UNIVERSITETET I OSLO

Det matematisk-naturvitenskapelige fakultet

Eksamen i :	INF 110 — Algoritmer og datastrukturer
Eksamensdag :	Lørdag 8. desember 2001
Tid for eksamen :	09.00 - 15.00
Oppgavesettet er på :	5 sider inkludert vedlegget
Vedlegg :	Koden til permutasjon fra forelesningene
Tillatte hjelpemidler :	Alle trykte og skrevne

Kontroller at oppgavesettet er komplett før du begynner å besvare det

Oppgave 1 (20 %)

Et rørleggerfirma har et langt rør med lengde L som det selger i biter. Bestillingene på rørbiter (egentlig lengden på bestilte rørbiter) ligger lagret i en array a[n] (der altså n er antall bestillinger). Vi antar at summen av bestillingene er større enn L, og oppgaven er å minimalisere avkappet, det vil si lengden av den delen av røret som ikke blir solgt.

I programmeringen skal du gå ut fra at både L og a[ ] er heltall.

Oppgave 1 A

Bruk en enklest mulig (orden O(n)) grådig algoritme til å skrive en metode til å foreslå hvordan røret bør kappes opp. Skriv ut hvilke lengder fra a[ ] du kapper røret opp i og lengden av avkappet. Begrunn hvorfor algoritmen din er O(n).

Oppgave 1 B

Bruk permutasjoner og skriv en metode som finner en optimal løsning (permutasjonsprosedyrene fra forelesningene er vedlagt).
Skriv ut som i 1A.

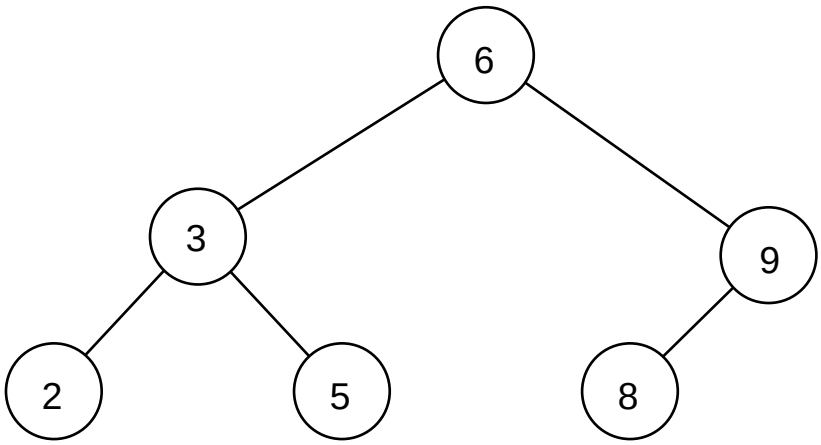
Oppgave 1 C

Forklar hvordan du, etter å ha sortert a[ ], kan finne minste antall og største antall kapp du kan lage. Bruk dette til å foreslå forbedringer i metoden du skrev i 1 B. Her skal du ikke programmere, men bare beskrive med ord hva du ville ha gjort.

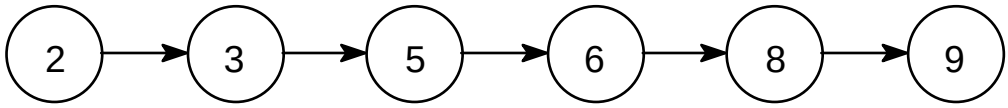
Oppgave 2 (30 %)

Vi skal i denne oppgaven se på hvordan vi kan omforme binære søketrær til sorterte lister.

Her er et eksempel på hvordan dette søketreet:



kan omformes til denne sorterte enveislisten:



Nodene i treet er objekter av følgende klasse:

```
class Node {
    int verdi;      // sorteringskriterium
    Node vsub;      // (peker til) venstre subtre
    Node hsub;      // (peker til) høyre subtre
    - - - - - - - - // eventuelle metoder
}
```

I denne oppgaven har vi et program hvor vi har deklarert en **`Node rota`**; som peker på rot-noden i det binære søketreet.

Oppgave 2 A

Skriv en metode **`void lagEnveisListe(Node hode)`** i **`class Node`** som lager en sortert enveis liste av binærtreet med **`vsub`** i nodene som nestepeker i lista.
Programbiten som kaller metoden er:

```
Node liste,
     hode = new Node();
rota.lagEnveisListe(hode);
liste = hode.vsub;
```

Etter kallet skal **`vsub`** i noden som bringes med som parameter, peke på første node i lista, og **`hsub`** på siste node i lista. I den siste noden i lista skal **`vsub`** peke på **`null`**.

Oppgave 2 B

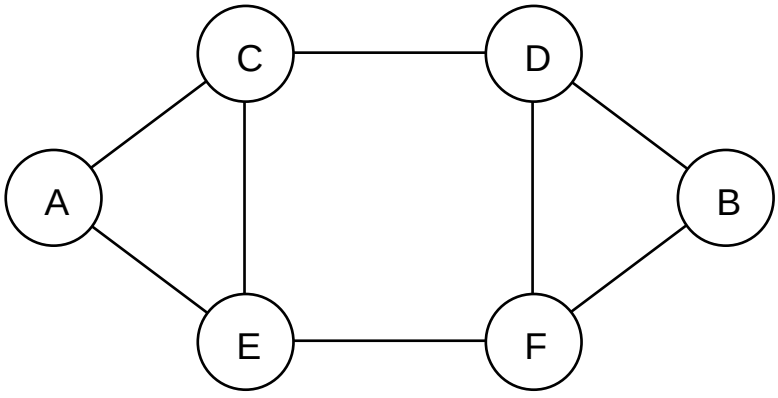
Skriv en metode **`void lagToveisListe(Node hode)`** i **`class Node`** som lager en sortert toveis liste av binærtreet med **`vsub`** som nestepeker og **`hsub`** som forrigepeker i lista.

Metoden i 2B kalles med helt tilsvarende kode som i 2A. Nå skal **`hsub`** i den første noden i lista og **`vsub`** i siste node begge peke på **`null`**.

Oppgave 2 C

Forklar hvorfor det ikke er lurt å prøve det omvendte: å lage et binært søketre av en sortert liste.

Oppgave 3 (50 %)



I denne oppgaven skal vi se hvordan vi kan finne flest mulig ulike (disjunkte) stier mellom to noder i en graf. Vi ser av grafen nedenfor at det opplagt er maksimalt to disjunkte stier mellom A og B: A-E-F-B og A-C-D-B. Hvis vi derimot først velger f.eks. stien A-C-E-F-B, greier vi ikke å trekke flere stier mellom A og B som er disjunkt med den første.

Mer presist, la G = (V,E) være en urettet graf, der V er mengden av noder, E er mengden av kanter i G, og anta at det mellom to noder går høyst en kant, og at ingen kant går fra en node til seg selv.

La A og B være to forskjellige noder i V. Nedenfor skal du også anta at det ikke går noen kant mellom A og B (dvs at $(A,B) \notin E$).

En sti fra A til B er en mengde av noder $\{V_0, V_1, \ldots, V_k\}$ slik at $V_0 = A$, $V_k = B$, og $(V_{i-1},V_i) \in E$ for $1 \leq i \leq k$. To stier fra A til B kalles disjunkte hvis A og B er de eneste nodene som er med i begge stiene.

Nodene i programdelene du skal skrive, er representert med objekter av klassen Gnode:

```
class GNode {
   String navn;
   GNode [ ] kanter;
   boolean medISti;
}
```

Når programdelene du skal skrive starter, er grafen initiert, og **GNode [ ] kanter** inneholder pekere til de andre nodene som det er kanter til fra denne noden i G. En kant i G er følgelig representert med en peker i begge nodene kanten går mellom.

Oppgave 3 A

Hvis vi bruker en enkel (endimensjonal) array av nodepekere til å holde oversikt over de disjunkte stiene vi har funnet fra A til B (og vi registrerer dem slik det er gjort i første avsnitt i oppgaven), hvor lang er da denne arrayen maksimalt? Begrunn svaret.

Oppgave 3 B (35 %)

Skriv en metode: **void maxAntallStier(GNode a, GNode b)** og andre metoder og data du evt. trenger til å finne det maksimalt mulige antall disjunkte stier fra A til B i G. Vi antar at algoritmen du bruker gjør *rekursiv, dybde-først søk* – men andre riktige løsninger vil selvagt også bli akseptert.
(*Hint: Hva gjør du når du finner B ?*)
Du skal også lage kode som tar vare på de ulike stiene du finner. Til sist skal du skrive ut stiene i (en av de) løsningen(e) med maksimalt antall stier som du har funnet. Bruk samme format på utskriften som i innledningen og skriv ut Stringen **navn** for hver node i hver sti, samt max antall stier funnet.

Oppgave 3 C

Forklar hvorfor vi ikke kan bruke en algoritme som finner korteste vei fra A til B som subalgoritme for å løse problemet i oppgave 3B.
(*Hint: Legg inn nye noder i eksempelgrafen ovenfor og lag et moteksempel*)

Vedlegg – permutasjon:

```
void bytt (int [] a, int k, int m)
  { int temp = a[k];
     a[k] = a[m];
     a[m] = temp;
  }
void roterVenstre(int [] a, int i)
   { int x,k;
     x = a[i];
     for (k= i+1; k < n; k++)
       a[k-1] = a[k];
     a[n-1] = x;
   }

void permuter (int [] a, int i)
   { // finn neste permutasjon og kall "brukPerm(a)
     // N.B. Permutasjonene startes ved kallet:  permuter(a,0);

    if ( i == n-1) brukPerm(a) ;
    else {
        permuter(a,i+1);
        for (int t = i+1 ; t < n; t++)
        {  bytt (a,i,t);
           permuter(a,i+1);
        }
        roterVenstre(a,i);
     }
   }
```

Slutt på oppgavesettet, Lykke til !

Arne Maus og Ragnar Normann