Høgskolen i Østfold

EKSAMEN

Emnekode: ITF20006 000	Emne: Algoritmer og datastrukturer
Dato: 20. mai 2011	Eksamenstid: 09:00 til 13:00
Hjelpemidler: 8 A4-sider (4 ark) med egne notater	Faglærer: Gunnar Misund

Oppgavesettet består av 4 sider inklusive denne forsiden. Kontroller at oppgaven er komplett før du begynner å besvare spørsmålene.

I implementasjonsoppgaver gir Java-kode best uttelling, men godt dokumentert pseudo-kode kan også brukes. Gjør dine egne forutsetninger hvis du føler behov for det.

Oppgavesettet består av 4 deler, med tilsammen 13 deloppgaver. Alle oppgavene skal besvares. For hver deloppgave er det angitt hvor mye oppgaven teller ved sensur. Karakteren fastsettes dog på basis av en helhetsvurdering av besvarelsen.

SKRIV TYDELIG :-)

Sensurdato: 16. juni 2011

Karakterene er tilgjengelige for studenter på studentweb senest 2 virkedager etter oppgitt sensurfrist. Følg instruksjoner gitt på: www.hiof.no/studentweb

Oppgave 1: 20%

Her følger fem Java-funksjoner. For hver av de skal du forklare hva de gjør, samt angi funksjonenes orden i O-notasjon, inkludert begrunnelse. Hver algoritme teller 4%.

```
static void func1(int[] arr, int x) {
    for (int i = arr.length-1; i >= 0; i--) {
        if (arr[i] == x) {
            System.out.println(i);
        }
    }
}

static void func2(int[] arr) {
    int s = arr.length / 2;
    for (int i = s; i < arr.length; i += 2) {
        System.out.println(arr[i]);
    }
}

static void func3(int[] arr, int i1, int i2) {
    int t = arr[i1];
    arr[i1] = arr[i2];
    arr[i2] = t;
}

static void func4(int[] arr) {
    for (int n = 0; n < arr.length - 1; n++) {
        int im = n;
        for (int m = n + 1; m < arr.length; m++) {
            if (arr[m] < arr[im]) {
                im = m;
            }
        }
        func3(arr, im, n);
    }
}
```

```
static void func5(int[] arr, int f, int t, int k) {
  if (f < t) {
      int m = f + (t - f) / 2;
      if (k < arr[m]) {
          func5(arr, f, m, k);

      } else if (k > arr[m]) {
          func5(arr, m + 1, t, k);

      } else {
          System.out.println(m);
      }
  }
}
```

Oppgave 2: 30%

Her skal du jobbe med binære søketrær (hver deloppgave teller 10%).

A) Bygg opp et binært søketre fra denne sekvensen:

`42 20 11 21 13 67 53 99 7 12 55`

Implementer funksjonen `void insert (Node n, Node root)`*, som setter inn noden* `n` *i det binære søketreet med rot i* `root`*. Se vedlegg for definisjonen av* `class Node`*.*

B) Gjør grundig rede for hvordan man sletter en node fra et binært søketre. Ta deretter utgangspunkt i treet som er bygget opp i A). Slett først noden 21, deretter 20, og til slutt 42. Illustrer hvordan du går fram.

C) Implementer metoden `void deleteSmallest (Node root)`*, som fjerner noden med minst verdi i det binære søketreet med rot i* `root`*.*

Oppgave 3: 20%

Denne oppgaven dreier seg om standard køer (FiFo: First in, First out), slike man f.eks. finner foran kassa i kantina (deloppgavene teller likt).

A) Forklar hvordan du kan bruke en enkeltlenket liste (hver node har en referanse til neste node) til å implementere en FiFo kø. Angi i pseudokode hvordan man setter inn og tar ut et element i køen.

B) *Anta at vi har en situasjon der vi vet at køen aldri vil overstige et visst antall elementer. Forklar hvordan vi kan bruke en tabell for å lage en* effektiv *implementasjon av en slik kø. Angi i pseudokode hvordan man setter inn og tar ut et element i denne køen.*

Oppgave 4: 30%

Nettverket du skal jobbe med i denne oppgaven er gitt som en liste av noder, der hver node har en liste av sine naboer, og avstanden til de:

```
A: B-8 D-12 F-42 C-14 (Node A har som nabo B, med avstand 8, etc.)
B: D-3
C: E-7
D: A-12 F-20
E: F-1
F: D-20 E-1
```

A) Tegn dette nettverket. Angi nodenes navn, retningen på kantene, samt avstandene.

B) Angi pseudokoden for en såkalt bredde-først traversering av et nettverk. Foreta en slik traversering av nettverket med start i A, og angi rekkefølgen på nodene som denne traverseringen resulterer i.

C) Finn de korteste veiene fra node A til alle de andre nodene ved hjelp av Dijkstras algoritme. Lag en oversiktlig tabell som viser hvert trinn i algoritmen. Angi til slutt de korteste veiene fra A til de andre nodene.

Vedlegg

```
public class Node {
      public int val;
      public Node left;
      public Node right;
}
```

Slutt på oppgavesettet.