Eksamen i

Algoritmer og Datastrukturer

IAI 21899

Høgskolen i Østfold
Avdeling for informatikk
og automatisering

Lørdag 15. desember 2001, kl. 09.00 - 14.00
FORELØPIG LØSNINGSFORSLAG

Del 1: Algoritmeanalyse – Totalt 20%

Oppgave 1.1 - 20%

Vi antar at størrelsen på tabeller/lister er N, hvis ikke annet er angitt.

1. O(1)
2. O(N)
3. $O(N^2)$…OBS: Ved rekursjon og ”postorder” traversering: O(N)
4. O(N).
5. O(N).
6. O(logN) hvis treet er balansert, O(N) ellers.
7. O(NlogN) er beste mulige oppførsel.
8. Tilærmet O(N) så lenge tabellen er såpass mye større enn intervallet (ved bucketsort).
9. …lurespørsmål: O(NlogN)…hvis 99% hadde vært like, kunne man argumentert med at det vil bli tilnærmet O(N).
10. Tilnærmet O(1) i gjennomsnitt hvis tabellen ikke er særlig mer enn halvfull.

Slutt på del 1

Del 2: Trær I – Totalt 20%

Oppgave 2.1 - 10%

```
  void speile(BinNode rot) {
    if (rot == null) {
      return;
    }
    speile(rot.venstre);
    speile(rot.høyre);

    BinNode tmp   = rot.venstre;
    rot.venstre = rot.høyre;
    rot.høyre = tmp;
  }
```

Ombyttingen kunne også skjedd før de rekursive kallene.

Oppgave 2.2 - 10%

```
BinNode speiletKopi(BinNode rot) {
     if (rot == null) {
       return null;
     }

     return new BinNode(rot.tegn, speiletKopi(rot.høyre),
speiletKopi(rot.venstre));
  }
```

Et alternativ er å først lage en kopi av treet som det er, og deretter speile det med metoden fra forrige spørsmål.

Slutt på del 2

Del 3: Trær II – Totalt 15%

Oppgave 3.1 - 5%

Rekkefølge: A JBE HGD CFI

Traverseringen utføres ved hjelp av en kø. Start med å legge inn rota i køen. Start en løkke som går så lenge køen ikke er tom. For hver iterasjon tar man ut første element i køen (”traverserer ” det) og legger inn i køen alle barna til denne noden.

Oppgave 3.2 - 10%

Her foretar man en level order traversering og setter pekerne underveis.

Slutt på del 3

Del 4: Rekursjon – Totalt 20%

Oppgave 4.1 – 10%

For eksempel:

i	j	Første pokus kall	Rekursive pokus kall	Antall	brett[i][j]=false (viske ut)
				0	
0	0	brett[i][j] == false			
0	1	pokus(0, 1)		1	(0, 1)
			pokus(0, 2)		(0, 2)
0	2	brett[i][j] == false			
0	3	brett[i][j] == false			
1	0	pokus(1, 0)		1	(1, 0)
1	1	brett[i][j] == false			
1	2	brett[i][j] == false			
1	3	brett[i][j] == false			

Opptegning av brettet etter utvalgte kall vil også hjelpe til å klargjøre framdriften.

Oppgave 4.2 – 10%

Algoritmen visker ut alle svarte ruter. I tillegg teller den opp hvor mange ”sammenhengende” (horisontale og vertikale naboer, ikke diagonaler) svarte felter som finnes på brettet.

OBS: Returverdien til den rekursive rutinen pokus blir kun fanget opp i driverrutinen hokus…det er nettopp dette som gjør at vi kun teller samenhengende områder, og ikke hver sorte rute…

Den doble for-løkka går MxN ganger. I tillegg så besøkes/viskes alle svarte felter en gang. Til sammen gir dette O(MxN).

Slutt på del 4

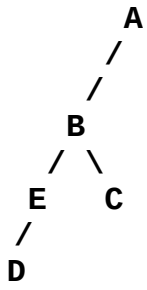
Del 5: Nettverk – Totalt 25%

Oppgave 5.1 – 5%

Trivielt.

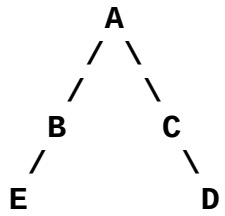
Oppgave 5.2 – 5%

Rekkefølge: A B E D C
Traverseringstre:



Oppgave 5.3 – 5%

Rekkefølge: A B C E D
Traverseringstre:



Oppgave 5.4 – 10%

Jfr. Oppgave 4! Her er det bare å generalisere brettalgoritmen til å fungere for et nettverk. Orden vil bli lineær av antall noder, med tilsvarende argumentasjon som i 4.

Slutt på del 5

Slutt på oppgavesettet