

Institutt for datateknologi og informatikk

Eksamensoppgave i IDATG2102 – Algoritmiske metoder

Faglig kontakt under eksamen:

Frode Haug

Tlf:

950 55 636

Eksamensdato:

5.desember 2022

Eksamenstid (fra-til):

13:00-17:00 (4 timer)

Hjelpemiddelkode/Tillatte hjelpemidler:

F - Alle trykte og skrevne.
(kalkulator er *ikke* tillatt)

Annen informasjon:

Målform/språk:

Bokmål

Antall sider (inkl. forside):

4

Informasjon om trykking av eksamensoppgaven

Originalen er:

1-sidig ☒ 2-sidig ☐

sort/hvit ☒ farger ☐

Skal ha flervalgskjema ☐

Kontrollert av:

Dato

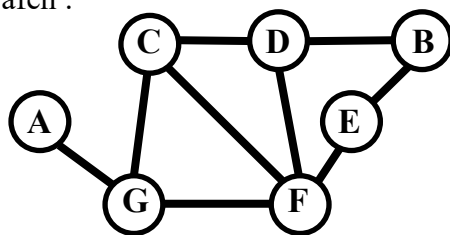
Sign

Oppgave 1 (teori, 25%)

Denne oppgaven inneholder tre totalt uavhengige oppgaver fra pensum.

- a) 8 Skriv/tegn det resulterende 2-3-4 treet når bokstavene: NEUSCHWANSTEIN settes inn i det. Gjør også om sluttresultatet til et Red-Black tre.
- b) 9 Quicksort skal utføres på bokstavene/keyene: OBERSTDORF
Lag en oversikt/tabell der du for hver rekursive sortering skriver de involverte bokstavene og markerer/uthever hva som er partisjonselementet.

- c) 8 Vi har grafen :



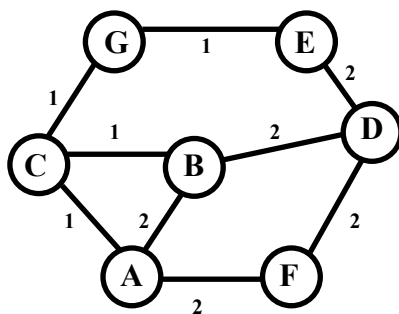
Skriv opp nabomatriksen.

Skriv/tegn dybde-først-søketreet, ved bruk av nabomatriksen og når koden DFS (...) i EKS_30_DFS_BFS.cpp brukes/kjører, og vi starter i node D.

Oppgave 2 (teori, 25%)

Denne oppgaven inneholder tre totalt uavhengige oppgaver fra pensum.

- a) 8 Følgende vektete (ikke-rettete) graf er gitt:



Vi bruker nabomatrikse, og starter i node D. Skriv/tegn opp minimums spenn-treet (MST) for denne grafen, etter at koden i EKS_31_MST.cpp er utført/kjørt.

Skriv også opp innholdet i/på fringen etterhvert som konstruksjonen av MST pågår.

NB: Husk at ved lik vekt så vil noden sist oppdatert (nyinnlagt eller endret) havne først på fringen ift. andre med den samme vekten.

b) 8 Vi har rutenettet med S(tart)- og M(ål)-ruter:

1 S	2		4	5	6
7		9	10	11	12
13 x		15 x	16	17 x	
19	20 x	21		23	24 M
25		27		29	30
31	32	33		35	36

Hva vil den minste f-verdien i rutene (x) 13, 20, 15 og 17 kunne være når det *kun* kan gå opp/ned/høyre/venstre (ikke på skrå) med en vekt på 1 (en), og som heuristikk brukes Manhattan distanse (summen av antall ruter horisontalt og vertikalt til og inkludert målet).

c) 9 Vi har følgende bokstaver og forelder-array ifm. Huffman-koding

(frekvens-arrayen er ukjent/irrelevant):

char(k)	' '	A	B	E	G	I	L	N	S	U
k	0	1	2	5	7	9	12	14	19	21
forelder[k]	32	29	-28	31	28	-27	-34	-30	27	33
k	27	28	29	30	31	32	33	34	35	
forelder[k]	30	-29	-31	-32	-33	34	35	-35	0	

Skriv/tegn opp Huffmans kodingstreet/-trien. Skriv bokstavenes bitmønster.

Vi har også følgende bitstrøm, som er kodet etter denne trien:

101001010111001100101001000111101100111010010111010000

Hva står i denne meldingen?

Oppgave 3 (koding, 30%)

Vi har et binært tre bestående av nodene:

```
struct Node {
    int ID; // Nodens ID/key/nøkkel/navn (et tall).
    Node *left, *right; // Referanse til begge subtrærne (evt. nullptr/NULL).
    Node* parent; // Peker oppover igjen til forelder/mor,
}; // evt. nullptr/NULL om noden er rota.
```

Legg merke til parent, som *alltid* peker til nodens mor/forelder (rotas parent er nullptr/NULL). Ifm. de to funksjonene du skal lage nedenfor, kan du forutsette at parameteren *n* ikke peker til nullptr/NULL. **Hint:** Tegn opp et litt større tilfeldig binært (søke)tre, så er det lettere å studere/tenke på hvordan de ulike funksjonene skal operere.

NB: I *hele* oppgave 3 skal det *ikke* innføres globale data eller flere `struct`-medlemmer enn angitt ovenfor. Det skal heller *ikke* brukes andre hjelpestrukturer - som f.eks. array, stakk, kø eller liste.

a) Lag den ikke-rekursive funksjonen `Node* nestePreorder(const Node* n)`
Funksjonen mottar pekeren `n` som parameter. Denne peker til en helt vilkårlig node ett eller annet sted i treet. Funksjonen skal returnere en peker til den *neste* noden i *preorder rekkefølge*. Er `n` selv den siste noden i treet, så returneres `nullptr/NULL`.

Hint: Har en node venstre og/eller høyre barn, så er dette rimelig enkelt. Har den *ikke* det (altså `n` er selv en bladnode), må det letes oppover i treet igjen etter nærmeste høyre-node/-subtre som er ubesøkt.

b) Lag den ikke-rekursive funksjonen `Node* forrigePreorder(const Node* n)`
Funksjonen mottar pekeren `n` som parameter. Denne peker til en helt vilkårlig node ett eller annet sted i treet. Funksjonen skal returnere en peker til den *forrige* noden i *preorder rekkefølge*. Er `n` selv rota, så returneres `nullptr/NULL`.

Hint: Er `n` rota, et venstre barn eller mor har ingen venstre, så er dette rimelig enkelt. Er ikke noe av dette tilfelle, så er *forrige* i *preorder rekkefølge* en bladnode!

Oppgave 4 (koding, 20%)

Vi skal se på tall `n`, der summen av alle heltalllige divisorer til `n` er *større enn* `2n`.

Tre eksempler:

12 sine divisorer er 1, 2, 3, 4, 6 og 12, der $1+2+3+4+6+12 = 28$ $28 > 24 (= 2*12)$

70 sine divisorer er 1, 2, 5, 7, 10, 14, 35 og 70, der $1+2+5+7+10+14+35+70 = 144$ $144 > 140 (= 2*70)$

168 sine divisorer er 1, 2, 3, 4, 6, 7, 8, 12, 14, 21, 24, 28, 42, 56, 84 og 168
der $1+2+3+4+6+7+8+12+14+21+24+28+42+56+84+168 = 480$ $480 > 336 (= 2*168)$

Skriv et program (`main`) som finner og skriver ut *antall* slike tall under 1.000.000 (1 million) der differansen mellom summen av alle tallets divisorer og `2n` er heltallelig kvadratisk.

(For de to første eksemplene over er differansen på summen og to ganger tallet lik 4, som er 2^2
I det tredje eksemplet er differansen 144, som er 12^2)

Litt hjelp: Biblioteksfilen `cmath` i C++ inneholder bl.a. funksjonen `sqrt(y)` for å beregne kvadratroten av `y`. Svaret er en `float`, som altså må finnes ut om er det samme som heltall også.

NB: I *hele* dette oppgavesettet skal du *ikke* bruke kode fra (standard-)biblioteker (slik som bl.a. STL og Java-biblioteket). Men de vanligste inkluder/import du brukte i 1.klasse er tilgjengelig. Koden kan skrives valgfritt i C++ eller Java.

Løkke tæll!
FrodeH