Løsningsforslag til Eksamen i

DATS2300 / ITPE2300 Algoritmer og datastrukturer 28.02.2018

Oppgave 1

Metoden med navn ukjent er satt opp i vedlegget. Hva blir utskriften fra følgende kodebit? Gi en forklaring!

```
int[] a = {1,3,5,7,9};
int[] b = {2,3,5,6,8};
int[] c = new int[a.length + b.length];

int k = ukjent(a,b,c);

for (int i = 0; i < k; i++) System.out.print(c[i] + " ");
```

Vedlegg:

```
public static int ukjent(int[] a, int[] b, int[] c)
{
  int i = 0, j = 0, k = 0;
  while ( i < a.length && j < b.length)
  {
    if (a[i] < b[j]) c[k++] = a[i++];
    else if (a[i] == b[j]) { i++; j++; }
    else  c[k++] = b[j++];
  }
  while (i < a.length) c[k++] = a[i++];
  while (j < b.length) c[k++] = b[j++];
  return k;
}
```

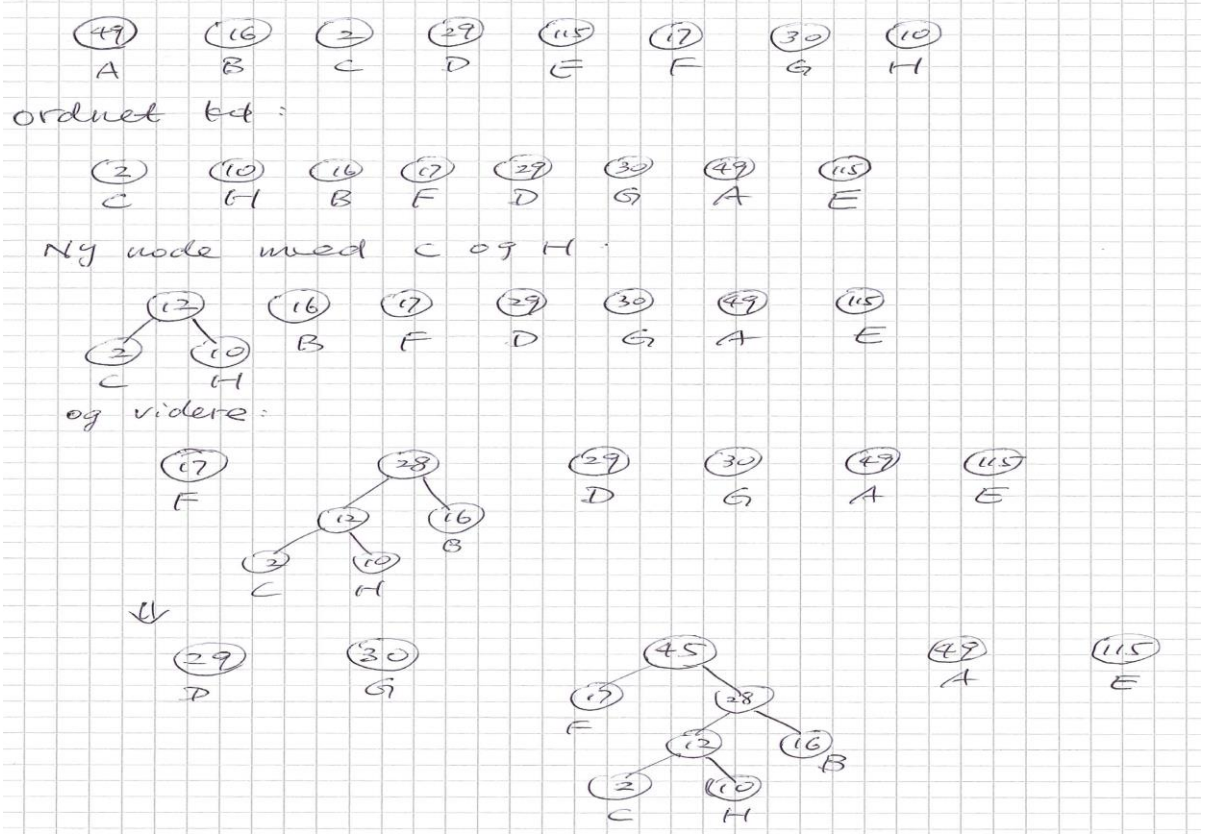
Løsning:

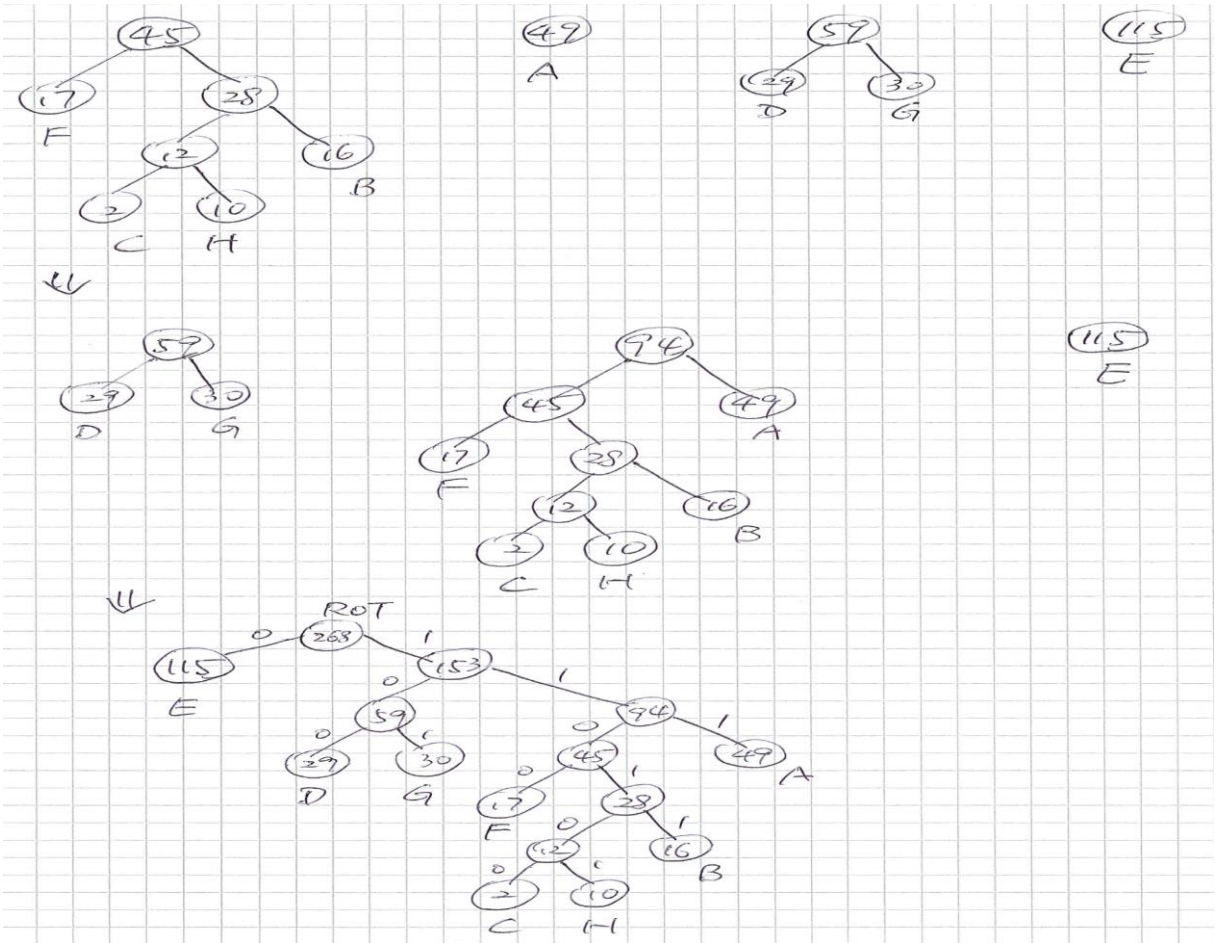
Hvis vi ser på tabellene a = {1, 3, 5, 7, 9} og b = {2, 3, 5, 6, 8} som to mengder, vil metoden *ukjent* gjøre at tabellen c vil inneholde den *eksklusive unionen* (eller det som også kalles den *symmetriske differensen*) til a og b, dvs. ($a - b$) U ($b - a$) = {1, 2, 6, 7, 8, 9}. Returverdien er antall elementer i den eksklusive unionen. Utskriften blir: 1 2 6 7 8 9.

Oppgave 2

Vi skal komprimere en sekvens med tegn ved hjelp av Huffman-teknikken. Sekvensen inneholder kun tegnene A, B, C, D, E, F, G og H med frekvenser på henholdsvis 49, 16, 2, 29, 115, 17, 30 og 10. Tegn det Huffman-treet dette gir. Sett så opp for hvert av de 8 tegnene den bitkoden som treet bestemmer. Da en liten del av sekvensen ble komprimert ved hjelp av disse bitkodene, ble resultatet: 11010110111000100110100110111111. Hvilken delsekvens var det?

Løsning:


49 A
16 B
2 C
29 D
115 E
17 F
30 G
10 H
ordnet bt:
2 C
10 H
16 B
17 F
29 D
30 G
49 A
115 E
Ny node med c og H:
12
2 C
10 H
16 B
17 F
29 D
30 G
49 A
115 E
og videre:
17 F
28
12
16 B
2 C
10 H
29 D
30 G
49 A
115 E
45
17 F
28
12
16 B
2 C
10 H


45
17 F
28
12
16 B
2 C
10 H
49 A
57
29 D
30 G
115 E
59
29 D
30 G
94
45
49 A
17 F
28
12
16 B
2 C
10 H
115 E
ROT
0
268
1
115 E
0
153
1
0
59
1
94
0
1
0
29 D
1
30 G
0
45
1
49 A
0
17 F
1
28
0
1
12
16 B
0
1
2 C
10 H

A	111
B	11011
C	110100
D	100
E	0
F	1100
G	101
H	110101

Delsekvensen er: HGFEDCBA

Oppgave 3

Lag konstruktøren public Mengde(int[] b, int n) (se vedlegget). Tabellen a skal ha lengde n og skal få som innhold de n første verdiene fra parametertabellen b. Hvis de n første verdiene i b ikke er sortert stigende eller inneholder like verdier, skal det kastes en IllegalArgumentException med en tekst. Lag så metoden toString (se vedlegget). Den skal returnere en tegnstreng som inneholder verdiene i a innrammet av hakeparenteser med komma og blank mellom hver verdi. Flg. kodebit viser hvordan dette skal virke:

```
int[] b = {1,2,5,9,11,13,0,0,0,0};

Mengde B = new Mengde(b, 6);   // de 6 første verdiene i b

System.out.println(B);         // et implisitt kall på toString
// Utskrift: [1, 2, 5, 9, 11, 13]
```

Vedlegg:

```
public class Mengde
{
  private int[] a;

  public Mengde()  // konstruktør
  {
    a = new int[0];
  }

  public Mengde(int[] b, int n)  // konstruktør
  {
    // kode mangler - skal lages
  }

  public String toString()
  {
    // kode mangler - skal lages
  }

} // class Mengde
```

Løsning:

```
public Mengde(int[] b, int n)
{
   if (n < 0 || n > b.length)
   {
      throw new IllegalArgumentException("n er utenfor tabellen!");
   }

   for (int i = 1; i < n; i++)  // usortert? duplikater?
   {
      if (b[i-1] >= b[i])
     {
       String melding = b[i-1] > b[i] ? "Usortert!" : "Duplikat!";
       throw new IllegalArgumentException(melding);
     }
   }

   // henter de n første verdiene i b
   a = Arrays.copyOf(b, n);    // en metode fra klassen Arrays
}

public String toString()
{
   return Arrays.toString(a);  // en metode fra klassen Arrays
}
```

Oppgave 4

I denne oppgaven skal du tegne binære søketrær.

Gitt tallene 16, 4, 8, 12, 7, 11, 15, 6, 7, 2, 3, 5. Legg dem inn, i den gitte rekkefølgen, i et på forhånd tomt binært søketre.

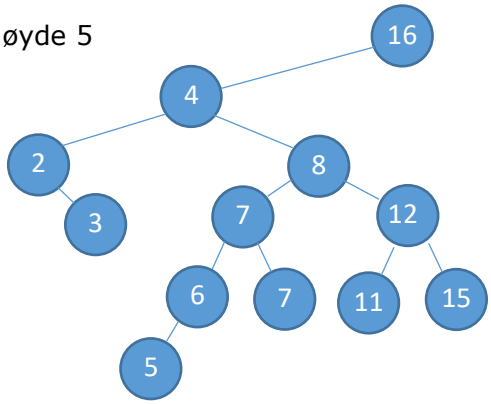
a) Tegn treet! Hvilken høyde har treet?

Du skal nå legge inn de samme tallene i et på forhånd tomt 2-3-4-tre.

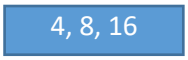
b) Tegn treet etter at du har lagt inn 16, 4, 8. Hvilken høyde har dette treet?
c) Tegn treet etter at du i tillegg har lagt inn 12, 7, 11. Hvilken høyde har dette treet?
d) Tegn det ferdige treet (etter at alle tallene er lagt inn). Hvilken høyde har dette treet?

Løsning:

a) Treet har høyde 5

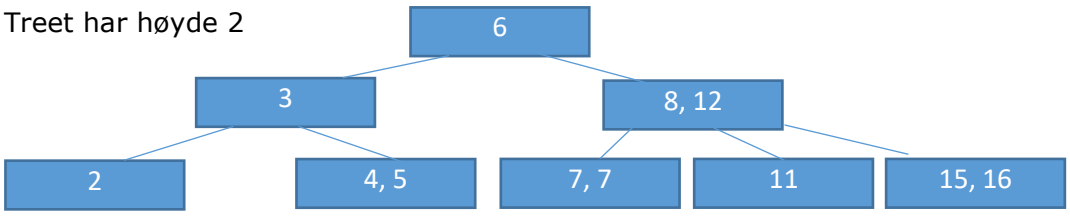


b) Treet har høyde 0

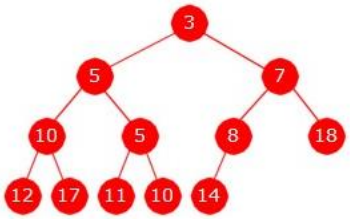


c) Treet har høyde 1

d) Treet har høyde 2



Oppgave 5



I denne oppgaven skal du jobbe med binær heap som datastruktur. Figuren i vedlagt bilde viser en binær heap (også kalt maksimumstre).

a) Forklar hva som er spesielt med en binær minimumsheap: hvilke egenskaper må til for at det er en binær minimumsheap?

b) Hva kan en binær minimumsheap brukes til?

c) Gitt heap datastrukturen vedlagt bilde, legg til verdiene 6, 10, og 12. Tegn heapen for hvert tall du legger til.

d) Forklar hvordan en tabell («array») kan brukes for å lagre en minimumsheap i minnet.

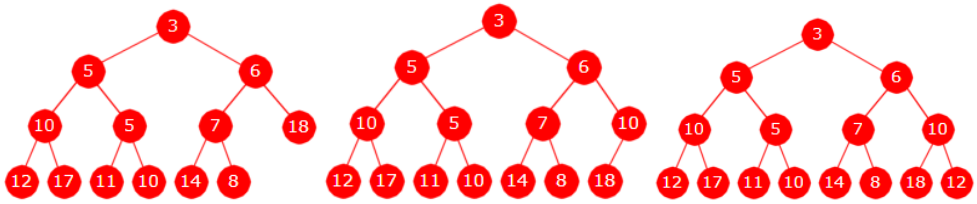
e) Tegn tabellen som tilsvarer heap datastrukturen i figuren.

Løsning

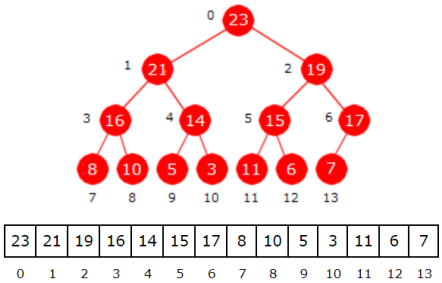
a) En binær minimumsheap (eng: a binary min-heap) er et binært og komplett minimumstre. Et binærtre kalles et minimumstre (eng: a min tree) hvis verdien i hver indre node er mindre enn eller lik verdiene i nodens barn.

b) Blant annet til sortering, prioritetskø

c)



d) Man nummererer hver node. Siden treet er komplett vil alle unntatt siste element i arrayet være benyttet. Eksempel:



e) [3, 5, 7, 10, 5, 8, 18, 12, 17, 11, 10, 14]

Oppgave 6

Klassen LenketHashTabell bruker «lukket adressering med separat lenking». Den inneholder en tabell med nodereferanser der alle i utgangspunktet er null. Et objekt legges inn på objektets tabellindeks (objektets hashverdi modulo tabellengden). Dvs. en node (med objektet) legges først i den (eventuelt tomme) lenkede nodelisten som hører til tabellindeksen.

En samling navn (tegnstrenger) skal legges inn. Det er en jobb å regne ut hashverdier for hånd. Dette er derfor allerede gjort for noen lengder/dimensjoner:

navn	Espen	Bo	Ali	Petter	Karl	Siri	Muhammad	Mari	August	Åse
hashverdi	80088	2357	65964	89125	69562	257197	7934	23763	65085	1983

Setningen LenketHashTabell<String> hash = new LenketHashTabell<>(n); oppretter en instans av klassen der den interne tabellen får dimensjon (lengde) lik n. Legg inn én og én verdi i den gitte rekkefølgen (dvs. Espen, Bo, Ali, osv). En node skal ha både verdi og hashverdi, men på en tegning holder det med verdi.

For enkelthets skyld velger vi å lage en hashtabell av lengde 10. Utfør følgende oppgaver med den lenkede hashtabellen:

a) Regn ut indeksen for hver hashverdi i hashtabellen.

b) Lag en tegning av datastrukturen når de fem første navnene er lagt inn

c) Lag så en tegning som viser når alle navnene er lagt inn. Du trenger ikke å ta hensyn til tetthet (load factor)

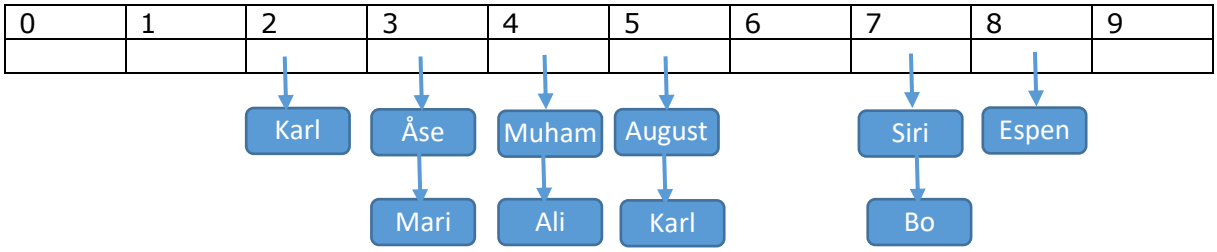
d) Hva betyr begrepet tetthet (load factor) i denne sammenheng, og hva må du gjøre om du skal ta hensyn til dette?

e) Den lenkede hashtabellen vi har brukt har lengde 10: Er dette en god lengde? Hvorfor / hvorfor ikke?

Løsning

a) Espen=>8, Bo=>7, Ali=>4, Petter=>5, Karl =>2, Siri => 7, Muhammad =>4, Mari =>3, August =>5, Åse => 3

b/c)



d) Tetthet er antall elementer delt på dimensjonen til tabellen. Når tettheten er stor er det stor sjanse for kollisjoner og flere elementer ligger på samme indeks. Ved mange kollisjoner vil søking bestå av lineært søk i lenket liste istedenfor et oppslag i tabell.

e) 10 er vanligvis ikke en god lengde: den er liten i forhold til antall elementer (høy load factor). Hvis man også bruker et (passe stort) primtall som dimensjon kan man minske sannsynligheten for kollisjoner.