Høgskolen i Gjøvik

Avdeling for Teknologi

E K S A M E N

FAGNAVN: **Algoritmiske metoder I**

FAGNUMMER: **L 189 A**

EKSAMENSDATO: **13. desember 1999**

KLASSE: **98HINDA / 98HINDB / 98HINEA**
(2DA / 2DB / 2EA)

TID: **09.00-14.00**

FAGLÆRER: **Frode Haug**

ANT. SIDER UTLEVERT: **5 (inkludert denne forside)**

TILLATTE HJELPEMIDLER: **Alle trykte og skrevne.**

- Kontroller at alle oppgavearkene er tilstede.
- Innføring med penn, evt. trykkblyant som gir gjennomslag.
  Pass på at du ikke skriver på mer enn ett innføringsark om gangen (da det blir uleselige gjennomslag om flere ark ligger oppå hverandre når du skriver).
- Ved innlevering skilles hvit og gul besvarelse og legges i hvert sitt omslag.
  Oppgavetekst, kladd og blå kopi beholder kandidaten.
- Ikke skriv noe av din besvarelse på oppgavearkene.
- Husk kandidatnummer på alle ark.

Oppgave 1 (teori, 25%)

Denne oppgaven inneholder tre uavhengige oppgaver fra kap. 4, 8 og 12 i læreboka.

a) **Er følgende utsagn sant eller usant ?** ”Selve blad-/terminalnodene i et binært tre forekommer i samme relative rekkefølge i forhold til hverandre, uavhengig om treet traverseres på en preorder, inorder eller postorder måte.” **Begrunn svaret.** (”Samme relative rekkefølge” betyr at bladnodene har samme innbyrdes rekkefølge, når man tar vekk/ser bort fra de interne nodene som er innimellom ved en utskrift.)

En annen måte (enn lærebokas fire måter) å traversere et tre på er ved først å visitere rota, deretter høyre subtre, og så til slutt venstre subtre. **Hva vil denne nye traverseringsmetoden ha til felles med en eller flere av de andre traverseringsmetodene i læreboka ?**

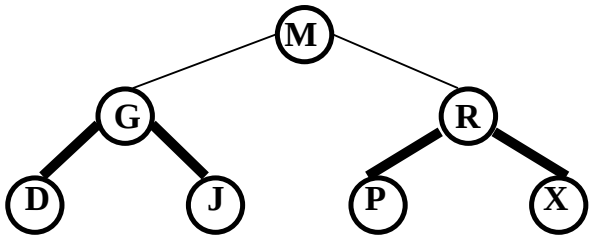
b) Vi skal utføre Shellsort på key’ene ”DILLETANTER”. Jfr. kode s.109 i læreboka.
For hver gang indre for-løkke er ferdig (etter: `a[j] = v;`):
Tegn opp arrayen og skriv verdiene til 'h' og 'i' underveis i sorteringen.
Marker spesielt de key’ene som har vært involvert i sorteringen (jfr. fig.8.7 s.108).

c) Vi skal utføre rekursiv Mergesort på key’ene ”DILLETANTER”. Jfr. kode s.166 i læreboka.
For hver gang tredje og siste for-løkke i koden s.166 er ferdig:
Tegn opp arrayen med de key’ene som har vært involvert i sorteringen (jfr. fig.12.1 s.167).

Oppgave 2 (teori, 25%)

Denne oppgaven inneholder tre uavhengige oppgaver fra kap. 15 og 33 i læreboka og om FSM.

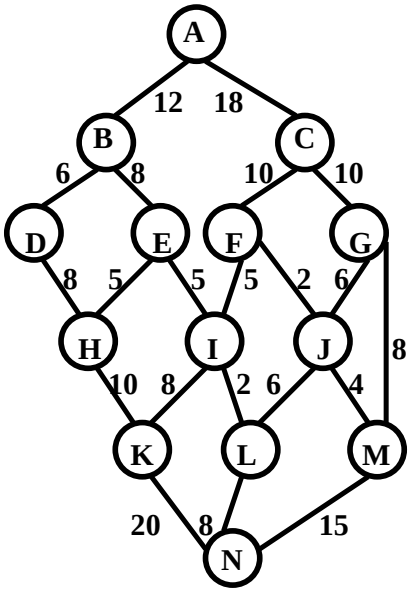
a) Følgende Red (tykke streker)-Black (tynne streker) tre er gitt:



Legg bokstavene ’E’, ’T’ og ’W’ etter tur inn i treet ovenfor.
Tegn opp treet for hver gang en ny bokstav er lagt inn i Red-Black treet.

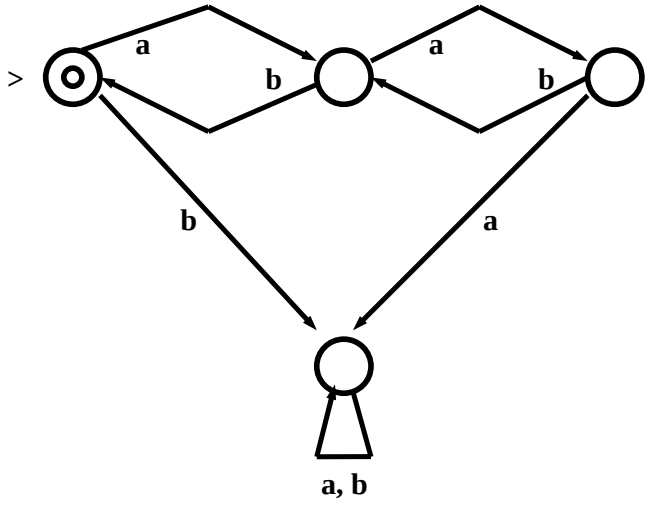
NB: For hver bokstav skal du ikke på nytt ta utgangspunkt i treet ovenfor.
Men alle de tre bokstavene skal til slutt befinne seg i det samme treet.

b) Vi har følgende nettverk (dvs. en graf som er både vekted og retted):



'A' er kilden og 'N' er kummen. Kapasiteten for hvert enkelt rør/kant er skrevet på figuren, og flyten i røret går alltid nedover på figuren. **Tegn figuren opp igjen, men nå med ny verdi på hver kant, som forteller hva som er max. flyt langs hver av kantene (fra kilde til kum).**

c) Følgende deterministiske endelige tilstandsmaskin (FSM) er gitt:



Hva slags setninger/språk godtar den ?

Oppgave 3 (koding, 30%)

I et binært tre så definerer vi "nivået" til en node som avstanden mellom noden og rotnoden. Det betyr at rotnoden har nivå 0, barna til rotnoden har nivå 1, barnebarna har nivå 2, o.s.v. Dersom vi summerer nivåene til alle nodene i et slikt tre og så deler denne summen med antall noder, får vi treets gjennomsnittsnivå. En node er deklarert på følgende måte:

```
struct node  {
    char ID;          //  Nodens id/navn (en bokstav).
    int  nivaa;       //  Nodens nivå i treet ift. rotnoden.
    node* left;       //  Peker til venstre subtre.
    node* right;      //  Peker til høyre subtre.
};
```

a) Når et slikt binært tre blir bygd, vil "nivaa" ikke bli initialisert etterhvert. Dvs. den vil i hver enkelt node kun inneholde tilfeldig "søppel". Unntaket er rota. Dens "nivaa" er satt til '0'.
Lag funksjonen `void sett-nivaa(node* p)` **som traverserer treet rekursivt, og som setter rett nivå i alle nodene i treet.**

NB: Du får ikke lov til å bruke noen ekstra global variabel (som "y" på side 61 i læreboka). Ei heller er det lov å sende med noen flere parametre til funksjonen "sett_nivaa".

b) For å beregne gjennomsnittsnivået i treet må vi lage og kjøre en annen og ny funksjon (etter at funksjonen i oppgave 3a har gjort jobben sin). Denne har følgende prototype/funksjonsheading:

```
void sett-verdier(node* p, int & sum_nivaa, int & antall)
```

Lag denne funksjonen som traverserer treet rekursivt, og som oppdaterer de to referanseoverførte variablene med totalsummen av alle nodenes nivå og totalt antall noder i treet.

NB: Vi forutsetter at de to referansevariablene som initielt medsendes er nullstilt.

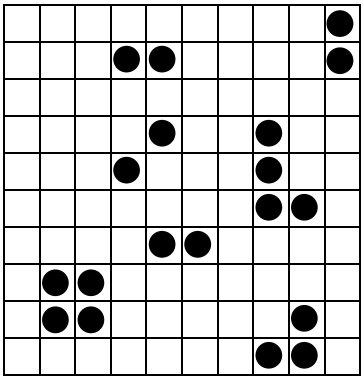
c) La oss nå tenke oss at vi nå har et binært søketre. Hver node er deklarert som ovenfor, bare at nå er "nivaa" uinteressant. **Lag en rekursiv funksjon** `void skriv-forfedre(node* p)` **som traverserer hele treet og for hver blad-/terminalnode skriver (til skjermen) en linje med bladnodens ID og ID'ene til alle dens forfedre (dens "mor", "bestemor", o.s.v.).** La det være ett blankt tegn mellom hver av ID'ene som skrives, og la det være en ny linje for hver bladnode.

NB1: Husk at "en bladnode" er en node som både har et venstre og et høyre z-barn.
Dvs. en bladnode er den nederste/siste noden på enhver grein i treet.

NB2: Det enkleste er nok, for hver bladnode du finner, å skrive alle dens forfedres ID ved å starte i rota og skrive de ID'er man møter på vei ned til den aktuelle bladnoden.

Oppgave 4 (koding, 20%)

Vi har et NxN rutenett. I en del av rutene er det plassert brikker. Dette kan f.eks. se slik ut:



To ruter med brikker i sies å tilhøre den samme gruppen, dersom de har en vannrett eller loddrett strek mellom seg. På figuren ovenfor vil det derfor være to grupper med fire brikker, en gruppe med tre brikker, tre grupper med to brikker og to grupper med en brikke. Anta at rutenettet er representert vha. den to-dimensjonale arrayen: `brett[N+1][N+1]`. Vi bruker indeksene fra 1-N. Initielt inneholder en rute '0' dersom den er tom, og '1' dersom det står en brikke i vedkommende posisjon.

Skriv et komplett program (bl.a. bestående av en eller flere rekursive funksjoner) som finner grupper og for hver av dem:

- **skriver hvilke ruter (identifisert vha. et 'i' og 'j' par) som inngår i gruppen**
- **skriver antall ruter i gruppen (dvs. antall brikker som totalt utgjør gruppen)**

og som til slutt skriver totalt antall ulike grupper i rutenettet.

Hint: Det kan nok være lurt å la "skuffene" inneholde andre verdier enn bare '0' og '1' for å få løst denne oppgaven.

Lykke til !

frode@haug.com