# UNIVERSITETET I OSLO

## Det matematisk-naturvitenskapelige fakultet

| | |
|---|---|
| Eksamen i | IN 110 — Algoritmer og datastrukturer |
| Eksamensdag: | 15. mai 1995 |
| Tid for eksamen: | 9.00–15.00 |
| Oppgavesettet er p 11 sider. | |
| Vedlegg: | Ingen |
| Tillatte hjelpemidler: | Alle trykte og skrevne |

Kontroller at oppgavesettet er komplett fr
du begynner  besvare sprsmlene.

## Merk:

Oppgavesettet består av tre deler som kan løses uavhengig av hverandre. Oppgavene innen hver del er tenkt løst i den rekkefølge de står, men dette er selvsagt intet krav til din besvarelse. Prosenten angitt påhver del antyder hvor mye vekt det vil bli lagt pådenne delen under sensureringen.

Programmer skal skrives i Simula. Du behøver ikke gi noen fullstendig dokumentasjon av programmene, men du skal skrive noen fålinjer som gir leseren nøklen til forståelse av programmet. Du kan anta at leseren kjenner problemstillingen i oppgaven meget godt.

Når det i teksten under refereres til "læreboka" menes boka "Data Structures and Algorithm Analyses" av Mark Allen Weiss.

Alle steder der det er spørsmål etter et program (eller en programbit) skal du skrive dette helt ut, og *ikke* bare henvise til liknende programmer f.eks. i læreboka.

*(Fortsettes p side 2.)*

*Les oppgavene nøye, og lykke til!* Almira Karabeg og Stein Krogdahl

*(Fortsettes p side 3.)*

# Del 1 (35%)

I denne delen skal vi se påfire uavhengige oppgaver.

### Oppgave 1-a

Vi skal se påutvidbar hashing (extendible hashing). Vi antar at det er plass til maksimalt fire nøkler (med tilhørende data) i hver blokk (altså: m=4), og at nøklene er bitsekvenser med 8 bit. Pået gitt tidspunkt er hash-systemet fylt opp som følger:

Anta så at man setter inn følgende nøkler i hash-systemet:

00101111 og deretter 10111110

Tegn to figurer: Én som viser systemets tilstand etter at første verdi er satt inn og én som viser systemets tilstand når begge verdier er satt inn.

### Oppgave 1-b

Vi har definert et rotrettet tre, der nodene i treet er identifisert med tallene fra 1 til N, og der roten har nummer R. Treet er representert ved en 'integer array frd(1:N)' der 'frd(k)' angir foreldrenoden til noden k. For roten antar vi: 'frd(R)=0'.

For en gitt node X vil denne representasjonen gjøre det lett åfølge veien gjennom treet fra noden X til roten R. Vi får imidlertid ofte bruk for ågåden motsatte vei, altsåat vi for en gitt node X ønsker åskrive ut nodene langs veien fra R til X, i den rekkefølge. For lett åkunne gjøre dette vil vi bruke informasjonen i frd-arrayen til åsette opp en todimensjonal tabell 'integer array retning(1:N, 1:N)' slik at vi kan fåskrevet ut nodene langs veien fra R til X ved følgende algoritme:

*(Fortsettes p side 4.)*

```
integer node;
...
node:= R; outint(node, 10);
while node <> X do begin  node:= retning(node,X);  outint(node, 10) end;
```

Du skal skrive en prosedyre som ut fra informasjonen i arrayen 'frd' fyller arrayen 'retning' slik at algoritmen over virker. Prosedyren skal ha følgende form:

```
procedure fyllretning(frd, retning);
   integer array frd;     ! Dimensjonert (1:N). Er fylt nr prosedyren kalles ;
   integer array retning; ! Dimensjonert (1:N,1:N). Skal fylles av prosedyren ;
begin ... end;
```

### Oppgave 1-c

Om vi lagrer et antall elementer ved hjelp av hashing og har en god hashfunksjon og et fornuftig forhold mellom antall elementer og lengden av hash-tabellen såkan det være rimelig åsi at aksesstiden til et vilkårlig element er $O(1)$. Anta nåat vi har et hash-system med separat kjeding og med en gitt (konstant) lengde påhash-tabellen. Antall elementer $n$ viser seg imidlertid åbli mange ganger større enn lengden av hash-tabellen. Hvordan er det da rimelig åangi aksesstiden i $O$-notasjon.

### Oppgave 1-d

Ole skal skrive en spesial-prosedyre for åsortere fem tall. Prosedyren skal bygge påat den sammenlikner to og to tall og gjør omordninger ut fra dette. Han påstår han skal kunne skrive prosedyren slik at det aldri vil bli utført mer enn syv sammenlikninger før sorteringen er ferdig. Er det håp om at han kan klare dette, eller tror du han tar feil? Forklar kort.

## Del 2 (40%)

Vi skal se påbinære trær, og vi er bare interessert i trærnes form, ikke identiteten eller innholdet av den enkelte node. En måte åangi formen til et vilkårlig binærtre er åførst tenke seg at alle de tomme subtrærne i treet erstattes av

*(Fortsettes p side 5.)*

en spesiell node-type (som vi markerer med en T i figuren under). Deretter skriver vi ut dette utvidede treet i postfiks form, og vi angir her alle de opprinnelige nodene med samme symbol, nemlig N, og alle de tomme trærne med T. Som et eksempel kan vi se pået tre med 4 noder, det tilsvarende utvidede treet, og den resulterende postfiks form:

T T T N N T T N N

Dette skal vi kalle åangi treet (det med 4 noder!) på **utvidet postfiks form**. Angående slik utvidet postfiks form gjelder følgende (som du ikke behøver å begrunne): *En sekvens av T-er og N-er vil være utvidet postfiks form av et binærtre med n noder hvis og bare hvis sekvensen har n N-er og $n + 1$ T-er, samt at man påhvert punkt ved lesing fra venstre mot høyre alltid har lest flere T-er enn N-er.*

### Oppgave 2-a

Vi skal skrive en prosedyre som ut fra et tre gitt påutvidet postfiks form skal generere det tilsvarende treet. Nodene i treet som genereres skal være av følgende klasse:

```
class node;
begin
    ref(node) vsub, hsub;
    ... ! eventuelle data som ikke interesserer oss ;
end;
```

Prosedyren skal ha følgende form, og skal levere roten av det ferdige treet som resultat:

```
ref(node) procedure gentre(n,S);
   integer n; ! Antall (ekte) noder i treet ;
   character array S; ! En character array dimensjonert (1:2*n+1), som
                        inneholder T-er og N-er i en sekvens som angir
```

*(Fortsettes p side 6.)*

```
                        et binrtre p  utvidet postfiks form ;
begin
   ...
end;
```

Merk altsåat det bare skal genereres n noder, og at de tomme trærne bare skal representeres ved **none**-pekere. Prosedyren skal ogsåfungere om det angitte treet er tomt.

Hint: Bruk en stakk av (sub)trær.

### Oppgave 2-b

Vi skal nåse pådet ågenerere alle mulige binærtrær med forskjellig form, men med et gitt antall noder $n$. For $n = 4$ får vi f.eks. 14 forskjellige trær, nemlig de følgende 7, samt de 7 speilvendte:

Vi skal gjøre denne genereringen ved ågenerere alle mulige sekvenser av T-er og N-er som er utvidet postfiks form av binærtrær med $n$ noder (se karakteriseringen av slike sekvenser over).

Du skal programmere dette som en prosedyre 'lagalle' som skal ha følgende form:

```
procedure lagalle(n); integer n;
begin
   character array S(1:2*n+1);
   ..... ! Andre deklarasjoner;

   procedure lagrek(k); integer k; ! Denne skal gjre den rekursive ;
   begin                           ! genereringen av sekvensene     ;
      ...
   end;

   ...
end;
```

*(Fortsettes p side 7.)*

Sekvensene kan passelig genereres i S ved at vi i utgangspunktet genererer alle sekvenser av N-er og T-er, og sålegger inn avskjæring slik at vi bare får de sekvenser som representerer binære trær i utvidet postfiks form. Gjør avskjæringen slik at genereringen blir såeffektiv som mulig. For hver ferdig sekvens skal man kalle prosedyren:

procedure bruksekv(n,S); ...;

Her er n og S spesifisert som i prosedyren 'gentre' beskrevet tidligere.

### Oppgave 2-c

I et tre har hver node en dybde (og vi regner at roten har dybde null). Om 't' er et tre lar vi 'ant(t)' og 'sumd(t)' bety h.h.v. antall noder og summen av dybden over alle nodene i treet. For et ikketomt binærtre 't' der roten har de to subtrærne 't1' og 't2', gjelder:

$$sumd(t) = sumd(t1) + ant(t1) + sumd(t2) + ant(t2)$$

Gi en kort begrunnelse for at dette er riktig. Hva må$sumd(t)$ være for et tomt tre $t$?

*(Fortsettes p side 8.)*

### Oppgave 2-d

Vi er nåinteressert den gjennomsnittlige dybden tatt over alle nodene i et gitt binærtre. Treet er gitt påutvidet postfiks form. Du skal skrive en prosedyre:

real procedure snittdybde(n,S); ...;

som beregner dette gjennomsnittet *uten ågenerere selve treet*. Her er n og S spesifisert som i prosedyren 'gentre' beskrevet tidligere. Du kan anta at $n$ er minst 1.

### Oppgave 2-e. Kan puffes (Dårligste karakter er 4.0)

Vi kunne nålett skrive en 'real procedure totalsnitt(n)' som går gjennom alle trær med $n$ noder og for hver av disse beregner den gjennomsnittlige dybde av nodene, og som til slutt tar gjennomsnittet av dette igjen over alle trærne.

Vi skal imidlertid *ikke* gjøre denne programmeringen, men i stedet konstatere at det i passelige bøker (ogsåi læreboka) står at 'totalsnitt(n)' vokser med $n$ som $O(\sqrt{n})$.

Tenker vi nåpåbinære søketrær, såvet vi at om vi setter $n$ verdier inn i et (påforhånd tomt) søketre såvil hver innsettingssekvens gi et søketre av en bestemt form. I læreboka står det videre at om vi bruker en "tilfeldig innsettingsrekkefølge" og vi kaller den gjennomsnittlige dybden av nodene i det resulterende tre for $d(n)$, såvil $d(n)$ vokse med $n$ som $O(\log n)$.

Dette kan synes som en motsetning, da både 'totalsnitt(n)' og $d(n)$ påen måte representerer den gjennomsnittlige dybden av nodene i et tilfeldig framkommet tre. Din oppgave er ågi en kort utredning om hvordan dette henger sammen, og hvorfor det likevel ikke er noen motsetning. Du skal ikke gi noen komplisert matematisk utredning, men bare fåfram hovedpoenget (5 – 10 linjer?).

*(Fortsettes p side 9.)*

# Del 3 (25%)

Vi skal her se påartikulasjonspunkter m.m. i urettede grafer, og vi skal hele tiden anta at den grafen vi ser påer sammenhengende. Det følgende er en skisse av en prosedyre som finner artikulasjonspunkter. Dette er den samme prosedyren som er angitt i læreboka, bare med den forskjell at den er skrevet i Simula-aktig notasjon og at den ogsåbehandler startnoden riktig (linjene 10 – 14). Vi angir først deklarasjonene som er globale til prosedyren. Linjenummereringen i selve prosedyren skal brukes siden. De engelske navnene tilsvarer de i læreboka.

```
integer counter;        ! Global teller, initialisert til 1;

class node;             ! Nodene i grafen er representert ved denne ;
begin
   boolean visited;     ! Initialisert til FALSE ;
   integer num, low;
   ref(node) parent;    ! Initialisert til NONE ;
   boolean returnert;   ! Brukes bare i startnoden, initialisert til FALSE ;
   <Data-struktur som angir nodens nabo-noder>;
end;

procedure find_art(v); ref(node) v;
begin  ref(node) w;
1   v.visited:= true;
2   v.num:= counter; counter:= counter+1;
3   v.low:= v.num;

4   for w:- <hver av nabo-nodene til v i en eller annen rekkeflge> do
5   begin
6      if not w.visited then
7      begin
8         w.parent:- v;
8         find_art(w);
10        if v.num = 1 then ! Vi er i startnoden ;
11        begin
12           if v.returnert then <Skriv ut at v er et artikulasjonspunkt>
13                          else  v.returnert:= true;
14        end else
15        begin
16           if w.low >= v.num then <Skriv ut at v er et artikulasjonspunkt>;
17           v.low:= min(v.low, w.low);
```

*(Fortsettes p side 10.)*

```
18        end;
19     end else
20     if v.parent =/= w then v.low:= min(v.low, w.num);
21  end;
end;
```

Som generelt eksempel skal vi se påfølgende graf G:

**Oppgave 3-a**

Du skal her tenke deg at prosedyren 'find_art' blir kalt utenfra i noden S. Prosedyren vil da gjøre et dybde-først-søk gjennom grafen. Du kan fritt velge den rekkefølgen prosedyren i linje 4 ser påsine nabo-noder.

Du skal tegne opp en forstørret kopi av grafen, og pådenne skal du angi hvordan dybde-først-søket gikk og hvordan verdiene ble satt. Mer presist skal du gjøre følgende:

- For hver kant, angi den med heltrukken linje om den er med i dybde-først-spenntreet, og angi den stiplet om den er en tilbake-kant (back-edge). Gi ogsåkantene retning ut fra hvordan de (første gang) ble fulgt i løpet av søket.

- For hver node, angi verdiene på'num' og 'low' slik: 'num/low'

- Sett en ekstra ring rundt de noder som er artikulasjonspunkter.

**Oppgave 3-b**

I denne oppgaven er vi ute etter åfinne såkalte artikulasjons-*kanter*. Dette er kanter som, om de fjernes, vil gjøre at grafen deles i to (sammenhengende) komponenter. Grafen over har én artikulasjonskant.

Din oppgave er åangi forandringer i prosedyren 'find_art' slik at den vil gjøre

*(Fortsettes p side 11.)*

utskrift av typen: '<Kanten (u,v) er en artikulasjonskant>' nøyaktig én gang for hver artikulasjonskant (u,v) i grafen. Si gjerne først med ord hva den essensielle forandringen vil være, og angi sånøyaktige de forandringer du vil gjøre i prosedyreteksten ved åreferere til linjenumrene.