

i Innledning

IN2010/INF2220 - Algoritmer og datastrukturer

Tirsdag 27. november 2018

Kl. 14.30 - 18.30 (4 timer)

Oppgavesettet består av totalt 15 oppgaver.

Poengsum er angitt for hver oppgave.

Maksimum poengsum for hele oppgavesettet er 100 poeng.

Tillatte hjelpemidler: Ingen.

I dette oppgavesettet skal du svare med digital håndtegning på oppgave 8 "Huffman" og 9 "Topologisk sortering". Du bruker skisseark du får utdelt. Det er anledning til å bruke flere ark per oppgave. Se instruksjon for utfylling av skisseark på pult.

Det er IKKE anledning til å bruke digital håndtegning på andre oppgaver enn oppgave 8 og 9. Det blir IKKE gitt ekstratid for å fylle ut informasjonsboksene på skisseark (engangskoder, kand.nr. o.l.).

Generelle råd:

Husk å begrunne svar der det er bedt om det.

Vekten till en oppgave indikerer vanskelighetsgraden og tidsbruken du bør bruke på oppgaven, ut i fra våre estimat. Dette kan være greit å benytte for å disponere tiden best mulig, dvs. ikke bruk mye tid på en oppgave med liten prosentatsats (gå videre).

Det kan være lurt å starte med å lese gjennom hele oppgavesettet. Både for å få en oversikt og for å notere eventuelle spørsmål til faglærer.

Lykke til!

1 O-notasjon (4 poeng)

For hver av de fire kodebitene under, hva er «worst-case»-tidskompleksiteten, som funksjon av inputparameteren n ? Angi svaret ved O-notasjon.

```
// kodebit (i)
for (int i = 1; i <= 8; i++) {
    // Gjør noe
}
```

```
// kodebit (ii)
for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= n; j++) {
        sum++;
    }
}
```

```
// kodebit (iii)
for (int i = 1; i <= 2*n; i++) {
    for (int j = 1; j <= i; j++) {
        teller++;
    }
}
```

```
// kodebit (iv)
for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= i; j++) {
        for (int k = 1; k <= i+j; k++) {
            resultat += i*j;
        }
    }
}
```

```
}  
}  
}
```

Skriv ditt svar her...

Format - | **B** *I* U x_2 x^2 | I_x | | | | Ω | | Σ |

Words: 0

Maks poeng: 4

2 Binære søketrær (8 poeng)

I et binært søketre har vi noder med heltallsverdier.

Vi angir **en nodes plassering** i treet med en tekststreng slik:

RLLRLR 761

Dette betyr at det i treet finnes en node med verdi 761, og for å finne noden må vi fra rota først følge kanten mot høyre (R), så mot venstre (L), så venstre, høyre, venstre, høyre.

Sett følgende verdier inn i et tomt binært søketre. (For å svare på spørsmålene må du lage en kladd.)
Rekkefølgen kan ikke endres:

25, 36, 40, 30, 38, 48, 50, 45, 28

Når du har tegnet treet etter innsettingen, skal du svare på disse 16 spørsmålene (Du skal ikke tegne treet i svaret ditt.):

- a) Hvor mange kanter er det fra rota til dypeste bladnode/løvnnode?
- b) Hvor mange søskenpar finnes i treet?
- c) Skriv i stigende orden verdiene til barnet/a til noden med verdi 30.
- d) Er nodene med verdi 28 og 38 søsken?
- e) Hvor mange enebarn (noder uten søsken) finnes i treet?
- f) Hvor mange noder er løvnoder/bladnoder?
- g) Skriv i stigende orden verdien(e) til nodene som har nøyaktig ett barn.

- h) Finnes det en eller flere noder uten forelder i treet?
- i) Hva blir tekststrengen som definerer plasseringen til noden med verdi 38 i treet?
- j) Hvor mange noder har strukturelt forskjellige subtrær?
- k) Hvilken verdi skal stå bak stien RRRL ?
- l) Finnes det en node med stien L i treet?
- m) Hvilken bladnode ble satt inn først?
- n) Skriv i stigende orden nodene med dybde 2.
- o) Hvor mange noder til må vi sette inn for å få et perfekt balansert tre?
- p) Kan treet fargelegges som et rød-svart tre uten omstruktureringer?

Skriv ditt svar her...

Format | **B** | *I* | U | x_2 | x^2 | I_x | | | | | | | Ω | | | Σ | |

Words: 0

Maks poeng: 8

3 Trær: writePathToNode (6 poeng)

Fra forrige oppgave:

I et binært søketre har vi noder med heltallsverdier. Vi definerer en nodes plassering i treet med en tekststreng slik:

RLLRLR 761

Dette betyr at det i treet finnes en node med verdi 761, og for å finne noden må vi fra rota først følge kanten mot høyre (R), så mot venstre (L), så venstre, høyre, venstre, høyre.

Nytt:

Gitt følgende nodeklasse:

```
class Node {
    Node l, r;    // left, right
    int v;       // value
}
```

}

Skriv en Java-metode **writePathToNode** som gitt en verdi leter i treet og skriver ut stien og verdien hvis det finnes en node med denne verdien i treet. Det er ikke tillatt å endre nodeklassen, men du kan lage hjelpemetoder hvis du mener det er hensiktsmessig. Eksempel på utskrift:

RLLRLR 761

eller

Verdien 761 finnes ikke i treet.

Skriv ditt svar her...

1	
---	--

Maks poeng: 6

4 Trær: removeLessThan (6 poeng)

Fra forrige oppgave:

I et binært søketre har vi noder med heltallsverdier. Vi definerer en nodes plassering i treet med en tekststreng slik:

RLLRLR 761

Dette betyr at det i treet finnes en node med verdi 761, og for å finne noden må vi fra rota først følge kanten mot høyre (R), så mot venstre (L), så venstre, høyre, venstre, høyre.

Gitt følgende nodeklasse:

```
class Node {
    Node l, r;    // left, right
    int v;       // value
}
```

Nytt:

Skriv en Java-metode **removeLessThan (Node n, int value)** slik at hvis **Node root** er et binært søketre, vil tilordningen

```
root = removeLessThan(root, value);
```

fjerne alle noder med verdier mindre enn **value** fra treet. Du kan lage hjelpemetoder hvis du mener det er hensiktsmessig.

1	
---	--

Maks poeng: 6

5 Hashing (5 poeng)

Vi skal legge disse tallverdiene inn i en hashtabell med lengde 11:

{42, 78, 57, 18, 12, 74, 99, 20, 33, 61, 19, 91}

I denne oppgaven skal du bruke hashfunksjonen

$$h(k) = k \bmod 11$$

a) Angi om du vil benytte åpen eller lukket hashing og evt. hvilke tilleggsfunksjoner du bruker utenom h . Begrunn svaret.

b) Deretter viser du hvordan hashtabellen blir.

Svaret skal være 11 linjer; én indeks pr. linje på denne måten:

indeks: verdi [, verdi]

indeks: verdi [, verdi]

indeks: verdi [, verdi]

Verdiene skrives med komma i mellom hvis flere verdier på én indeks. La verdien som kommer først på indeksen stå lengst til venstre. La det være tomt etter kolonet hvis det ikke er noen verdi på indeksen.

Skriv ditt svar her...

1	
---	--

Maks poeng: 5

6 Mapping-metoder (6 poeng)Skriv kode for metoden **put** som setter et element **e** (av type **Element**) inn i tabellen **hashTable**:**Object [] hashTable = new Object[n];**Nøkkelen til **e** finnes i **e.k**, som er en **int** variabel.**put** skal bruke *lukket hashing med lineær prøving*. Hashfunksjonen er $h(x) = x \bmod n$. Metoden skal returnere true/false avhengig av om elementet ble satt inn eller ikke.**Skriv ditt svar her...**

1	
---	--

7 Heap (4 poeng)

Gitt følgende array (index 0 er tom). Representerer dette arrayet en heap? Begrunn svaret.

[, 9, 5, 7, 8, 25, 13, 9, 6]

Skriv ditt svar her...

Maks poeng: 4

8 Huffman (4 poeng)

Basert på følgende frekvens av tegn, tegn det resulterende huffmantreet og gi tegnenes huffmankoder.

A(106), B(200), C(500), D(701), E(305)

I denne oppgaven skal du svare med digital håndtegning. Bruk eget skisseark (utdelt). Se instruksjon for utfylling av skisseark på pult.

Maks poeng: 4

9 Topologisk sortering (4 poeng)

Tegn de følgende grafene. Bestem for hver av dem om det finnes en topologisk sortering. Hvis ja, skriv ned ett eksempel, hvis nei, forklar hvorfor.

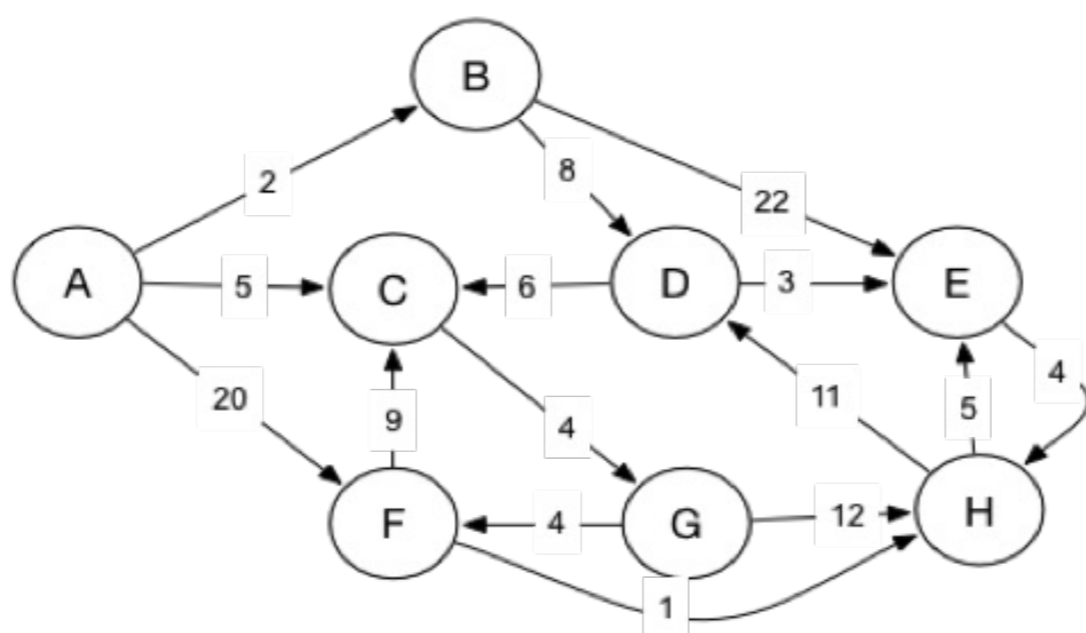
a) { (E, A), (A, D), (D, C), (C, B), (E, C) }

b) { (A, D), (D, B), (B, C), (C, A), (C, E) }

I denne oppgaven skal du svare med digital håndtegning. Bruk eget skisseark (utdelt). Se instruksjon for utfylling av skisseark på pult.

Maks poeng: 4

10 Dijkstra (10 poeng)



Oppgave a:

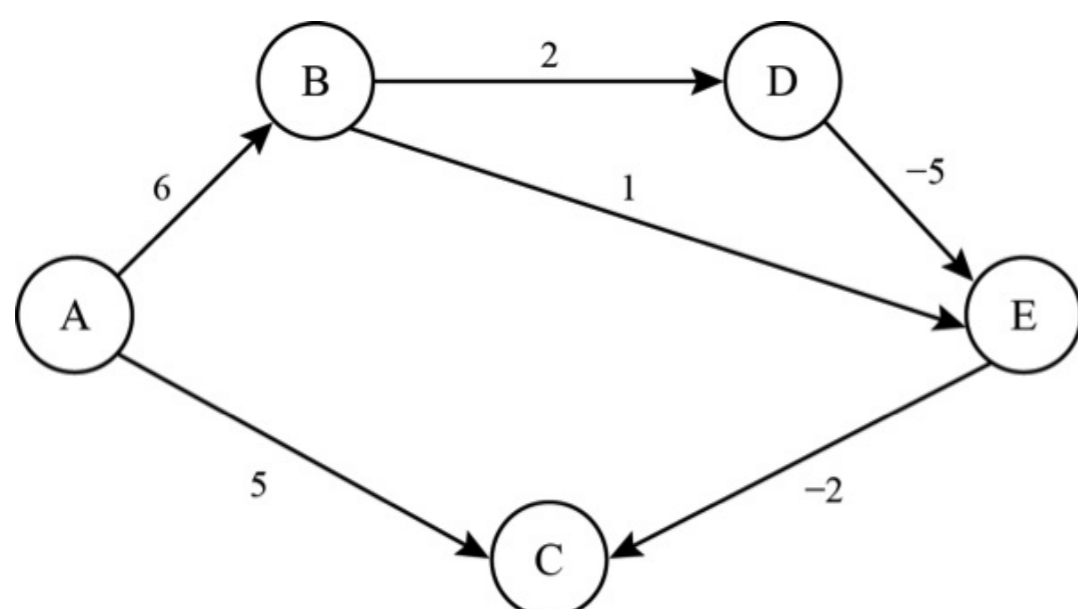
Bruk Dijkstras algoritme til å finne korteste vei fra node A til andre noder. Vis trinnene i tabellen under. Dette gjør du ved at du skriver inn nye verdier fra venstre mot høyre i hver celle i tabellen og setter siste verdien i **fet** skrifttype. Hvis du i løpet av algoritmen har flere ukjente noder med samme avstand, bruk alfabetisk rekkefølge til å bestemme hvilken node som blir valgt først. Du skal også oppgi noder i den rekkefølgen algoritmen markerer dem som kjent.

List nodene i den rekkefølge de ble gjort kjent: _____

Node	Kjent	Avstand	Sti
A			
B			
C			
D			
E			
F			
G			
H			











Lag en tabell med 9 rader og 4 kolonner lik den ovenfor. La bredden på kolonnene være 500. (Tabellverktøyet er knappen til høyre for Ω -knappen.)

Oppgave b:



Denne grafen har ingen negative sykler. Kan Dijkstras algoritme finne korteste vei for denne grafen? Begrunn.

Skriv ditt svar her...

Format - | **B** *I* U x_2 x^2 | \int_x |   |   |   | Ω   | Σ | ABC  | 

Words: 0

Maks poeng: 10

11 Urettet graf (10 poeng)

Implementer en Java-metode *isTree* som returnerer **true** hvis en gitt urettet graf er et tre og **false** ellers. Du kan lage hjelpemetoder hvis du mener det er hensiktsmessig. Anta at nodene er representert ved et heltall fra 0 til $|V|-1$ og at grafen er av klassen *Graph* som angitt i vedlegget.

Skriv ditt svar her...

1

Maks poeng: 10

12 Emner (14 poeng)

Anta at du skal ta N emner på IFI der hvert emne har 10 studiepoeng og alle emnene går hvert semester. Anta at du har en progresjon der du tar maks 30 studiepoeng hvert semester. Noen emner har forkunnskapskrav. Hvis emne B krever som forkunnskap emne A, må du ha tatt A i et tidligere semester før du kan ta emne B. Anta at du representerer emner og forkunnskapskravene som en graf. For å gjøre det enkelt representerer vi de N emnene med heltall (integer) fra 0 til $(N-1)$.

Implementer en Java-metode *numberOfSemesters* som returnerer antall semester du minst må ha for å gjennomføre alle emnene. Du kan lage hjelpemetoder hvis du mener det er hensiktsmessig. Anta at grafen er gitt ved klassen Graph i vedlegget (tilsvarende struktur som i forrige oppgave).

Skriv ditt svar her...

1		
---	--	--

Maks poeng: 14

13 Shell-sortering (6 poeng)

Shell-sortering er en sorteringsalgoritme som bygger på innstikksortering. I den varianten av shell-sortering som vi skal se på her brukes en inkrement-sekvens som for $n = 10$ vil være $h_3=7$, $h_2=3$, $h_1=1$. (Generelt: alle tall på formen 2^k-1 fra n ned til 1). I hver iterasjon av shell-sortering gjøres innstikksortering på elementene med avstand h_k i arrayen.

Dvs at for $n = 10$ vil det først gjøres innstikksortering på de tre del-arrayene

$a[1], a[8]$ og $a[2], a[9]$ og $a[3], a[10]$

(dvs elementer med 7 i avstand),

deretter gjøres innstikkstortering på de tre del-arrayene

$a[1], a[4], a[7], a[10]$ og $a[2], a[5], a[8]$ og $a[3], a[6], a[9]$

(dvs elementer med 3 i avstand),

før det til slutt gjøres innstikksortering på hele arrayen $a[1], \dots, a[10]$

(dvs elementer med 1 i avstand).











Oppgave a:

Er Shell-sortering en stabil sorteringsalgoritme? Begrunn svaret. (Riktig svar uten begrunnelse gir ikke poeng.)

Oppgave b:

Shell-sortering som beskrevet over har en kjøretid på $O(n^{3/2})$? Hvordan er dette sammenlignet med innstikksortering?

Skriv ditt svar her...

Format - | **B** *I* U x_2 x^2 | \int_x |   |   |   | Ω   | Σ | ABC  | 

Words: 0

Maks poeng: 6

14 Sortering (7 poeng)

Gitt følgende klasse:

```
class Elev implements Comparable<Elev> {  
    int karakter; // heltall fra 1 (dårligst) til 6 (best)  
  
    int compareTo(Elev o) {  
        return this.karakter - o.karakter;  
    }  
}
```

Beskriv med tekst og/eller kode hvordan du vil tilpasse en av algoritmene fra pensum til raskest mulig å sortere en array med n elever (`Elev[] elever = new Elev[n]`) fra best (karakter 6) til dårligst (karakter 1). Begrunn også valg av algoritme.

1	
---	--

Maks poeng: 7

15 NP-kompletthet (6 poeng)

a) Anta at du vet at et problem A er NP-komplett og at et problem B er i NP, og at du skal vise at også problem B er NP-komplett. Må du da vise at A kan reduseres til B eller at B kan reduseres til A? Begrunn svaret.

b) I et selskap skal det plasseres N gjester ved et rundt bord. Vi vet hvilke par av gjester som trives sammen, og hvilke som ikke gjør det. Dette er angitt ved hjelp av en boolsk $N \times N$ -matrise (som er symmetrisk). Vi ønsker å finne ut om det er mulig å lage en bordplassering slik at alle som sitter ved siden av hverandre trives sammen.

Vi tenker oss av både verdien N og trives-matrisen, kan variere fritt. Du kan anta som kjent at dette problemet er i NP, og din oppgave er å vise at det også er NP-komplett. Du skal bruke ett av de NP-komplette problemene fra pensum (VERTEX-COVER, SUBSET-SUM, HAMILTONIAN-CYCLE og TSP) som utgangspunkt for beviset.

Skriv ditt svar her...

Format | **B** | *I* | U | x_2 | x^2 | \int_x | | | | | | | Ω | | | Σ | ABC |

Words: 0

Maks poeng: 6

Question 11
Attached



```

class Graph {

    private int V;    // No. of vertices
    // an array of linked lists
    private LinkedList<Integer> adj[]; //Adjacency List (nabo-liste)

    // Constructor
    Graph(int v){
        V = v;
        adj = new LinkedList[v];
        for (int i=0; i<v; ++i)
            adj[i] = new LinkedList();
    }

    // Function to add an edge into the graph
    void addEdge(int v,int w){
        adj[v].add(w);
        adj[w].add(v);
    }

    // Returns true if the graph is a tree, else false.
    boolean isTree()
    {... }
}

```

Question 12
Attached




```
class Graph {  
  
    private int V; // No. of vertices  
    // an array of linked lists  
    private LinkedList<Integer> adj[]; //Adjacency List (nabo-liste)  
  
    // Constructor  
    Graph(int v){  
        V = v;  
        adj = new LinkedList[v];  
        for (int i=0; i<v; ++i)  
            adj[i] = new LinkedList();  
    }  
  
    // Function to add an edge into the graph  
    void addEdge(int v,int w){  
        adj[v].add(w);  
    }  
  
    // Returns number of semesters  
    int numberOfSemesters()  
    {... }  
  
    ...  
}
```