Algoritmer og datastrukturer
Løsningsforslag

Eksamen – 2. desember 2009

Oppgave 1A

Hvis vi ser på tabellene *a* = {1,2,3,4,5} og *b* = {3,4,5,6,7} 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}. Returverdien er antall elementer i den eksklusive unionen. Utskriften blir: 1 2 6 7

Oppgave 1B

Først legges *A*, *B* og *C* i køen. Så tas den første (dvs. *A*) ut og legges inn bakerst. Dermed inneholder køen *B C A*. Dette gjentas to ganger. Dvs. køen vil deretter inneholde *C A B* og til slutt *A B C*. Dermed blir utskriften: *A B C*.

Oppgave 1C

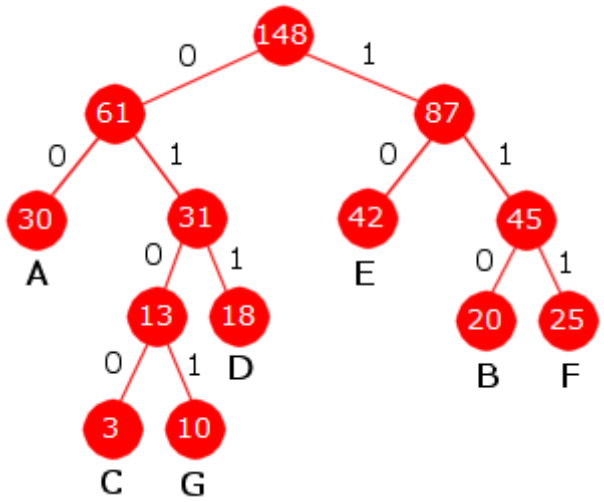
Verdiene fra de to køene sammenlignes parvis, tas ut og legges inn (bakerst). Hvis det kommer et par som er ulike, stopper sammenligningene og det som eventuelt er igjen i køene, tas ut og legges inn. Dermed vil køene være som de var.

```
public static <T> boolean erLike(Kø<T> A, Kø<T> B)
{
  // De må ha samme antall for å kunne være like
  if (A.antall() != B.antall()) return false;

  boolean erLike = true;
  int n = A.antall(), i = 0;

  for (; i < n && erLike; i++)
  {
    erLike = A.kikk().equals(B.kikk());
    A.leggInn(A.taUt());  // tar ut og legger inn
    B.leggInn(B.taUt());  // tar ut og legger inn
  }

  for (; i < n; i++)      // tar med resten av køene
  {
    A.leggInn(A.taUt());  // tar ut og legger inn
    B.leggInn(B.taUt());  // tar ut og legger inn
  }

  return erLike;
}
```

Oppgave 2A

```
public static String toString(int[] a)
{
  if (a == null) return null;

  if (a.length == 0) return "[]";

  StringBuilder s = new StringBuilder();

  s.append('[');
  s.append(a[0]);

  for (int i = 1; i < a.length; i++)
  {
    s.append(',');
    s.append(' ');
    s.append(a[i]);
  }

  s.append(']');

  return s.toString();
}
```

Oppgave 2B



Huffmantreet for A, B, C, D, E, F og G

A = 00
B = 110
C = 0100
D = 011
E = 10
F = 111
G = 0101

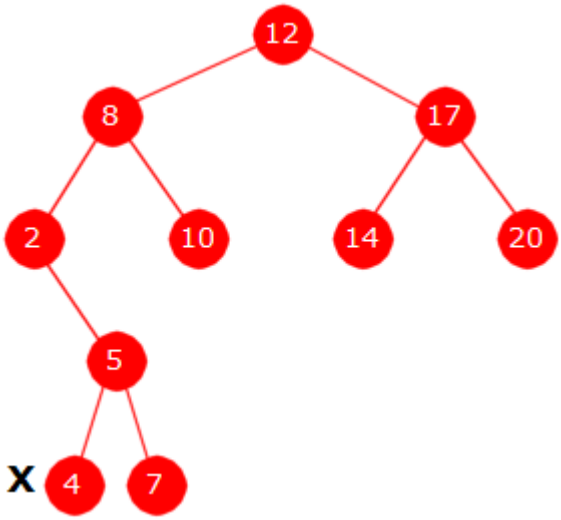
Delsekvensen var ABCDEFG

Oppgave 2C

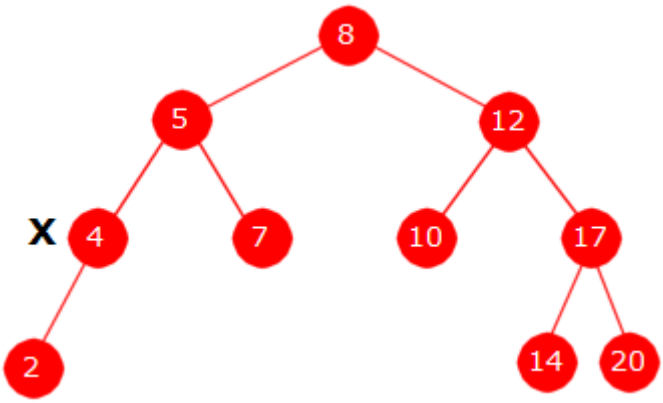
```
private static Node byggHuffmanTre(String[] koder)
{
  Node rot = new Node();

  for (int i = 0; i < koder.length; i++)
  {
    String s = koder[i];
    if (s != null)  // tegnet med ascii-verdi lik i er med
    {
```

```
        Node p = rot;
        for (int j = 0; j < s.length(); j++)
        {
          if (s.charAt(j) == '0')
          {
            if (p.venstre == null) p.venstre = new Node();
            p = p.venstre;
          }
          else
          {
            if (p.høyre == null) p.høyre = new Node();
            p = p.høyre;
          }
        }
        p.tegn = (char)i;  // legger tegnet i bladnoden
      }
    }
    return rot;
  }
```

Oppgave 3A



Tallene 12, 8, 10, 17, 14, 2, 5, 20, 4, 7

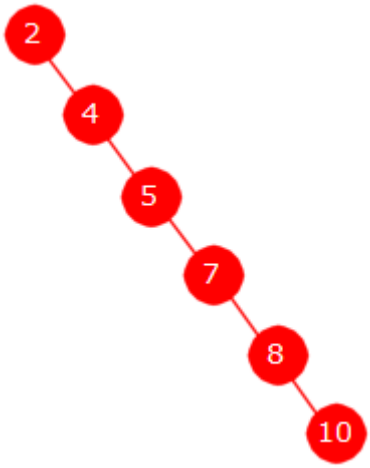


Tallene 8, 12, 10, 17, 5, 7, 4, 20, 2, 14

Oppgave 3B

```
public T nestMinst()
{
  if (antall < 2) throw new
    NoSuchElementException("Treet har færre enn to noder!");

  Node<T> p = rot, f = null;

  while (p.venstre != null)
  {
    f = p;  // f er forelder til p
    p = p.venstre;
  }

  if (p.høyre == null) return f.verdi;

  p = p.høyre;

  while (p.venstre != null)
  {
    p = p.venstre;
  }
  return p.verdi;
}
```

Oppgave 3C

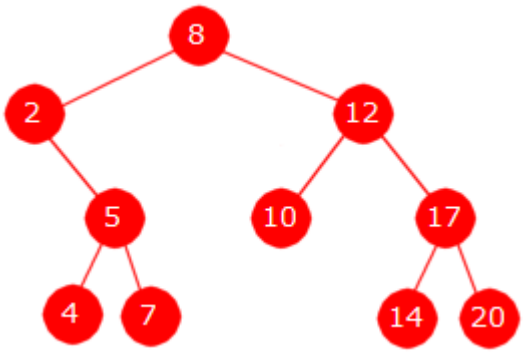



```
public boolean erFullstendigHøyreskjevt()
{
  Node<T> p = rot;

  while (p != null)
  {
    if (p.venstre != null) return false;
    p = p.høyre;
  }

  return true;
}
```

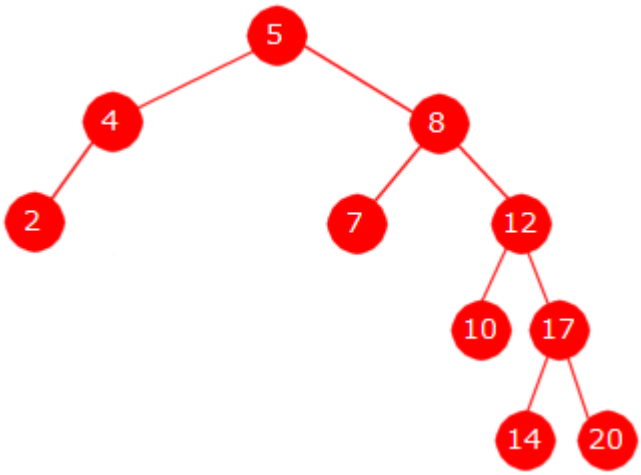
Tallene 2, 4, 5, 7, 8, 10

Oppgave 3D

```
private static <T> Node<T> hRotasjon(Node<T> p)
{
  Node<T> q = p.venstre;

  p.venstre = q.høyre;
  q.høyre = p;

  return q;
}
```



En høyrerotasjon på roten i det første treet



En høyrerotasjon på roten i det andre treet

Første versjon av metoden gjørFullstendigHøyreskjevt:

Det enkleste er å først gjøre en serie rotasjoner på rotnoden inntil den ikke lenger har venstre barn. Deretter fortsetter vi på nodene nedover mot høyre fra rotnoden:

```
public void gjørFullstendigHøyreskjevt()
{
  if (tom()) return;

  while (rot.venstre != null) rot = hRotasjon(rot);

  Node<T> p = rot;

  while (p.høyre != null)
  {
    while (p.høyre.venstre != null) p.høyre = hRotasjon(p.høyre);
    p = p.høyre;
  }
```

Andre versjon av metoden gjørFullstendigHøyreskjevt:

Dette kan gjøres på mange måter. Hvis treet traverseres i inorden, kan pekerne endres under traverseringen. Alternativt kan nodene legges fortløpende f.eks. over en kø. Deretter lages det et fullstendig høyreskjevt tre ved å ta ut én og én node fra køen og for hver node gjøre noden til høyre barn til forrige node:

```java
private static <T> void inorden(Node<T> p, Kø<Node<T>> kø)
{
  if (p.venstre != null) inorden(p.venstre, kø);
  kø.leggInn(p);
  if (p.høyre != null) inorden(p.høyre, kø);
}

public void gjørFullstendigHøyreskjevt2()
{
  if (tom()) return;

  Kø<Node<T>> kø = new TabellKø<Node<T>>();
  inorden(rot,kø);

  rot = kø.taUt();     // den første blir ny rot
  rot.venstre = null;

  Node<T> p = rot;
  while (!kø.tom())
  {
    p.høyre = kø.taUt();
    p = p.høyre;
    p.venstre = null;
  }
}
```