

Institutt for datateknologi og informatikk

Ekstra eksamensoppgave i IDATG2102 – Algoritmiske metoder

Faglig kontakt under eksamen: Frode Haug
Tlf: 950 55 636

Eksamensdato: 29.mars 2023
Eksamenstid (fra-til): 09:00-13:00 (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 Shellsort skal utføres på bokstavene «SMITTEVERN». For hver gang indre for-løkke i eksemplet med Shellsort er ferdig (dvs. rett etter: $a[j] = \text{verdi};$) :
Skriv/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.
- b)** 12 I de følgende deloppgaver er det key'ene "S M I T T E V E R N"
(i denne rekkefølge fra venstre mot, og blanke regnes *ikke* med) som du skal bruke.
For alle deloppgavene gjelder det at den initielle heap/tre er *tom* før første innlegging ("Insert") utføres. Skriv/tegn den resulterende datastruktur når key'ene legges inn i:
- 1) en heap
 - 2) et binært søketre
 - 3) et 2-3-4 tre
 - 4) et Red-Black tre
- c)** 5 Skriv/tegn opp Merkle treet som er basert på 7 blokker.

Oppgave 2 (teori, 25%)

Denne oppgaven inneholder tre totalt uavhengige oppgaver fra pensum.

- a)** 8 I forbindelse med dobbelt-hashing har vi teksten «SMITTEVERNET» og de to hash-funksjonene $\text{hash1}(k) = k \bmod 17$ og $\text{hash2}(k) = 6 - (k \% 6)$ der k står for bokstavens nummer i alfabetet (1-29). Vi har også en array med indeksene 0 til 16.
Skriv hver enkelt bokstav sin k -verdi og returverdi fra både hash1 og hash2 .
Skriv også opp arrayen hver gang en bokstav hashes inn i den.
- b)** 9 Følgende kanter i en (ikke-rettet, ikke-vektet) graf er gitt:
FA DB DC FC BE CA
Utfør Union-Find *m/weight balancing (WB)* og *path compression (PC)* på denne grafen.
Skriv/tegn opp innholdet i gForeldre etter hvert som unionerOgFinn2 kjøres/utføres. Bemerk hvor WB og PC er brukt.
Skriv/tegn også opp den resulterende union-find skogen.

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

1		3	4	5	6	7
8	9	10				14
15	16	17	18	19		21
			25	26	27	28
29	30	31	32	33		
36			39		41	42
43	44	45	46	47	48	49

Hva vil den minste f-verdien i rutene (x) 10, 25, 46 og M 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).

Oppgave 3 (koding, 26%)

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

```
struct Node {
    int ID; // Nodens ID/key/nøkkel/navn (et tall).
    Node *left, *right; // Referanse til begge subtrærne (evt. 'nullptr/NULL').
    Node(int id) { ID = id; left = right = nullptr; }
};
```

Vi har *kun* den globale variabelen:

```
Node* gRoot = nullptr; // Rot-peker (har altså ikke at head->right er rota).
```

Det skal her lages/kodes to *helt* uavhengige funksjoner.

Begge funksjonene kalles initielt fra main med gRoot som parameteren t.

Hvordan et tre har blitt bygd/satt opp, og hvordan pekere ellers er satt, trenger du ikke å tenke på.

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

a) Lag den rekursive funksjonen

```
bool erSosken(const Node* t, const Node* s1, const Node* s2)
```

Funksjonen skal sjekke og returnere (true/false) om nodene tilpekt/referert av s1 og s2 er søsken under den samme moren/foreldret (t) eller ei. Hvem av dem som evt. peker til den venstre eller høyre noden er ikke definer/klart.

b) Lag den rekursive funksjonen

```
void kappTreNedentil(Node* t, const int verdi)
```

Funksjonen skal kutte forbindelsen fra `t` til alle barn som har en ID *større eller lik* verdi. Dvs. sette aktuelle pekere i `t` til `nullptr`.

NB: At det videre nedover i de utkuttete subtrærne evt. er IDer som er mindre enn verdi tar vi ikke hensyn til. At det i C++ på denne måten egentlig oppstår en memory-lekkasje, ved at det ikke sies `delete` om alle nodene i subtrærne, tar vi heller ikke hensyn til.

Vi forutsetter at rota ikke har en verdi som gjør at den skal kuttes vekk/ut.

Oppgave 4 (koding, 24%)

a) Lag den ikke-rekursive funksjonen

```
char forsteSingleBokstav(const char tekst[], const int n)
```

som finner og returnerer den *første* bokstaven i `tekst` som *forekommer bare en gang*.

`n` er antall tegn i teksten. Det er her i oppgave 4a) *ikke* lov til å bruke noen hjelpestrukturer - som f.eks. array, stakk, kø eller liste. Funksjonen blir derfor sikkert av $O(n^2)$.

`tekst` inneholder *kun* bokstavene A-Z (*kun* store bokstaver, og *ingen* blanke/space).

Finnes det *ingen* bokstav som forekommer *bare en gang*, returneres `' '` (blank/space).

b) Lag en ny versjon av funksjonen i oppgave 4a) som returnerer det samme.

Men denne gangen er det lov til å bruke *en* hjelpearray. Funksjonen bør bli av $O(\frac{3}{2}n)$.

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

Løkke tæll!
FrodeH