Eksamen 2018 – Ordinær- Løsningsforslag

Sensureringen av eksamensbesvarelser i DATS2300 høsten 2018 har tatt utgangspunkt i den generelle kvalitative beskrivelsen for karakterene er som følger (Universitet og høyskolerådet, 2004):

A - fremragende - Fremragende prestasjon som klart utmerker seg. Kandidaten viser svært god vurderingsevne og stor grad av selvstendighet.

B - meget god - Meget god prestasjon. Kandidaten viser meget god vurderingsevne og selvstendighet.

C - god - Jevnt god prestasjon som er tilfredsstillende på de fleste områder. Kandidaten viser god vurderingsevne og selvstendighet på de viktigste områdene.

D - nokså god - En akseptabel prestasjon med noen vesentlige mangler. Kandidaten viser en viss grad av vurderingsevne og selvstendighet.

E - tilstrekkelig - Prestasjonen tilfredsstiller minimumskravene, men heller ikke mer. Kandidaten viser liten vurderingsevne og selvstendighet.

F - ikke bestått - Prestasjon som ikke tilfredsstiller de faglige minimumskravene. Kandidaten viser både manglende vurderingsevne og selvstendighet.

Eksamen høst 2018 besto av fem oppgaver som har blitt vektet likt. Ved sensurering har det blitt lagt vekt på hva kandidaten viser av forståelse av kursets pensum opp mot læringsutbyttet. Det vil si at det vektlegges hvordan kandidaten kommer frem til et svar ved å bruke pensum. Under følger en gjennomgang av oppgavene og mulig svar som vil gi god uttelling mot karakterbeskrivelsene over.

Pensum

Pensum i kurset er dekket av online-kompendiet til Ulf Uttersrud,
https://www.cs.hioa.no/~ulfu/appolonius

Alle tema som har blitt tatt opp i ukeoppgaver, forelesninger, og obligatoriske oppgaver er en del av pensum.

Læringsutbytte

Etter å ha gjennomført dette emnet har studenten følgende læringsutbytte, definert i kunnskap, ferdigheter og generell kompetanse.

Kunnskap

Studenten kan:

- forklare oppbyggingen og hensikten med datastrukturer som tabeller, lister, stakker, køer av ulike typer, heaper, hashtabeller, trær av ulike typer, grafer og filer
- gjøre rede for virkemåten og effektiviteten til ulike varianter av algoritmer for opptelling, innlegging, søking, sletting, traversering, sortering, optimalisering og komprimering

Ferdigheter

Studenten kan:

- designe, implementere og anvende datastrukturer for ulike behov
- analysere, designe, implementere og anvende de algoritmene som trengs for å løse konkrete oppgaver
- bruke både egenutviklede og standardiserte algoritmer og datastrukturer til å løse sammensatte og kompliserte problemer

Generell kompetanse

Studenten kan:

- delta i diskusjoner og gi råd om hvilke datastrukturer og algoritmer det er mest hensiktsmessig å bruke i ulike situasjoner
- formidle viktigheten og nødvendigheten av å bruke gode strukturer og effektive algoritmer i programmeringsprosjekter

Oppgave 1: Rekursjon og binærtrær

a) i) F C A B D E I H G J, ii) A B C D E F G H I J, iii) B A E D C G H J I F

b) En rekursiv metode er en metode som kaller seg selv.

i. 1) Ting kan ikke gjentas – ett kall til metoden med ett sett med argumenter kan kun forekomme en gang. 2) All rekursjon må ende i et basistilfelle som er eksplisitt håndtert.

ii. Funksjonen vil kunne gå i evig løkke (føre til stack overflow).

c)

```
    /**
     * Skriver ut inorden ved hjelp av rekursjon
     */
    public void printInOrder() {
        if (this.left_child != null) {
            this.left_child.printInOrder();
        }

        System.out.print(this.value + ", ");

        if (this.right_child != null) {
            this.right_child.printInOrder();
        }
    }
}
```

d) Du kan bruke en stack til å traversere treet istedenfor rekursjon.

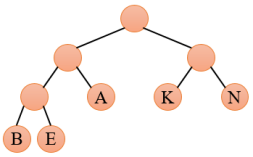
Oppgave 2: Huffmantrær

a) A: 3, B: 1, E: 1, K: 2, N: 2.

b) Av alle trær i Huffmanskogen er det kanoniske venstreorienterte treet der hvert nivå er fylt inn fra venstre og sortert alfabetisk.

c)

i.



ii. B: 111, E: 110, A: 10, K:01, N:00

d)

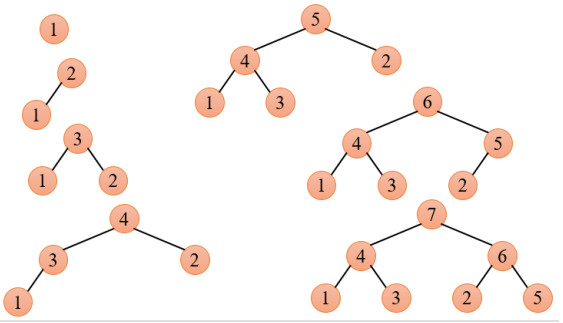
i. 111 10 00 10 00 01 10 01 110

ii. Meldingen bruker 20 bit

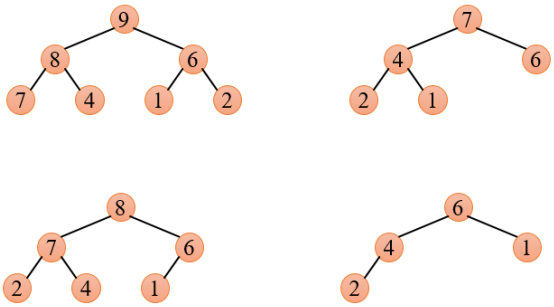
Oppgave 3: Maksimumsheap

a) Komplett binærtre hvor foreldrenoden er større enn eller lik sine barn

b)



c)



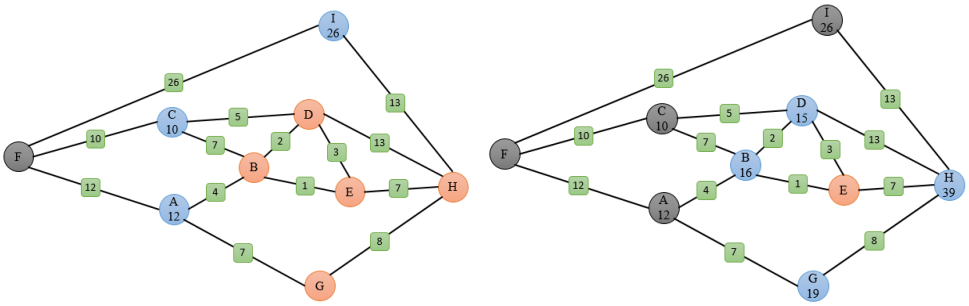
d) Bruk nodeindeksen til hver node som tabell-indeks og lagrer heapen i en tabell. Ta så ut alle tallene fra heapen ved å bytte med siste plass. Hvis vi bruker maxheapen fra deloppgave c så bytter vi først plass med 9 og 2, før vi setter 2 på riktig plass. Vi fortsetter med å bytte 8 og 1 og setter 1 på riktig plass i heapen etc. Dette er generelt sett referert til som heapsortering.

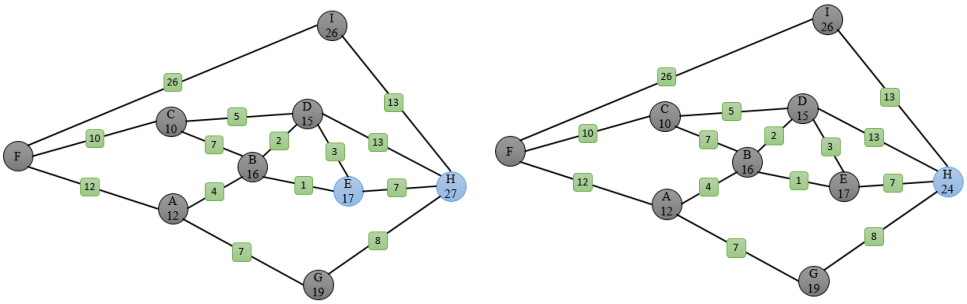
Oppgave 4: Korteste vei i en graf

a) Dijkstras algoritme starter med en prioritetskø av noder. Alle noder får prioritet uendelig unntatt startnoden. Besøk så hver kant fra startnoden og oppdater prioriteten til de besøkte nodene til å være korteste vei til noden så langt. Fortsett så å ta ut noden med høyest prioritet (lavest avstand) fra køen og gjenta. Gjenta prosessen til du har nådd målnoden.

b) Avstanden er 24:

i.





ii. F – A – B – E – H

Oppgave 5: Lenket liste

a) Utskriften er
A, B, C, D, E, F, G, H, I
B, D, E, G, H, I
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 6 is out of bounds of linked list

Først skrives listen ut, så fjernes node på plass 2 (C – listen inneholder nå ABDEFGHI), så på plass 4 (F – listen inneholder nå ABDEGHI), så på plass 0 (A – listen inneholder nå BDEGHI) og vi skriver ut. Så fjerner vi element på plass 6, og får out of bounds feil siden den ikke finnes i listen og programmet avsluttes.

b)

```
/**
 * Funksjon som fjerner noden q fra en dobbelt lenket liste
 * @param q
 */
void remove(Node q) {
    Node p = q.prev;
    Node r = q.next;

    p.next = r;
    r.prev = p;
    q = null;
}
```

c)

```
/**
 * Funksjon som fjerner noden på plass index
 * @param index Indexen å fjerne
 */
void remove(int index) {
```

```
    Node q = this.start;

    if(index < 0 || index >= size) {
        throw new ArrayIndexOutOfBoundsException("Index " + index + "
is out of bounds of linked list");
    }

    for (int i=0; i<index; ++i) {
        q = q.next;
    }

    remove(q);

    if (index == 0) {
        this.start = this.start.next;
    }

    this.size -= 1;
}
```