OBLIGATORISK OPPGAVE 3 - INF110, høsten 2003

1 Tekstfilkomprimering

I denne oppgaven, som er en individuell oppgave i forhold til i fjor, skal du lage to programmer KompTekst.java og DeKompTekst.java som hhv. 'pakker ned' og 'pakker ut' en vilkårlig tekstfil (innfila). Vitsen med denne nedpakkingen på an annen fil (pakkefila), er at denne pakkefila er (langt) mindre enn den originale fila, men inneholder nok informasjon til å gjenskape denne (noe ala WinZip). Vi skal, når vi pakker ned og opp en tekstfil på et slikt komprimert filformat, tillate oss å miste formateringsinformasjon som linjeskift, men alle ordene må kunne gjenskapes i riktig rekkefølge (av 'DeKompTekst.java') og med enkelt mellomrom mellom ordene på den gjenskapte fila. For å løse denne oppgaven skal du bruke noen av de teknikkene du har lært om binære søketrær.

Hensikten med denne obligatoriske oppgaven er at du skal trene du i å programmere disse trærne selv. Vi skal derfor kreve at du bruker disse mekanismene og bare ellers nytter de to subklasser InnB og UtB av hhv klassene Inn og Ut i inf101-pakken. Vi skal også kreve at du bruker en bestemt node 'class TFNode' i trærne (slik at kodingen ikke blir ren avskrift av forelesningsnotatene). Løsninger med f.eks bruk av de ferdige klassene i Java-biblioteket for trær, vil ikke bli godkjent.

```
class TFNode
{   static int løpendeKode = 0,    // used during coding of the words
               antall = 0;         // number of different words on file
    String ord;                    // the word in this node
    int freq = 1;                  // number of 'ord' on infile
    int koding;                    // coding of 'ord' on packed file
    TFNode vsubT, hsubT,           // Text tree pointers
           vsubF, hsubF;           // Frequency tree pointers

    TFNode (String ord)
    {   <... skrives av deg ...>
    }

    /** Insert 'nyOrd' in text-tree **/
    void insertT (String nyOrd)
    {   <... skrives av deg ...>
    }

    /** Insert node 't' in frequency-tree **/
    void insertF (TFNode t)
    {   <... skrives av deg ...>
    }

    <... + andre nyttige metoder som skrives av deg ...>

} // end class TFNode
```

Programmet KompTekst.java består av flere faser, og leser hele innfila i to ganger:

- Innlesning av innfila ord for ord og oppbygging av et søketre (Tekst-treet) over alle de ulike ordene og hvor mange ganger hvert ord forekommer (frekvensen) på innfila.

- Innsetting av de samme nodene også i et binært søketre som er sortert på frekvensene (Frekvens-treet).

- Tildeling av koder til de ulike ordene slik at det ordet med høyest frekvens får kode '0', nest høyeste frekvens får kode '1', ... osv.

- Utskrift av første del av pakke-fila: Ført to heltall: Antall ulike ord og totalt antall ord på innfila. Deretter kommer alle ordene i rekkefølge sortert på frekvensen slik at det ordet som har kode '0' kommer først, deretter ord med kode '1',... osv.

- Innfila leses nå en gang til (lukk den og åpn den igjen), og for hvert ord på innfila, skrives koden til dette ordet (med metoden 'utKode' i klassen ' UtB') ut på pakke-fila. Dette blir da del 2 av pakke-fila.

Programmet DeKompTekst.java blir nå enkelt å lage. Det leser pakke-fila en gang, bygger først opp en enkel datastruktur for kodingen av ordene ut fra del 1 av pakkefila. Så leser den selve kodingen av innfila (med innKode i klassen 'InnB') på del 2 og skriver ut en gjenskapt innfil (minus evt. linjeskift). Legg selv inn linjeskift så ingen linje blir lenger enn si 70 tegn.

Programmene skal startes med følgende parametere:

```
>java TekstKomp 'innfil' og >java TekstDeKomp 'pakkefil'
```

Du skal, etter at du har først har testet programmet ut på noen små tekstfiler du selv har laget, legge ved første side av utskrift av pakke-fila og første side av den utpakkede fila du får når du kjører programsystemet ditt på 'darwin1.txt' (som er del 1 av Charles Darwin's bok "The Descent of Man"). Fila 'darwin1.txt' finner du på samme område som programeksemplene på hjemmesida.

Hvis du tar utgangspunkt i søketre-eksemplet fra forelesningen (Ibsen), må du fjerne de metodene fra det eksempelet som du ikke trenger her og tilpasse de resterende metodene til kravene ovenfor om at nodene skal kunne være med i to trær samtidig, for å få godkjent besvarelsen.

Hint:

- Metodene innKode og utKode i klassene InnB og UtB brukes for å skrive koden for et ord (en byte for de mest brukte, to byte for de andre) ut på pakkefila. Grunnen til at du får disse, og ikke må skrive dem selv, er at visse tegn ikke aksepteres på en fil (kontrolltegnene), Man får simpelt hen ikke lest tilbake det man trodde man skrev ut. Hvis du vil forbedre disse metodene og klassene, så gjerne det - men du er advart. Du finner klassene på fila: InnBUtB.java som er sammen med programeksemplene på hjemmesida.

- Husk å gi pakke-fila et nytt tillegg til filnavnet.

2 Frivillige utvidelser

Det er **IKKE** noe krav, men hvis du synes oppgaven er for lett eller for kort, kan den forbedres med bl.a.:

- Forsøk å få med noe av formatteringen på innfilen over i pakkefila og den utpakkede fila (**middels vanskelig**).

- Metodene innKode og utKode i klassene InnB og UtB kan bare kode inntil 21000 ulike verdier i 1 eller 2 byte. Utvid kodingen til opptil 3 byte koding (**lett**).

- Sammenlign ytelsen mellom denne og 'WinZip' (**meget lett**).

- Ordlisten over de ulike ordene på pakke-fila tar stor plass. Prøv å bruke Huffmankoding (læreboka s. 357-362) for å lage den delen kortere. Husk at du fortsatt må bruke 'utKode()' og 'innKode()' - eller noe lignende for å lagre kompakt på pakke-fila. Det vil da være OK å lagre 14 bit i to byter med disse metodene. Du gjentar da på bokstavnivå for denne delen (og med koding i bit) omlag det samme som oppgaven gjør på ordnivå (med koding i byte) for resten av filen (**vanskelig**).