Høgskolen i Gjøvik

Avdeling for elektro- og allmennfag

KONTINUASJONSEKSAMEN

FAGNAVN: **Algoritmiske metoder I**

FAGNUMMER: **L 171 A**

EKSAMENSDATO: **17. august 1998**

KLASSE: **96HINDA / 96HINDE (2DA / 2DB)**

TID: **09.00-14.00**

FAGLÆRER: **Frode Haug**

ANT. SIDER UTLEVERT: **4 (inkludert denne forside)**

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

- **Kontroller at alle oppgavearkene er tilstede.**
- **INNFØRING MED PENN, evnt. 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 (ikke oppgavearkene).**

Oppgave 1 (teori)

Denne oppgaven inneholder tre uavhengige oppgaver fra kap. 11 (a og b) og 15 (c) i læreboka.

a) Følgende prioritetskø, organisert som en heap, er gitt:

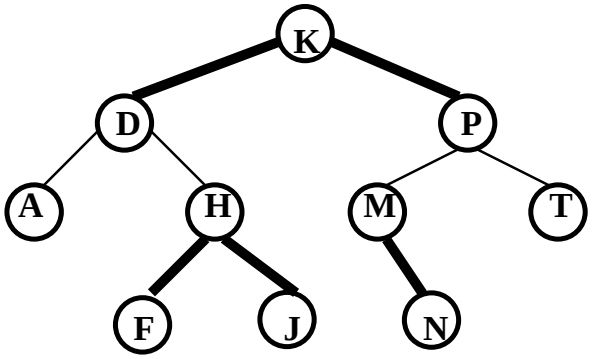
92 84 70 34 29 57 62 13 21 26 7 18

Utfør etter tur følgende operasjoner på denne heap: insert(80), insert(88), remove(), remove() og replace(9). **Tegn opp heapen etter at hver av operasjonene er utført.**

NB: For hver av operasjonene skal du operere videre på den heapen som ble resultatet av den forrige operasjonen.

b) Teksten «H E I A S T O R H A M A R» i denne rekkefølge fra venstre til høyre, blanke regnes ikke med) skal **heap-sorteres vha. bottom-up heap konstruksjon**.
(Se fig.11.8, koden s.156 og fig.11.9 i læreboka.)
Tegn opp heapens innhold etterhvert som heapen konstrueres og deretter sorteres etter denne metoden (Dvs. lag en figur etter samme prinsipp som fig.11.9.)

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



Tegn opp resultatet når bokstavene «O» og «G» legges inn i treet ovenfor.

NB: For hver gang (bokstav) skal du på nytt ta utgangspunkt i treet ovenfor.
Dvs. bokstavene «O» og «G» skal ikke til slutt befinne seg i det samme treet.

Oppgave 2 (teori)

Denne oppgaven inneholder tre uavhengige oppgaver fra kap. 22, 30 og 32 i læreboka.

a) Vis **konstruksjonsprosessen** når bokas metode **for Huffman-koding** (s.324-330) brukes på teksten «SOMMERFERIE OG SOMMERFERIE FRU BLOM SA BLOMSTEN» (inkludert de blanke).
Hvor mange bits trengs for å kode denne teksten ? Dvs. skriv/tegn opp:

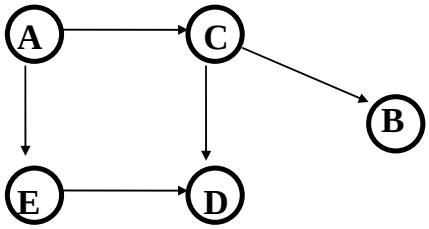
- tabellen for bokstavfrekvensen (jfr. fig.22.3).
- tabellen for «dad»en (jfr. fig.22.6).
- Huffmans kodingstreet/-trien (jfr. fig.22.5 og koden øverst s.328).
- bokstavenes bitmønster, med «code» og «len» (jfr. fig.22.7 og koden øverst s.329).
- totalt antall bits som brukes for å kode teksten.

b) Følgende kanter i en (ikke-retted, ikke-vekted) graf er gitt:

AB AE BC DG EG CF BF CE AD DF BE

Vi skal nå utføre union-find på denne grafen. Tegn opp **arrayen «dad»s innhold etterhvert** som skogen bygges opp (jfr. fig.30.6) vha. «find»-funksjonen s.444 i læreboka. Ut fra dette: tegn også opp den resulterende **union-find skogen** (jfr. nedre høyre skog i fig.30.5).

c) Følgende rettede asykliske (ikke-vektede) graf («dag») er gitt:



Angi **ALLE mulige topologiske sorteringssekvenser** av nodene i denne grafen.

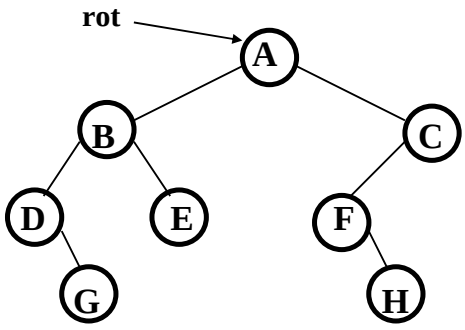
Oppgave 3 (koding)

I hele oppgave 3 tenker vi oss nodene representert på følgende måte:

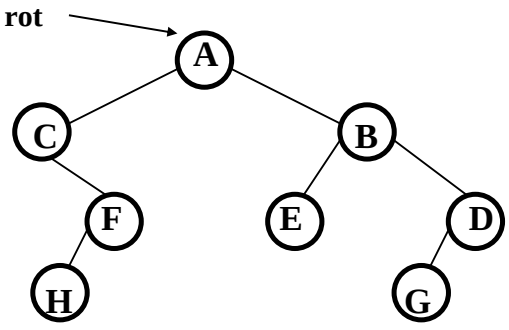
```
struct node  {
      char id;
      node* left;
      node* right;
};
```

a) **Skriv EN rekursiv funksjon:** `void speilvend(node* t);` **som omformer et binært tre med utgangspunkt i noden «rot» til det speilvendte treet.**

For eksempel:



Opprinnelig tre



Speilvendt tre

b) **Skriv EN rekursiv funksjon:** `bool identisk(node* t, node* u);` **som returnerer «true» dersom trærne tilpekt av «t» og «u» er identiske (ellers returneres «false»).**

NB: Trærne er identiske hvis og bare hvis de er strukturelt identiske og at innholdet («id») i korresponderende noder er det samme.

Oppgave 4 (koding)

N personer skal fordele N jobber seg imellom. Av ulike årsaker (f.eks. annonsering, intervjuer, lønn, frikjøp fra forrige arbeidsgiver eller lønnet studietid (!)) er det ulike kostnader/utgifter med å la hver enkelt person få de ulike jobbene. Kostnaden med å la person nr. i få jobb nr. j er å finne i «array-skuffen» `kostnad[i][j]`. **Du skal skrive et fullstendig program som sørger for at alle personene får hver sin jobb, samtidig som totalkostnadene minimeres. Programmet skal primært være basert på en rekursiv funksjon som gjør arbeidet med å finne den optimale fordelingen.**

NB: Alle oppgavene teller likt, dvs. 25% vekt på hver av dem !

Algoritmisk lykke til !
FrodeH