Høgskolen i Gjøvik
Avdeling for informatikk og medieteknikk

E K S A M E N

EMNENAVN: **Algoritmiske metoder**

EMNENUMMER: **IMT2021**

EKSAMENSDATO: **19. desember 2008**

KLASSE(R): **07HB - INDA / PUA / DRA / ISA**

TID: **09.00-14.00**

EMNEANSVARLIG: **Frode Haug**

ANTALL SIDER UTLEVERT: **4 (inkludert denne forside)**

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

- Kontroller at alle oppgavearkene er til stede.
- Innføring med penn, eventuelt trykkblyant som gir gjennomslag. Pass på så du ikke skriver på mer enn ett innføringsark om gangen (da det blir uleselige gjennomslag når flere ark ligger oppå hverandre).
- Ved innlevering skilles hvit og gul besvarelse, som legges i hvert sitt omslag.
- Oppgavetekst, kladd og blå kopi beholder kandidaten til klagefristen er over.
- Ikke skriv noe av din besvarelse på oppgavearkene. Men, i oppgavetekst der du skal fylle ut svar i tegning/tabell/kurve, skal selvsagt dette innleveres sammen med hvit besvarelse.
- Husk kandidatnummer på alle ark. Ta vare på dette nummeret til sensuren faller.

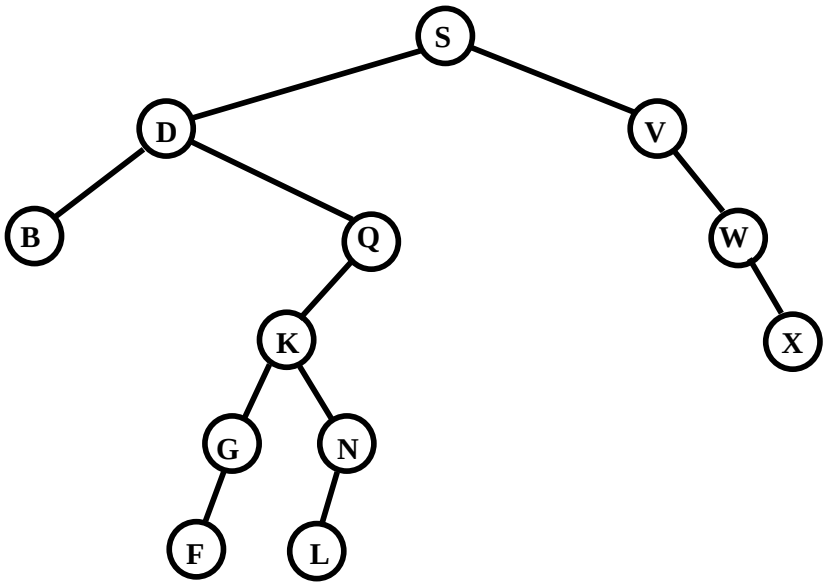
Oppgave 1 (teori, 20%)

Denne oppgaven inneholder tre uavhengige oppgaver fra sortering, kap.12 og 14 i læreboka.

a) N tall skal sorteres. Disse er *kun* av *to ulike* verdier (f.eks. 71 og 312). Vi ønsker at sorteringen skal være av orden O(N). **Hvilke(n) metode(r) i læreboka kun du bruke direkte slik de(n) står? Og hvilke(n) kan du *meget* enkelt justere *litt* på, slik at de(n) kan brukes?**
I det siste tilfellet: Evt. forklar kort hva som må endres i vedkommende metode(r).

b) Vi skal utføre *rekursiv* Mergesort på key'ene "STRAFFEKAST". 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).

c)



Du skal nå fjerne («remove») noen noder fra dette treet. **Tegn opp treet for hver gang og fortell hvilken av «if ..... else if ..... else»-grenene i koden s.210 som er aktuelle når du etter tur fjerner henholdsvis tegnene 'V', 'Q' og 'D'.**

NB: Du skal for hver fjerning *på nytt* ta utgangspunkt i treet ovenfor.
Dvs. på intet tidspunkt skal det fra treet være fjernet to/tre av bokstavene samtidig.

Oppgave 2 (teori, 30%)

Denne oppgaven inneholder tre uavhengige oppgaver fra kap.22, 30 og 31 i læreboka.

a) Vis **konstruksjonsprosessen** når bokas metode **for Huffman-koding** (s.324-330) brukes på teksten "VAFFELSTEKER STYRK STERK STEKTE VAFLER FRA VAFFELRORA HAN HADDE" (inkludert de blanke – husk den etter "FRA").
Hvor mange bits trengs for å kode denne teksten? Dvs. skriv/tegn opp:

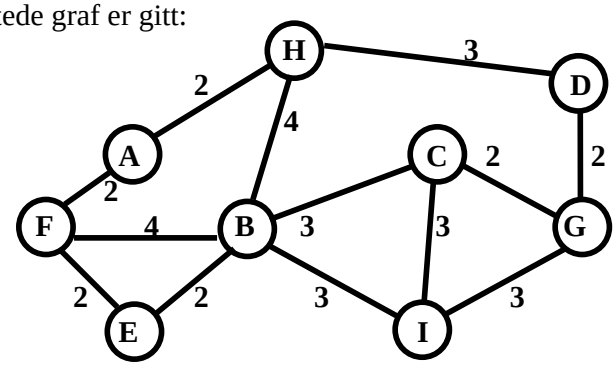
- tabellen for bokstavfrekvensen (jfr. fig.22.3).
- tabellen for "dad"en (jfr. fig.22.6).
- Huffmans kodingstreet/-trien (jfr. fig.22.5 og koden øverst s.328).
- bokstavenes bitmønster og "len" (jfr. fig.22.7 og koden øverst s.329).
- totalt antall bits som brukes for å kode teksten.

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

EF DC BC BF AC GH GC FA BH

Vi skal nå utføre *union-find m/path compression og weight balancing* på denne grafen. **Tegn opp arrayen "dad"s innhold etterhvert** som skogen bygges opp (jfr. fig.30.9) vha. "find"-funksjonen s.447 i læreboka. Ut fra dette: **tegn også opp den resulterende union-find skogen** (dvs. noe lignende til nedre høyre skog i fig.30.8).

c) Følgende vektede graf er gitt:



Hver nodes naboer er representert i en *naboliste*, der disse er *sortert alfabetisk.*
Koden s.455 i læreboka utføres på denne grafen – der "priority" er lik "val[k] + t->w".
Hvilke kanter er involvert i korteste-sti spenntreet fra "I" og til *alle* de andre nodene?
Tegn også opp innholdet i prioritetskøen etterhvert som koden utføres (jfr. fig.31.13 i læreboka). *NB:* Husk at ved lik prioritet så vil noden sist oppdatert (nyinnlagt eller endret) havne først i køen.

Oppgave 3 (koding, 25%)

Vi har et *binært tre* (*ikke* søketre) bestående av nodene:

```
struct Node  {
  int   ID;       //  Nodens ID/key/nøkkel/navn (et tall).
  Node* left;     //  Peker til venstre subtre, evt. z-noden om det er tomt.
  Node* right;    //  Peker til høyre subtre, evt. z-noden om det er tomt.
  Node (int id, Node* l, Node* r)  { ID = id; left = l; right = r; }
};
```

Vi har også de to globale variablene:

```
Node* z   = new Node(0, NULL, NULL);  //  z-noden.
Node* rot = z;                        //  Rot-peker.
int   x;                        //  Innlest og ønsket verdi fra brukeren.
```

a) Vi forutsetter at treet *er* organisert slik at det tilfredsstiller heap-betingelsen (at enhver "mor" er større enn eller lik begge barna sine).
Lag den *rekursive* funksjonen `int tellStorre(Node* p, int v)`
Funksjonen skal (i treet tilpekt av `p`) returnere antall noder med `ID` *større enn* `v`.
Den startes fra `main` med følgende kall: `cout << tellStorre(rot, x);`
Legg vekt på å besøke så få noder som mulig, dvs. gjør avskjæring så fort dette er greit.

b) Nå vet vi ikke noe om hvordan treets verdier er organisert.
Lag den rekursive funksjonen `bool erHeapOrdnet(Node* p)`
Funksjonen skal (i treet tilpekt av `p`) returnere `true/false` til om treet tilfredsstiller heap-betingelsen (se oppgave 3a) eller ei.
Den startes fra `main` med følgende kall: `bool heap = erHeapOrdnet(rot);`

Oppgave 4 (koding, 25%)

Følgende const'er og variable er definert (bruk disse *aktivt* i koden du skriver!):

```
const int DELTAGERE  =    50; // Antall deltagere (på konferansen).
const int BUDSJETT   = 50000; // Budsjett for hotelloppholdet for alle deltagerne.
const int HOTELL     =     4; // Antall hoteller.
const int UKER       =     6; // Antall uker.
int pris[HOTELL+1];           // Totalprisen for oppholdet på hotell nr.i.
                              // For en deltager, hele oppholdet, samme pris alle uker.)
int ledig[HOTELL+1][UKER+1];  // Antall ledige enmanns-rom på de ulike hotellene,
                              //      de ulike aktuelle ukene.
```

Du skal lage kode som finner ut hvilket hotell og hvilken uke `DELTAGERE` konferansedeltagere med et totalbudsjett for selve hotelloppholdet på kroner `BUDSJETT` bør bestille for at deres *samlede* utgift for hotelloppholdet skal bli lavest mulig.
Utskrift skal være: Hotellnummer, ukenummer og totalsummen de må betale til hotellet,
evt. en melding om at det ikke var mulig å finne et billig nok hotell med nok ledige enmanns-rom.

Hint: Løsningen trenger/bør *ikke* være rekursiv! All koden du skriver kan like gjerne ligge i `main`.
NB1: Hvordan de to arrayene (`pris` og `ledig`) blir fylt med verdier/data trenger du ikke å tenke på – de bare inneholder relevante verdier.
NB2: Er det flere alternativer som er billigst (totalsum og ukenr), skal den første velges/presenteres.
NB3: I ingen av arrayene bruker vi indeks nr.0 (i noen av dimensjonene).
NB4: De aktuelle ukene har altså fortløpende nummer i intervallet 1-`UKER`. At dette i *praksis* f.eks. er ukenumrene 39-44 trenger du ikke å tenke på (det finner brukeren av programmet ut av).

Løkke tæll!

frode@haugianerne.no