Høgskolen i Gjøvik

Avdeling for Teknologi

KONTINUASJONSEKSAMEN

FAGNAVN: **Algoritmiske metoder I**

FAGNUMMER: **L 189 A**

EKSAMENSDATO: **17. august 2000**

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) Dersom vi får oppgitt preorder og inorder rekkefølgen for nodene i et binært tre, så kan vi som oftest entydig konstruere det binære treets struktur. (Ihvertfall er dette alltid sant om key'ene i alle nodene er ulike/forskjellige/unike.) **Forklar hvorfor.**
Er det samme sant om vi får oppgitt preorder og postorder rekkefølge ? **Begrunn svaret.**
Er det samme sant om vi får oppgitt inorder og postorder rekkefølge ? **Begrunn svaret.**

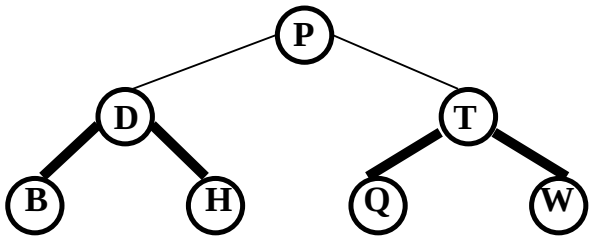
b) Vi skal utføre Shellsort på key'ene "TOMBOLABUA". Jfr. kode s.109 i læreboka.
For hver gang indre for-løkke er ferdig (etter: `a[j] = v;`):
Tegn opp arrayen og skriv verdiene til 'h' og 'i' underveis i sorteringen.
Marker spesielt de key'ene som har vært involvert i sorteringen (jfr. fig.8.7 s.108).

c) Vi skal utføre rekursiv Mergesort på key'ene "TOMBOLABUA". 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.

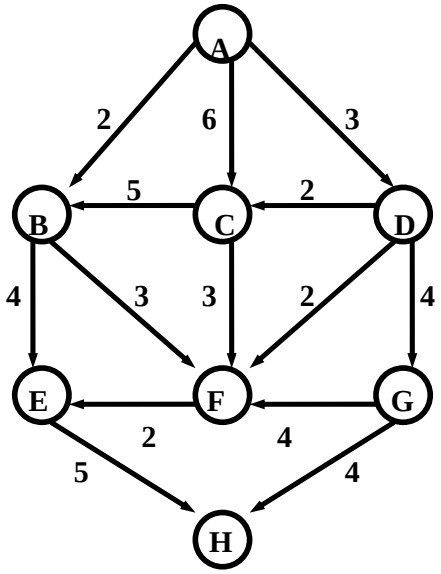
a) Følgende Red (tykke streker)-Black (tynne streker) tre er gitt:



Legg bokstavene 'A', 'P' og 'A' etter tur inn i treet ovenfor.
Tegn opp treet for hver gang en ny bokstav er lagt inn i Red-Black treet.

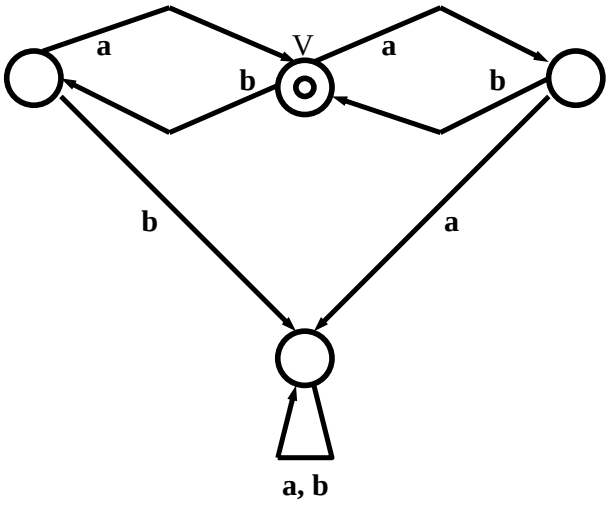
NB: For hver bokstav skal du ikke på nytt ta utgangspunkt i treet ovenfor.
Men alle de tre bokstavene skal til slutt befinne seg i det samme treet.

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



'A' er kilden og 'H' er kummen. Kapasiteten for hvert enkelt rør/kant er skrevet på figuren, og flyten i røret går i pilretningen. **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, 25%)

I denne oppgaven skal du lage to funksjoner som opererer på binære søketrær. Dataene som lagres i hver node i treet er et heltall (’verdi’) og en enkelt bokstav (’id’). Nodene har som vanlig pekere til venstre og høyre barn/subtre. I tillegg har hver node en peker til foreldernoden (”mora”/forgjenger) i treet. Disse pekerne til foreldernoden skal kun brukes i oppgave 3b. Nodene har også en heltallsvariabel ’sum’ som bare skal brukes i oppgave 3a.

En node er derfor deklarert på følgende måte:

```
struct node  {
    char id;          //  Nodens id/navn (en bokstav).
    int verdi;        //  Nodens verdi.
    int sum;          //  Summen av alle barnas verdi.
    node* left;       //  Peker til venstre subtre.
    node* right;      //  Peker til høyre subtre.
    node* forelder;   //  Peker til nodens ”mor”/forelder.
};
```

Hvis venstre og/eller høyre subtre til en node er tomt, vil pekerne ’left’ og/eller ’right’ peke til z-noden. Vi krever ikke at søketreet skal være balansert. Nodene i søketreet ligger ordnet/sortert på variabelen ’verdi’, og treet kan gjerne inneholde duplikate/like noder (med lik ’verdi’).
(Hele søketreet kan nåes gjennom den globale variabelen ’root’.)

a) I denne oppgaven antar vi at variabelen ’sum’ i utgangspunktet er satt til ’0’ i alle nodene i treet. **Lag funksjonen** `void sett_sum(node* p)` **som traverserer treet rekursivt, og som for hver node i søketreet setter variabelen ’sum’ lik summen av alle verdiene i nodens subtrær pluss verdien (av variabelen ’verdi’) i noden selv.**
Parameteren ’p’ peker til roten for det aktuelle subtreet.

b) **Lag funksjonen** `void sett_forelder(node* p)` **som traverserer treet rekursivt, og som for hver node i søketreet setter variabelen ’forelder’ til å peke på nodens forgjenger/ forelder/”mor” i treet.**
Hver nodes pekervariabel ’forelder’ inneholder initielt en vilkårlig adresse (”søppel”). Parameteren ’p’ peker til roten for det aktuelle subtreet, og denne skal få sin ’forelder’ satt til z-noden.

NB: For begge oppgavene gjelder det at det ikke er lov å sende med noen flere parametre til funksjonene !

Oppgave 4 (koding, 25%)

Et firma produserer perlekjeder som består av fargede plastkuler tredd på en snor. Det brukes kuler i 8-(åtte) forskjellige farger. Hver farge svarer til et heltall, på denne måten:

Svart: 1 Lilla: 2 Blå: 3 Grønn: 4 Rød: 5 Orange: 6 Gul: 7 Hvit: 8

Perlekjedene må oppfylle følgende krav:

1. Fargene rød og grønn skal aldri forekomme ved siden av hverandre.
2. Fargene blå og orange skal aldri forekomme ved siden av hverandre.
3. Samme farge skal aldri forekomme mer enn én gang blant tre kuler som ligger rett etter hverandre i perlekjedet.

Et perlekjede som ikke bryter mot noen av disse tre kravene kaller vi et *lovlig* kjede.
For eksempel perlekjedene Svart-Blå-Rød-Gul-Svart og Hvit-Blå-Gul-Hvit-Rød er lovlige, mens perlekjedene Hvit-Rød-Grønn-Svart-Lilla (bryter krav 1) og Blå-Lilla-Svart-Lilla-Grønn (bryter krav 3) er *ulovlige*.

Til å representere perlekjedet og nåværende sammensetning av perlene så bruker vi variablene:

```
const int N = <en positiv realistisk heltallskonstant, f.eks. 10>
int kjede[N];
```

'Kjede' vil i de ulike "skuffene" inneholde heltall i intervallet 1-8 (som representerer de åtte fargene).

Skriv et komplett program (bl.a. bestående av en rekursiv funksjon) som finner (og skriver ut) alle mulige lovlige perlekjeder av lengde N. Skriv også ut antall ulike kjeder som ble generert.

Hint: - Du bør nok bl.a. lage en funksjon `void skriv_ut()` som skriver ut de 'N' "skuffene" i 'kjede', ved å gjøre om tallene til fargene "Svart, Lilla, Blå, …..., Gul, Hvit".

Lykke til !

frode@haug.com