Introduksjon til DAT200 eksamen

EMNE: DAT200 Algoritmer og Datastrukturer

DATO OG KLOKKESLETT: 9. desember 2019 klokka 09.00 - 13.00
TID FOR EKSAMEN: 4 timer
TILLATTE HJELPEMIDDEL: Ingen trykte og håndskrevne hjelpemidler, standard enkel kalkulator
EMNEANSVARLIG: Erlend Tøssebro
TELEFONNUMMER: 48107119

Merknad: Du må lese instruksjonene før du starter med å løse eksamensoppgavene!

Alle programeksempler er oppgitt i både Java og Python. Dette skyldes at DAT200 har endret programmeringsspråk fra Java til Python i år, men dette eksamenssettet er også tredje forsøk for de som hadde DAT200 i fjor med Java. For alle oppgaver som krever programmering så er det lov å programmere i både Java og Python.

For oppgavene som krever programmering: Det viktigste er at du viser at du forstår hvordan en løsning kan programmeres i enten Java eller Python. Enkle skrivefeil i navn på metoder vil ikke trekke ned. Å bruke et annet navn på en metode så lenge det går klart fram hva du prøver å gjøre vil heller ikke trekke ned.

Når du programmerer kan du bruke en syntaks fra C++ for å indikere at en metode tilhører en klasse. Dette betyr at i stedet for å skrive metoden inni klassen så skriver du klassenavn::metodenavn.

Du trenger ikke å inkludere import-setninger i programmene dine med mindre oppgaven eksplisitt spør om det.

1 Analyse av algoritmer

Gitt følgende funksjon, hva er kjøretida dens i O-notasjon? Problemstørrelsen n er lengden til lista. (4%)

Python: Anta at parameteren "liste" er en standard Python liste (array-liste).

```
def metode1(liste):
    tall1 = liste[0]
    tall2 = liste[-1]
    tall3 = liste[len(liste)//2]
    return (tall1 + tall2 + tall3)/3.0
```

Java:

```
public static double metode1(double[] liste) {
    double tall1 = liste[0];
    double tall2 = liste[liste.length-1];
    double tall3 = liste[(liste.length)/2];
    return (tall1 + tall2 + tall3)/3.0;
}
```

Velg ett alternativ

- O(1)
- O(log(n))
- O(n)
- O(n*log(n))
- O(n^2)
- O(n^3)
- O(2^n)

Maks poeng: 4

2 Analyse av algoritme 2

Hva er kjøretida til funksjonen "navnebroedre", i Python og Java? Student klassen er oppgitt under selve oppgaven for de som er interessert. Problemstørrelsen er antall studenter i lista. Anta at metodekallet get_navn_formell() kjører i O(1) tid. (4%)

Python: Anta at lista er ei standard Python liste (Array-liste) som inneholder objekter av klassen Student.

```
def navnebroedre(studentliste):
    resultat = []
    for i in range(len(studentliste)):
        har_navnebror = False
        for j in range(i+1, len(studentliste)):
            if studentliste[i].get_navn_formell() == studentliste[j].get_navn_formell():
                resultat.append(studentliste[j])
                har_navnebror = True
        if har_navnebror:
            resultat.append(studentliste[i])
    return resultat
```

Java:

```
public static ArrayList<Student> navnebroedre(ArrayList<Student> studentliste) {
    ArrayList<Student> resultat = new ArrayList<>();
    boolean harNavnebror;
    for (int i=0;i<studentliste.size();i++) {
        harNavnebror = false;
```

```
            for (int j=i+1;j<studentliste.size();j++) {
                if (studentliste.get(i).getNavnFormelt().equals(studentliste.get(j).getNavnFormelt())) {
                    resultat.add(studentliste.get(j));
                    harNavnebror = true;
                }
            }
            if (harNavnebror) {
                resultat.add(studentliste.get(i));
            }
        }
        return resultat;
    }
```

Velg ett alternativ

- O(1)
- O(log(n))
- O(n)
- O(n*log(n))
- O(n^2)
- O(n^3)
- O(2^n)

Klassen Student, Python:

```
class Student:
    def __init__(self, studentnummer, etternavn, fornavn, fodselsaar, studieretning, aarskurs):
        self.__studentnummer = studentnummer
        self.__etternavn = etternavn
        self.__fornavn = fornavn
        self.__fodselsaar = fodselsaar
        self.__studieretning = studieretning
        self.__aarskurs = aarskurs
    def get_studentnummer(self):
        return self.__studentnummer
    def get_navn_formell(self):
        return self.__etternavn + ", " + self.__fornavn
    def get_etternavn(self):
        return self.__etternavn
    def get_fornavn(self):
        return self.__fornavn
    def get_fodselsaar(self):
        return self.__fodselsaar
    def get_studieretning(self):
        return self.__studieretning
    def get_aarskurs(self):
        return self.__aarskurs
    def __str__(self):
        return f"Student {self.__studentnummer}: {self.__etternavn}, {self.__fornavn}, går i {self.__aarskurs}. årskurs
```

Klassen Student, Java

```
public class Student {
    private String fornavn;
    private String etternavn;
    private int studentnummer;
    private int fodselsaar;
    private String studieprogram;
    private int aarskurs;

    public static int neste_studentnummer = 1;

    public Student(String fornavn, String etternavn, int fodselsaar) {
```

```
        this.fornavn = fornavn;
        this.etternavn = etternavn;
        this.fodselsaar = fodselsaar;
        studieprogram = "Ikke definert";
        aarskurs = 1;
        this.studentnummer = neste_studentnummer;
        neste_studentnummer++;
    }

    public int getStudentnummer() { return studentnummer;}

    public String getFornavn() { return fornavn;}

    public String getEtternavn() { return etternavn;}

    public String getNavnFormelt() {
        return etternavn + ", " + fornavn;
    }

    public String getStudieprogram() { return studieprogram;}

    public Student setStudieprogram(String studieprogram) {
        this.studieprogram = studieprogram;
        return this;
    }

    public int getAarskurs() { return aarskurs; }

    public Student setAarskurs(int aarskurs) {
        this.aarskurs = aarskurs;
        return this;
    }

    public int getFodselsaar() { return fodselsaar; }

    @Override public String toString() {
        String resultat = "Student " + getStudentnummer() + ": "+ getNavnFormelt() + " " +
                " studerer " + studieprogram + " i " + aarskurs + " aarskurs \n";
        return resultat;
    }
}
```

Maks poeng: 4

3 Analyse av rekursiv algoritme

Gitt følgende rekursive funksjon, finn kjøretida dens i O-notasjon. (6%)

Python: Anta at parameteren liste er ei standard Python liste (array-liste).

```
def maksimal_selsekvens_sum_rekursiv(liste, start=0, slutt=-1):
    if slutt == -1:
        slutt = len(liste) - 1
    if start == slutt:
        if liste[start] > 0:
            return liste[start]
        else:
            return 0
    senter = (start + slutt)//2
    venstre_sum = maksimal_selsekvens_sum_rekursiv(liste, start, senter)
    hoyre_sum = maksimal_selsekvens_sum_rekursiv(liste, senter+1, slutt)
    sum_mot_venstre = 0
    maks_sum_mot_venstre = 0
```

```
    for i in range(senter, start, -1):
        sum_mot_venstre += liste[i]
        if sum_mot_venstre > maks_sum_mot_venstre:
            maks_sum_mot_venstre = sum_mot_venstre
    sum_mot_hoyre = 0
    maks_sum_mot_hoyre = 0
    for i in range(senter+1, slutt):
        sum_mot_hoyre += liste[i]
        if sum_mot_hoyre > maks_sum_mot_hoyre:
            maks_sum_mot_hoyre = sum_mot_hoyre
    return max([venstre_sum, hoyre_sum, maks_sum_mot_venstre+maks_sum_mot_hoyre])
```

Java:

```
public static int maksimalDelsekvensSumRekursiv(int[] liste) {
   return maksimalDelsekvensSumRekursiv(liste, 0, liste.length-1);
}

public static int maksimalDelsekvensSumRekursiv(int[] liste, int start, int slutt) {
if (start == slutt) {
    if (liste[start] > 0) {
        return liste[start];
    } else return 0;
}
int senter = (start+slutt)/2;
int venstreSum = maksimalDelsekvensSumRekursiv(liste, start, senter);
int hoyreSum = maksimalDelsekvensSumRekursiv(liste, senter+1, slutt);

int sumMotVenstre = 0;
int maksSumMotVenstre = 0;
for (int i=senter;i>=0;i--) {
    sumMotVenstre += liste[i];
    if (sumMotVenstre > maksSumMotVenstre) maksSumMotVenstre = sumMotVenstre;
}

    int sumMotHoyre = 0;
    int maksSumMotHoyre = 0;
    for (int i=senter+1;i<=slutt;i++) {
        sumMotHoyre += liste[i];
        if (sumMotHoyre > maksSumMotHoyre) maksSumMotHoyre = sumMotHoyre;
    }

    int maks = Math.max(hoyreSum, venstreSum);
    maks = Math.max(maksSumMotHoyre + maksSumMotVenstre, maks);
    return maks;
  }
```

Velg ett alternativ

- O(1)
- O(log(n))
- O(n)
- O(n*log(n))
- O(n^2)
- O(n^3)
- O(2^n)

Maks poeng: 6

4 Dynamisk programmering

Beskriv programmeringsteknikken "dynamisk programmering", gjerne gjennom et eksempel. (8%)

Skriv ditt svar her...

Format

Words: 0

Maks poeng: 8

5 Stabler

Hva er datastrukturen stabel (engelsk stack)? Hva er de vanlige operasjonene som man har på stabler? Hvordan er stabler vanligvis implementert? (8%)

Skriv ditt svar her...

Format

Words: 0

Maks poeng: 8

6 Stabile sorteringsalgoritmer

Hva betyr det at en sorteringsalgoritme er stabil? (4%)

Skriv ditt svar her...

Format

Words: 0

Maks poeng: 4

7 Quicksort

Gitt følgende liste: [29, 47, 93, 46, 37, 23, 1, 52, 89]

Bruk median-of-three pivot valg for å velge pivot. Pivot blir da: [ ] Husk at indeksen i ei liste starter på 0 (gjelder både Python og Java).

Gjør første splitt. Første liste blir da:

[, , , ,]

Andre liste blir da:

[, , , ,]

I alle fall én av de oppgitte listene har for mange elementer. Oppgi tallet 0 i de resterende plassene. Oppgi elementene i samme rekkefølge som de er i den opprinnelige lista. (11% total, 1% pr. riktig utfylt boks)

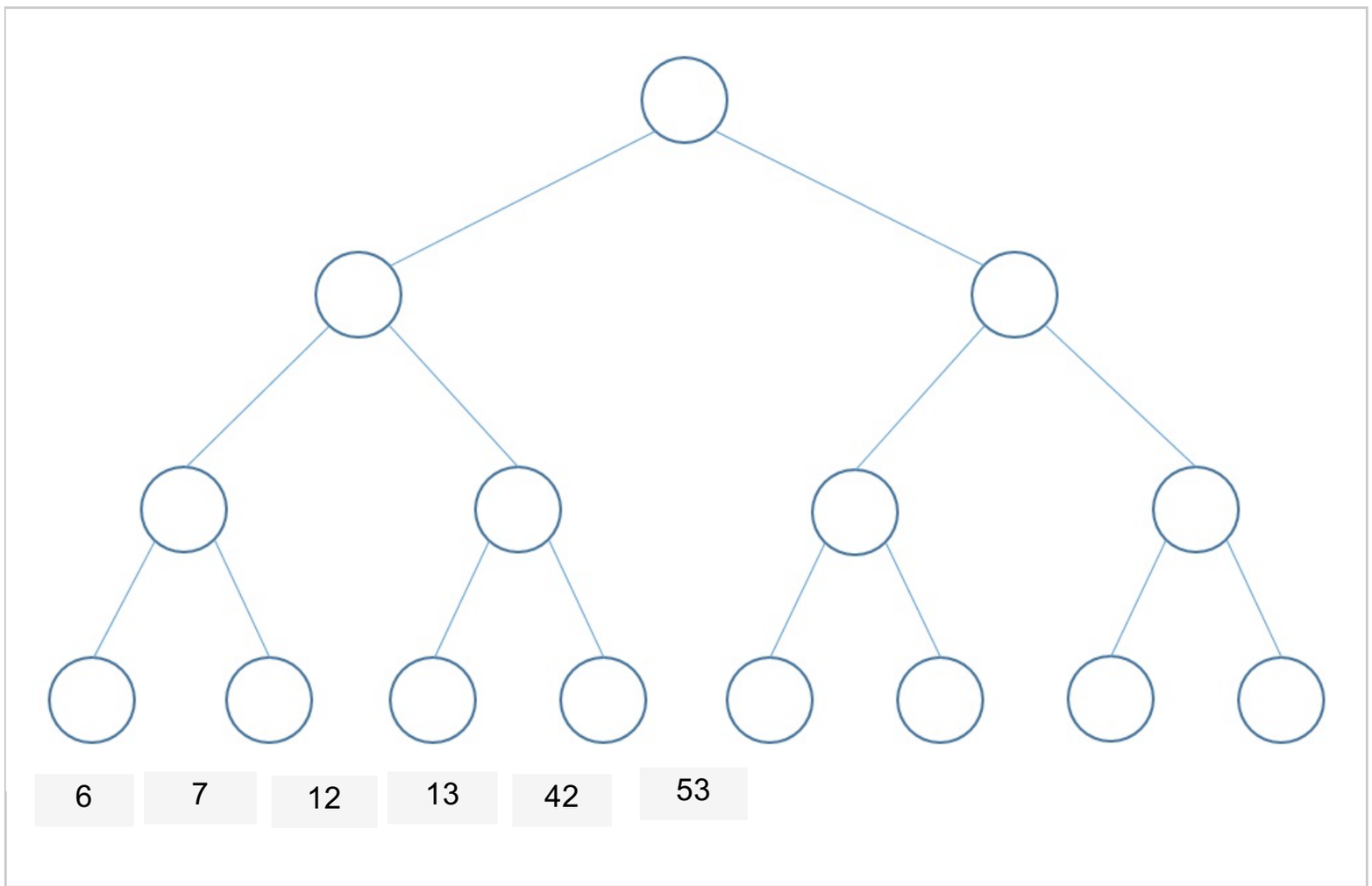
Maks poeng: 11

8

AVL trær

Sett inn elementene 13, 6, 42, 53, 7, 12 i et tomt AVL tre i den oppgitte rekkefølgen. Vis i figuren under hvordan det endelige AVL treet blir. Merk at treet som er tegnet opp bevisst har for mange noder. Noder som ikke er i AVL treet skal du ikke legge noen tall inn i. (9%)

Velg ett alternativ



Maks poeng: 9

9 Kjøretider for operasjoner på søkestrukturer

Hva er kjøretida til følgende operasjoner på følgende søkestrukturer? Anta worst-case med mindre noe annet er oppgitt for det spesifikke spørsmålet. (14% totalt, 2% pr. riktig svar)

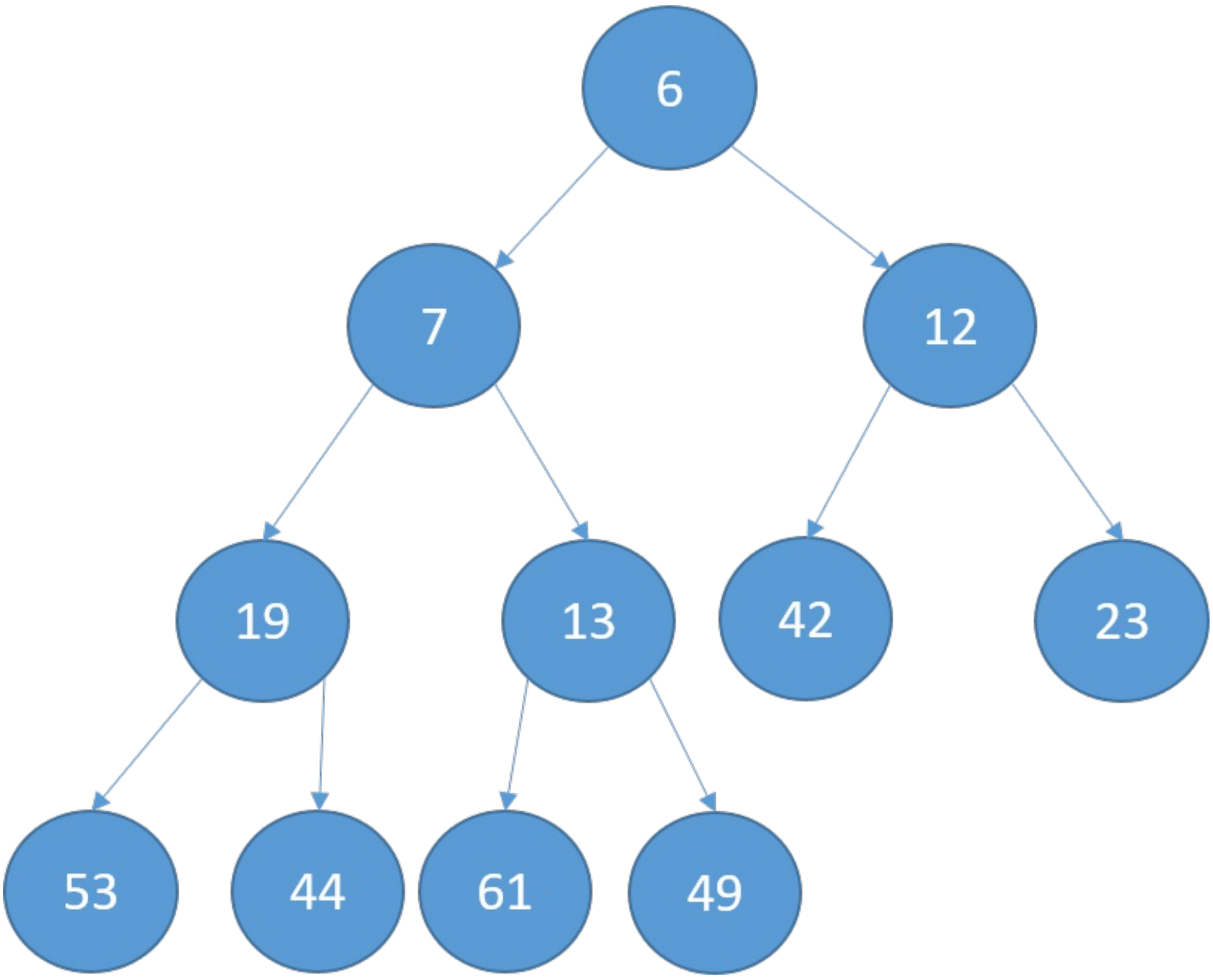
Velg den som passer for hvert element

	O(1)	O(log(n))	O(n)	O(n*log(n))	O(n^2)	O(n^3)	O(2^n)	Denne datastrukturen støtter ikke denne operasjonen
Innsetting i en hashtabell, average case	○	○	○	○	○	○	○	○
Innsetting i et AVL tre	○	○	○	○	○	○	○	○
Å finn et element i en hashtabell, average case	○	○	○	○	○	○	○	○
Å finne et element i en hashtabell, worst case	○	○	○	○	○	○	○	○
Å finne et element i et AVL tre	○	○	○	○	○	○	○	○
Å finne det minste elementet i en hashtabell	○	○	○	○	○	○	○	○
Å finne det minste elementet i et AVL tre	○	○	○	○	○	○	○	○

Maks poeng: 14

10 Haug

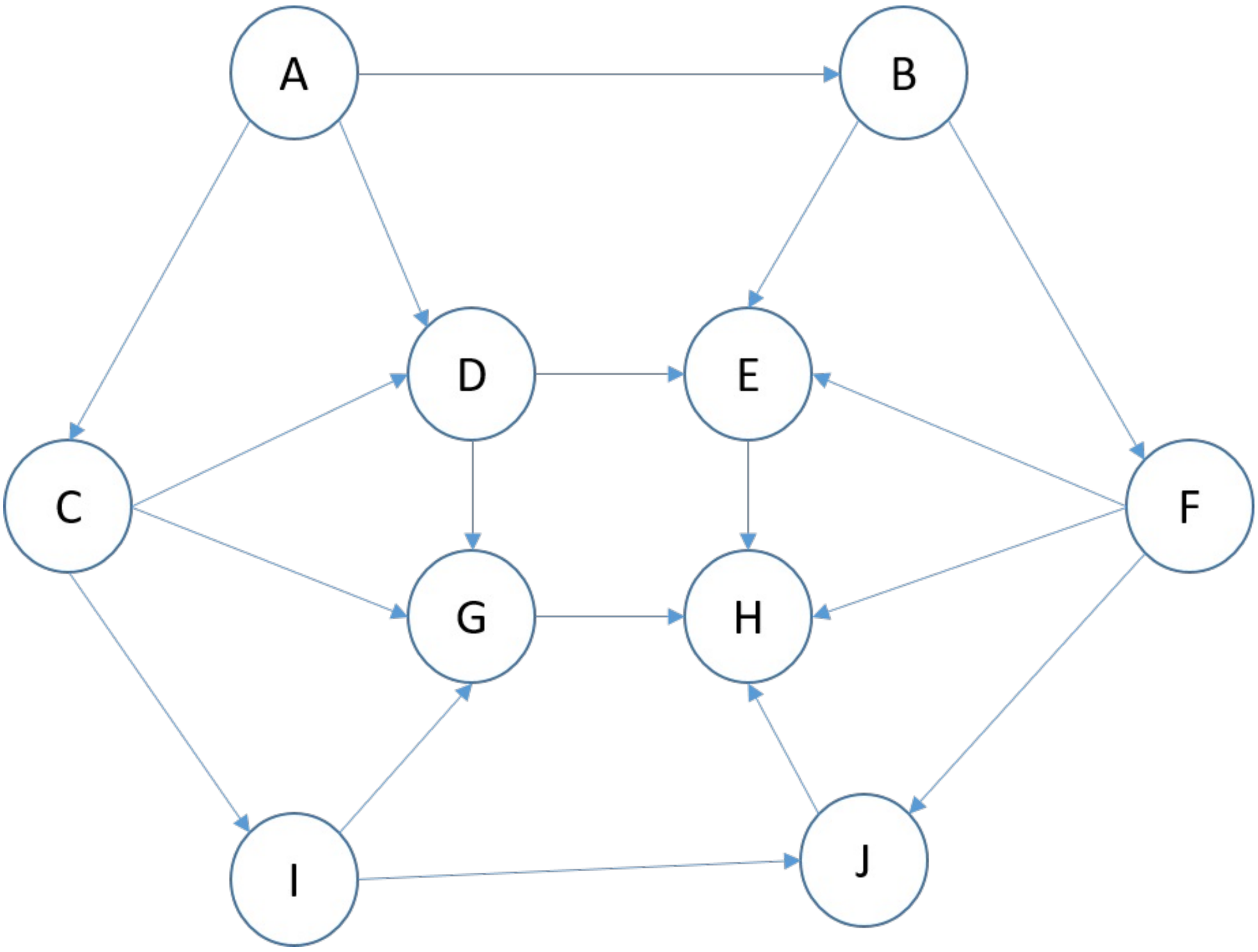
Gitt følgende haug, vis algoritmen for å hente ut (og slette) minste element fra haugen. Vis hvert steg på papir helt til det igjen er en lovlig haug. (10%)



Maks poeng: 10

11 Identifiser datastrukturen

Hvilken datastruktur er dette? Vær så spesifikk som mulig. Svar som er korrekte men for lite spesifikke vil få redusert poengscore. (4%)



Skriv ditt svar her...

Format

Words: 0

Maks poeng: 4

12 Tegn en graf

Gitt følgende nabomatrise, tegn grafen med noder og kanter. Tegn løsningen din på et ScanTron-ark. (6%)

	A	B	C	D	E
A	-	2	1	4	-
B	2	-	3	-	2
C	-	3	-	-	3
D	2	-	2	-	-
E	-	2	3	-	-

Maks poeng: 6

13 Korteste vei

Gitt en graf over veinettverket i Norge hvor noen av nodene er markert "tunnelåpning" og noen av nodene er markert "brannstasjon". Slike noder inneholder også en ID for hvilken tunnel eller brannstasjon de representerer:

Skriv en algoritme, i pseudokode, Python eller Java kode, som for hver tunnel finner de fem nærmeste brannstasjonene og lagrer navn og avstand til de fem brannstasjonene i ei liste for hver tunnel (separate lister for hver tunnel) og sortert i stigende rekkefølge på avstand slik at nærmeste brannstasjon kommer først. Husk at en tunnel typisk har to åpninger og kan ha flere. Alle tunnelåpningene til en enkelt tunnel har lagret samme ID. Du kan anta at du har ei separat liste over tunneler inkludert node-indekser til tunnelåpningene dens.

Denne oppgaven krever at du modifiserer en av algoritmene fra forelesningstimene og kjører den modifiserte algoritmen flere ganger. Beskriv modifikasjonen du vil gjøre og hvordan du vil bruke den modifiserte algoritmen flere ganger for å løse dette problemet. Korteste vei algoritmene fra timene er oppgitt under i Python kode og Java kode.

Du kan anta at grafen har en metode get_markering(node_index) som returnerer hvilken av disse markeringene en node har eller om den har ingen av dem, og en get_objekt_ID(node_index) som returnerer ID-en til tunnelen eller brannstasjonen. node_index er den samme som brukes av de andre metodene som get_kostnad og get_neighbours. (12%)

Skriv ditt svar her...

```
1
```

Python kode for korteste vei algoritmer fra timene:

```
def bredde_forst_sok(graf, fra_index, til_index=None):
    graf.fjern_kostnader()
    nodekoe = UbegrensetKoe()
```

```
        graf.set_kostnad(fra_index, 0)
        nodekoe.enqueue(fra_index)
        while len(nodekoe) > 0:
            node_index = nodekoe.dequeue()
            node_kostnad = graf.get_kostnad(node_index)
            naboer = graf.get_neighbours(node_index)
            for nabo in naboer:
                if graf.get_kostnad(nabo) is None:
                    graf.set_kostnad(nabo, node_kostnad + 1)
                    graf.set_forrige(nabo, node_index)
                    if nabo == til_index:
                        return node_kostnad + 1
                    nodekoe.enqueue(nabo)
        return None

    def dijkstra(graf, fra_index, til_index):
        graf.fjern_kostnader()
        prioritetsko = Haug()
        graf.set_kostnad(fra_index, 0)
        prioritetsko.add(0, fra_index)
        while len(prioritetsko) > 0:
            node_index = prioritetsko.remove()
            node_kostnad = graf.get_kostnad(node_index)
            naboer = graf.get_neighbours(node_index)
            if node_index == til_index:
                return node_kostnad
            for nabo in naboer:
                if graf.get_kostnad(nabo) is None:
                    graf.set_kostnad(nabo, node_kostnad + graf.get_weight(node_index, nabo))
                    graf.set_forrige(nabo, node_index)
                    prioritetsko.add(graf.get_kostnad(nabo), nabo)
                elif graf.get_kostnad(nabo) > node_kostnad + graf.get_weight(node_index, nabo):
                    graf.set_kostnad(nabo, node_kostnad + graf.get_weight(node_index, nabo))
                    graf.set_forrige(nabo, node_index)
                    prioritetsko.ok_prioritet(graf.get_kostnad(nabo), nabo)
        return None

    def bellman_ford(graf, fra_index, til_index=None):
        koe_operasjoner_for_node = {}
        graf.fjern_kostnader()
        nodekoe = UbegrensetKoe()
        graf.set_kostnad(fra_index, 0)
        nodekoe.enqueue(fra_index)
        koe_operasjoner_for_node[fra_index] = 1
        while len(nodekoe) > 0:
            node_index = nodekoe.dequeue()
            koe_operasjoner_for_node[node_index] += 1
            if koe_operasjoner_for_node[node_index] > 2*graf.antall_noder():
                print("Grafen har en negativ sykel!")
                return None
            node_kostnad = graf.get_kostnad(node_index)
            naboer = graf.get_neighbours(node_index)
            for nabo in naboer:
                if graf.get_kostnad(nabo) is None:
                    graf.set_kostnad(nabo, node_kostnad + graf.get_weight(node_index, nabo))
                    graf.set_forrige(nabo, node_index)
                    koe_operasjoner_for_node[nabo] = 1
                    nodekoe.enqueue(nabo)
                elif graf.get_kostnad(nabo) > node_kostnad + graf.get_weight(node_index, nabo):
                    graf.set_kostnad(nabo, node_kostnad + graf.get_weight(node_index, nabo))
                    graf.set_forrige(nabo, node_index)
                    if koe_operasjoner_for_node[nabo]%2 == 0:
                        nodekoe.enqueue(nabo)
                        koe_operasjoner_for_node[nabo] += 1
        if til_index is not None:
            return graf.get_kostnad(til_index)
        else:
            return None
```

Java kode for korteste vei algoritmene fra timene

```
public class BreddeForstSok <E> {

    private KortesteVeiGraf<E> grafen;
    private Queue<Integer> nodekoe;

    public BreddeForstSok(KortesteVeiGraf<E> grafen) {
        this.grafen = grafen;
        nodekoe = new LinkedList<>();
    }
```

```
        public void breddeForstSok(int startnode, int maalnode) {
            grafen.reset();
            nodekoe.clear();

            grafen.setKostnad(startnode, 0);
            nodekoe.add(startnode);
            while (!nodekoe.isEmpty()) {
                int nvNode = nodekoe.remove();
                if (nvNode == maalnode) break;
                int nvKostnad = grafen.getKostnad(nvNode);
                List<Integer> naboer = grafen.getNeighbours(nvNode);
                for (int nabo: naboer) {
                    if (grafen.getKostnad(nabo) == Integer.MAX_VALUE) {
                        grafen.setKostnad(nabo, nvKostnad + 1);
                        grafen.setForrigePaaVeien(nabo, nvNode);
                        nodekoe.add(nabo);
                    }
                }
            }

            int nvNode = maalnode;
            while (nvNode != KortesteVeiGraf.HAR_IKKE_FORRIGE) {
                System.out.println(grafen.getElement(nvNode) + ", " + grafen.getKostnad(nvNode));
                nvNode = grafen.getForrigePaaVeien(nvNode);
            }
        }
    }


public class Dijkstra <E> {
    private KortesteVeiGraf<E> grafen;
    private Haug<Integer> nodehaug; // Haug av indekser inn i nodelista til grafen

    public Dijkstra(KortesteVeiGraf<E> grafen) {
        this.grafen = grafen;
    }

    public void kjorDijkstra(int startnode, int maalnode) {
        grafen.reset();
        nodehaug = new Haug<>();

        grafen.setKostnad(startnode, 0);
        nodehaug.add(startnode, 0);

        while (nodehaug.size() > 0) {
            int nvNode = nodehaug.removeMin();
            if (nvNode == maalnode) break;
            int nvKostnad = grafen.getKostnad(nvNode);
            List<Integer> naboer = grafen.getNeighbours(nvNode);
            for (int nabo: naboer) {
                int nabokostnad = grafen.getWeight(nvNode, nabo) + nvKostnad;
                if (nabokostnad < grafen.getKostnad(nabo)) {
                    if (grafen.getKostnad(nabo) == Integer.MAX_VALUE) {
                        grafen.setKostnad(nabo, nabokostnad);
                        grafen.setForrigePaaVeien(nabo, nvNode);
                        nodehaug.add(nabo, nabokostnad);
                    } else {
                        grafen.setKostnad(nabo, nabokostnad);
                        grafen.setForrigePaaVeien(nabo, nvNode);
```

```
                        nodehaug.decreaseKey(nabo, nabokostnad);
                    }
                }
            }
        }

        int nvNode = maalnode;
        while (nvNode != KortesteVeiGraf.HAR_IKKE_FORRIGE) {
            System.out.println(grafen.getElement(nvNode) + ", " + grafen.getKostnad(nvNode));
            nvNode = grafen.getForrigePaaVeien(nvNode);
        }
    }
}

public class BellmanFord <E> {
    private KortesteVeiGraf<E> grafen;
    private HashMap<Integer, Integer> noderIKoe; // Nodeindeks -> Antall ganger noden er lagt til og tatt ut
    private Queue<Integer> nodekoe;

    public BellmanFord(KortesteVeiGraf<E> grafen) {
        this.grafen = grafen;
    }

    public void kjorBellmanFord(int startnode, int sluttnode) {
        grafen.reset();
        noderIKoe = new HashMap<>();
        nodekoe = new LinkedList<>();

        grafen.setKostnad(startnode, 0);
        nodekoe.add(startnode);
        noderIKoe.put(startnode, 1);

        while (!nodekoe.isEmpty()) {
            int node = nodekoe.remove();
            noderIKoe.put(node, noderIKoe.get(node) + 1);

            if (noderIKoe.get(node) > 2*grafen.antallNoder()) {
                throw new IllegalArgumentException("Grafen har en negativ sykel!");
            }

            int nvKostnad = grafen.getKostnad(node);
            List<Integer> naboer = grafen.getNeighbours(node);

            for (int nabo: naboer) {
                int nabokostnad = grafen.getWeight(node, nabo) + nvKostnad;
                if (nabokostnad < grafen.getKostnad(nabo)) {
                    grafen.setKostnad(nabo, nabokostnad);
                    grafen.setForrigePaaVeien(nabo, node);
                    Integer antallganger = noderIKoe.get(nabo);

                    if (antallganger == null) {
                        nodekoe.add(nabo);
                        noderIKoe.put(nabo, 1);
                    } else {
                        // Hvis noderIKoe er et partall er den ikke i køen.
                        if (noderIKoe.get(nabo) % 2 == 0) {
                            nodekoe.add(nabo);
                            noderIKoe.put(nabo, noderIKoe.get(nabo) + 1);
                        }
```

```
                }
            }
        }
    }

    int nvNode = sluttnode;
    while (nvNode != KortesteVeiGraf.HAR_IKKE_FORRIGE) {
        System.out.println(grafen.getElement(nvNode) + ", " + grafen.getKostnad(nvNode));
        nvNode = grafen.getForrigePaaVeien(nvNode);
    }
  }
}
```

Maks poeng: 12