Høgskolen i Gjøvik
Avdeling for informatikk og medieteknikk

Kontinuasjonseksamen

EMNENAVN: **Algoritmiske metoder**

EMNENUMMER: **IMT2021**

EKSAMENSDATO: **13. august 2008**

KLASSE(R): **06HBINDA / 06HBPUA / 06HBISA / ……**

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, 25%)

Denne oppgaven inneholder tre uavhengige oppgaver fra kap. 8, 12 og 15 i læreboka.

a) Vi skal utføre Shellsort på key'ene "KOPPEKAFF". 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' (4 og 1) og 'i' underveis i sorteringen.
Marker spesielt de key'ene som har vært involvert i sorteringen (jfr. fig.8.7 s.108).

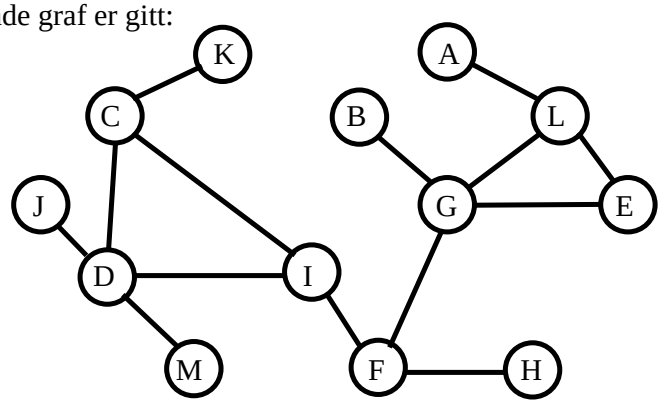
b) Vi skal utføre rekursiv Mergesort på key'ene "KAFFEKOPP". Jfr. kode s.166 i læreboka.
For hver gang tredje og siste for-løkke i koden s.166 er ferdig:
Tegn opp arrayen med de key'ene som har vært involvert i sorteringen (jfr. fig.12.1 s.167).

c) Legg key'ene "KAFFEKOPPHANK" (i denne rekkefølge fra venstre til høyre) inn i et 2-3-4 tre. **Tegn opp treet etterhvert som bokstavene legges inn.**
Gjør også om sluttresultatet til et Red-Black tre.

Oppgave 2 (teori, 25%)

Denne oppgaven inneholder tre uavhengige oppgaver fra kap.30, 22 og FSM.

a) Følgende graf er gitt:



Grafen er representert som/i en nabomatrise.
Startende i node 'D' - **tegn opp dybde-først søketreet for denne grafen.**
Forklar ut fra dette treet hvilke node(r) som er *artikulasjonspunkt(er)* i grafen.

b) Vis **konstruksjonsprosessen** når bokas metode **for Huffman-koding** (s.324-330) brukes på teksten "STORHAMAR ER BARE ET HELT STORT STORLAG I BYDELEN STORHAMAR I HAMAR BY" (inkludert de blanke – husk den etter "BYDELEN").
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.

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

$M = (\{ q_0, q_1, q_2, q_3, q_4 \}, \{ a, b \}, \delta, q_0, \{q_1, q_2, q_3 \})$

Transisjonstabellen:

$\delta(q_0, a) = q_1$	$\delta(q_0, b) = q_4$
$\delta(q_1, a) = q_3$	$\delta(q_1, b) = q_2$
$\delta(q_2, a) = q_4$	$\delta(q_2, b) = q_2$
$\delta(q_3, a) = q_3$	$\delta(q_3, b) = q_4$
$\delta(q_4, a) = q_4$	$\delta(q_4, b) = q_4$

Tegn tilstandsmaskinen. **Hva (slags setninger/språk) godtar den ?**

Oppgave 3 (tegning og koding, 25%)

Vi har et *binært søketre* bestående av nodene:

```
struct Node  {
  char  ID[20];   //  Nodens ID/key/nøkkel/navn (en tekst).
  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 (char* id, Node* l, Node* r) { strcpy(ID, id); left = l; right = r; }
};
```

Vi har også de to globale variablene:

```
Node* z   = new Node("NULL", NULL, NULL);  //  z-noden.
Node* rot = z;                             //  Rot-peker.
```

a) **Tegn treet når følgende noder (i gitte rekkefølge) er vanlig alfabetisk lagt inn:**
”Marita", "Mari", "Marit", "Ola", "Olav", "Ole", "Frode", "Frank", "Frida" og "Marianne".

Vi omdefiner herved (og i *resten av oppgaven*) måten treet er sortert på:
En tekst er mindre enn en annen når dets lengde er mindre enn en annen, eller om tekstene er like lange, sorteres de vanlig alfabetisk.
Tegn treet når de samme verdiene er lagt inn, men etter den nye sorteringensmetoden.

b) **Lag funksjonen** `Node* search(Node* p, char* t)`
Funksjonen skal (i treet tilpekt av ’p’, og sortert etter den nye metoden) lete etter en node med ID lik ’t’, og evt. returnere en peker til denne, ellers returneres NULL.
Den startes fra `main` med følgende kall: if (`search(rot, ”……”))` …….

c) **Lag funksjonen** `void insert(Node* p, char* t)`
Funksjonen skal (i treet tilpekt av ’p’, og sortert etter den nye metoden) sette inn en ny node med ID lik ’t’. Den startes fra `main` med følgende kall: `insert(rot, ”……”));`

Hint: Ifm. både oppgave 3b og 3c *kan* det være lurt å lage en felles hjelpe-funksjon (som brukes av begge funksjonene), som hjelper til med å avgjøre om en tekst er mindre enn en annen eller ei (ut fra de nye sorteringskriteriene).

Oppgave 4 (koding, 25%)

Vi har følgende regnestykke: TWO * TWO = SQUARE

Oppgaven går ut på å skrive et *komplett* program som finner og skriver ut *alle* løsninger der bokstavene er erstattet av et *utvalg* av sifrene 0-9.

Samme bokstav skal erstattes med samme unike siffer. 'S' og 'T' kan begge *ikke* være sifferet '0'. Programmet trenger/bør ikke være rekursiv, men legg vekt på hurtighet, effektivitet og enkelhet!

Løkke tæll!

frode@haugianerne.no