MAX(A) : Fjerner og returnerer en maksimal værdi fra hoben A .

Lad disse to operationer være implementeret som beskrevet i CLR, kapitel 7.

a) Illustrer udførelsen af EXTRACT-MAX(A) for hoben A i figur 1. Anvend stilen fra figur 7.5 i CLR side 151. Bemærk, at i denne figur vises en indsættelse, du skal illustrere udtagelse.



Figur 1: Hoben A , som EXTRACT-MAX skal udføres på.

Hoben skal nu understøtte en anden operation:

REMOVELARGEST(A, k) : Fjerner de k største elementer fra A . Hvis A har færre end k elementer tømmes A helt.

b) Antag at hoben kan indeholde op til n elementer. Beskriv hvordan REMOVELARGEST(A, k) kan understøttes i worst-case tid $O(k \log n)$.

Hoben skal nu igen understøtte en anden operation :

EXTRACTMORETHAN(A, x) : Fjerner og returnerer alle de værdier fra hoben A , som er større end x .

c) Antag at hoben kan indeholde op til n værdier. Beskriv hvorledes operationerne INSERT, EXTRACT-MAX og EXTRACTMORETHAN kan implementeres, så hoboperationerne har amortiseret tidskompleksitet $O(\log n)$. Det vil sige, at m hoboperationer udføres i tiden $O(m \log n)$

Hoben skal nu understøtte en anden operation :

EXTRACT-MIN(A) : Fjerner og returnerer en minimal værdi fra hoben A .

d) Antag at hoben kan indeholde op til n værdier. Beskriv hvorledes operationerne INSERT, EXTRACT-MAX og EXTRACT-MIN kan implementeres, så hoboperationerne har worst case tidskompleksitet $O(\log n)$ per operation. Det vil sige, at hver af operationerne skal kunne udføres i tiden $O(\log n)$

Opgave 3

Denne opgave handler om rekursion og rodfæstede binære træer. Hver knude har enten to eller ingen børn. Knuden x 's venstre barn er $left[x]$, og dens højre barn er $right[x]$. Hvis knuden x ikke har nogle børn, er $left[x]$ og $right[x]$ den specielle NIL værdi. Hvis knude x ikke har nogle børn, kaldes den et blad. Ellers kaldes den en intern knude. Såfremt rodknuden for et træ er NIL, er træet tomt. Til hver knude i træet er der knyttet en farve; knuden x har farven $farve[x]$.

Betragt følgende metode :

UDSKRIV(x)

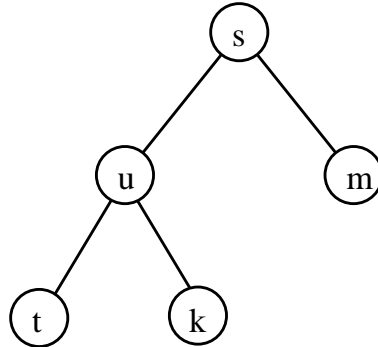
1 **if** $x \neq \text{NIL}$

2 **then** print $farve[x]$.

3 **if** $left[x] \neq \text{NIL}$

4 **then** UDSKRIV($left[x]$)

5 UDSKRIV($right[x]$)



Figur 2: Træ med knuder, der har farver angivet som bogstaver.

Lad x være rodknuden for træet i figur 2. Et kald til metoden UDSKRIV(x) vil da udskrive farvesekvensen “sutkm”.

a) Du skal ændre metoden UDSKRIV, således at det rekursive gennemløb ved kaldet til UDSKRIV(x) for rodknuden x i træet fra figur 2 udskriver farvesekvensen “smukt” i stedet.

Med den definition af binære træer, vi benytter i denne opgave, kan vi beregne antallet af knuder og antallet af blade i et træ ved hjælp af følgende to metoder:

$\text{NODES}(x)$

```
1 if  $x = \text{NIL}$ 
2   then return 0
3   else return  $1 + \text{NODES}(\text{left}[x]) + \text{NODES}(\text{right}[x])$ 
```

$\text{LEAVES}(x)$

```
1 if  $x = \text{NIL}$ 
2   then return 0
3   else if  $\text{left}[x] = \text{NIL}$ 
4     then return 1
5     else return  $\text{LEAVES}(\text{left}[x]) + \text{LEAVES}(\text{right}[x])$ 
```

b) En intern knude i et træ er en knude, som ikke er et blad. Konstruer en metode $\text{INTERN}(x)$, der givet en rodknude x for et træ returnerer antallet af interne knuder i træet. Angiv tidskompleksiteten af din løsning.

I de næste tre opgaver skal der konstrueres effektive rekursive metoder.

c) Et træ har en rød rodvej, hvis der er en vej i træet fra roden x til et blad, hvor alle knuderne på vejen fra x til bladet har farven rød. Konstruer en metode $\text{RØDRØDVEJ}(x)$, der returnerer tallet 1, hvis træet med roden x har en rød rodvej. Hvis en sådan vej ikke eksisterer, skal metoden returnere 0. Angiv tidskompleksiteten af din løsning.

d) Et træ har en rød bladvej, hvis der er en vej mellem to forskellige blade i træet, hvor alle knuderne på vejen mellem de to blade har farven rød. Konstruer en metode $\text{RØDBLADVEJ}(x)$, der returnerer tallet 1, hvis træet med roden x har en rød bladvej. Hvis en sådan vej ikke eksisterer, skal metoden returnere et tal forskelligt fra 1. Angiv tidskompleksiteten af din løsning.

e) Et træ siges at være komplet, hvis alle blade har samme dybde. Konstruer en metode $\text{KOMPLET}(x)$, som givet en rodknude x for et træ returnerer -1 , hvis træet ikke er komplet, og ellers træets højde. Angiv tidskompleksiteten af din løsning.