

i Introduksjon til DAT200 eksamen

EMNE: DAT200 Algoritmer og Datastrukturer

DATO OG KLOKKESLETT: 2. mars 2020 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 programeksemples 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 algoritme, enkel**

Hva er kjøretida til funksjonen fakultet, oppgitt under?

```
def fakultet(tall):  
    resultat = 1  
    for n in range(1, tall+1):  
        resultat *= n  
    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)

Java-versjon:

```
public static long fakultet(int tall) {  
    int resultat = 1;  
    for (int n=1;n<=tall;n++) {  
        resultat *= n;  
    }  
    return resultat;  
}
```

Maks poeng: 4

2

Analyse av algoritme, middels

Hva er kjøretida til funksjonen `finn_primtall`, oppgitt under?

```
def finn_primtall(til):
    print("1\n2\n3")
    for i in range(4, til+1):
        delelig = False
        for deletall in range(2, i//2 + 1):
            if i % deletall == 0:
                delelig = True
                break
        if not delelig:
            print(str(i))
```

Velg ett alternativ

- ☐ $O(1)$
- ☐ $O(\log(n))$
- ☐ $O(n)$
- ☐ $O(n \cdot \log(n))$
- ☐ $O(n^2)$
- ☐ $O(n^3)$
- ☐ $O(2^n)$

Java-versjon:

```
public static void finn_primtall(int maksimalverdi) {
    System.out.println("1\n2\n3");
    for (int i=4;i<=maksimalverdi;i++) {
        boolean delelig = false;
        for (int deletall = 2; deletall <= (i/2)+1; deletall++) {
            if (i % deletall == 0) {
                delelig = true;
                break;
            }
        }
        if (!delelig) {
            System.out.println(i);
        }
    }
}
```

Maks poeng: 4

3

Analyse av algoritme, avansert

Hva er kjøretida til følgende funksjon i O-notasjon? Anta at vekt_liste og verdi_liste er standard Python lister (Arraybaserte lister).

```
def knapsack(kapasitet, vekt_liste, verdi_liste):
    verdi_matrise = []
    for i in range(len(vekt_liste)):
        verdi_matrise.append([])
    for j in range(kapasitet + 1):
        verdi_matrise[0].append(0)
    for i in range(1, len(vekt_liste)):
        for j in range(kapasitet + 1):
            if vekt_liste[i] > j:
                verdi_matrise[i].append(verdi_matrise[i-1][j])
            else:
                verdi_matrise[i].append(maksimum(verdi_matrise[i-1][j], verdi_matrise[i-1][j-vekt_liste[i]] + verdi_liste[i]))
    return verdi_matrise[-1][-1]
```

Skriv ditt svar her...

Java-versjon:

```
public static int knapsack(int kapasitet, int[] vekt_liste, int[] verdi_liste) {
    int[][] verdi_matrise = new int[vekt_liste.length][kapasitet+1];
    for (int i=0;i<=kapasitet;i++) {
        verdi_matrise[0][i] = 0;
    }
    for (int i=1;i<vekt_liste.length;i++) {
        for (int j=0;j<=kapasitet;j++) {
            if (vekt_liste[i] > j) {
                verdi_matrise[i][j] = verdi_matrise[i-1][j];
            } else {
                verdi_matrise[i][j] = Math.max(verdi_matrise[i-1][j], verdi_matrise[i-1][j-vekt_liste[i]] +
verdi_liste[i]);
            }
        }
    }
    return verdi_matrise[vekt_liste.length - 1][kapasitet];
}
```

Maks poeng: 6

4 **Konvertering mellom listetyper**

Hvis du har en lenket liste og ønsker å bruke en algoritme som er raskere på en array-liste, kan den lenkede lista konverteres til en array-liste. Skriv en algoritme for dette. Hva er kjøretida til algoritmen i O-notasjon? Algoritmen blir vurdert både basert på om den er korrekt og hva kjøretida dens er i O-notasjon. Du kan skrive algoritmen i pseudokode eller Python kode.
Skriv ditt svar her...

1	
---	--

Maks poeng: 8

5 **Splitt og hersk**

Beskriv programmeringsteknikken Splitt og Hersk, gjerne gjennom et eksempel. Hvilke av sorteringsalgoritmene nevnt under bruker programmeringsteknikken splitt-og-hersk?

Sorteringsalgoritmene

- Innsettingssortering (Insertion Sort)
- Shell Sort
- Flettesortering (Merge Sort)
- QuickSort
- Tellesortering (Counting Sort)

Skriv ditt svar her...

Format ▾ | **B** *I* U \times_2 \times^2 | *I_x* | | | | |

Words: 0

Maks poeng: 10

6 **Rekursiv funksjon**

Skriv en rekursiv funksjon (Java: statisk metode) som tar inn en streng som representerer et ord, og finner ut om ordet er et palindrom. Et palindrom er et ord som blir likedan om det leses baklengs (fra høyre mot venstre). Eksempler på palindromer er ada, anna, rotor og regninger. Funksjonen din skal returnere True hvis ordet er et palindrom og False ellers.

Skriv ditt svar her...

1	
---	--

Maks poeng: 6

7 **Hashtabell, innsetting**

Sett inn elementene 7, 9, 15, 18, 17, 27 i hashtabellen under i den oppgitte rekkefølgen. Bruk modulo som hashfunksjon. Bruk kvadratisk prøving hvis det blir kollisjoner. Marker i tabellen under hvor hvert tall havner.

Finn de som passer sammen

	27	17	18	7	9	15	Tom
5	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
0	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
4	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
2	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
7	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
8	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
6	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
1	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
3	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Maks poeng: 9

8 **Om Hashtabeller**

Definer følgende termer rundt hashtabeller, 3% hver

- fyllingsgrad
- hashfunksjon
- kollisjon

Skriv ditt svar her...

Format ▾ | **B** *I* U x_2 x^2 I_x | | | | | |

Words: 0

Maks poeng: 9

9 **Hva er et tre**

Hva er datastrukturen tre? Definer datastrukturen.
Skriv ditt svar her...

Format

B

I

U

x_2

x^2

I_x

ABC

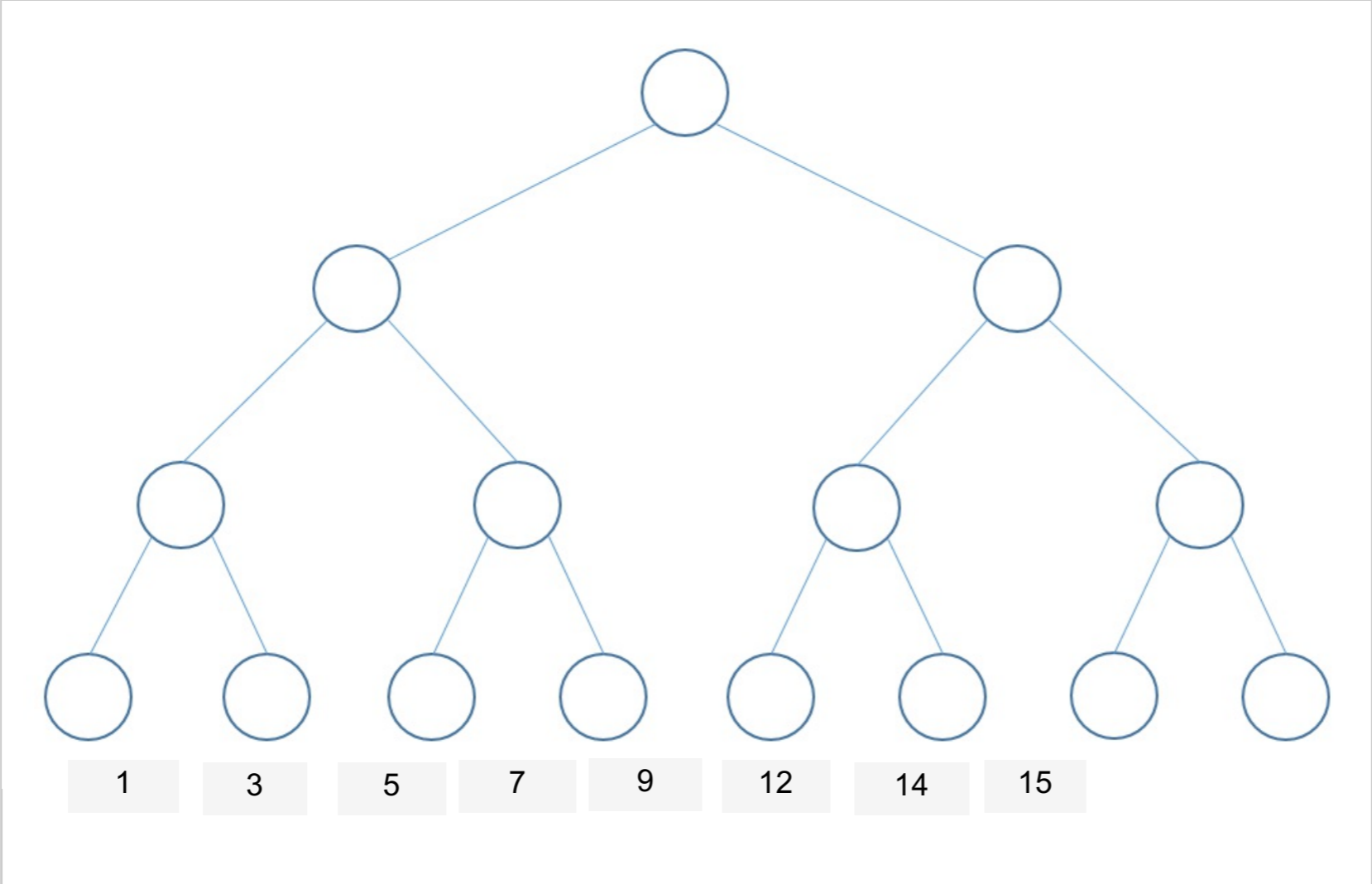
Words: 0

Maks poeng: 6

11 **Haug: Innsetting**

Start med en tom minimumshaug. Sett inn tallene 5, 7, 1, 9, 12, 3, 15, 14 i den rekkefølgen. Vis på figuren under hvordan haugen blir ved å trekke tallene inn i riktig node.

Flytt tallene inn i riktig node



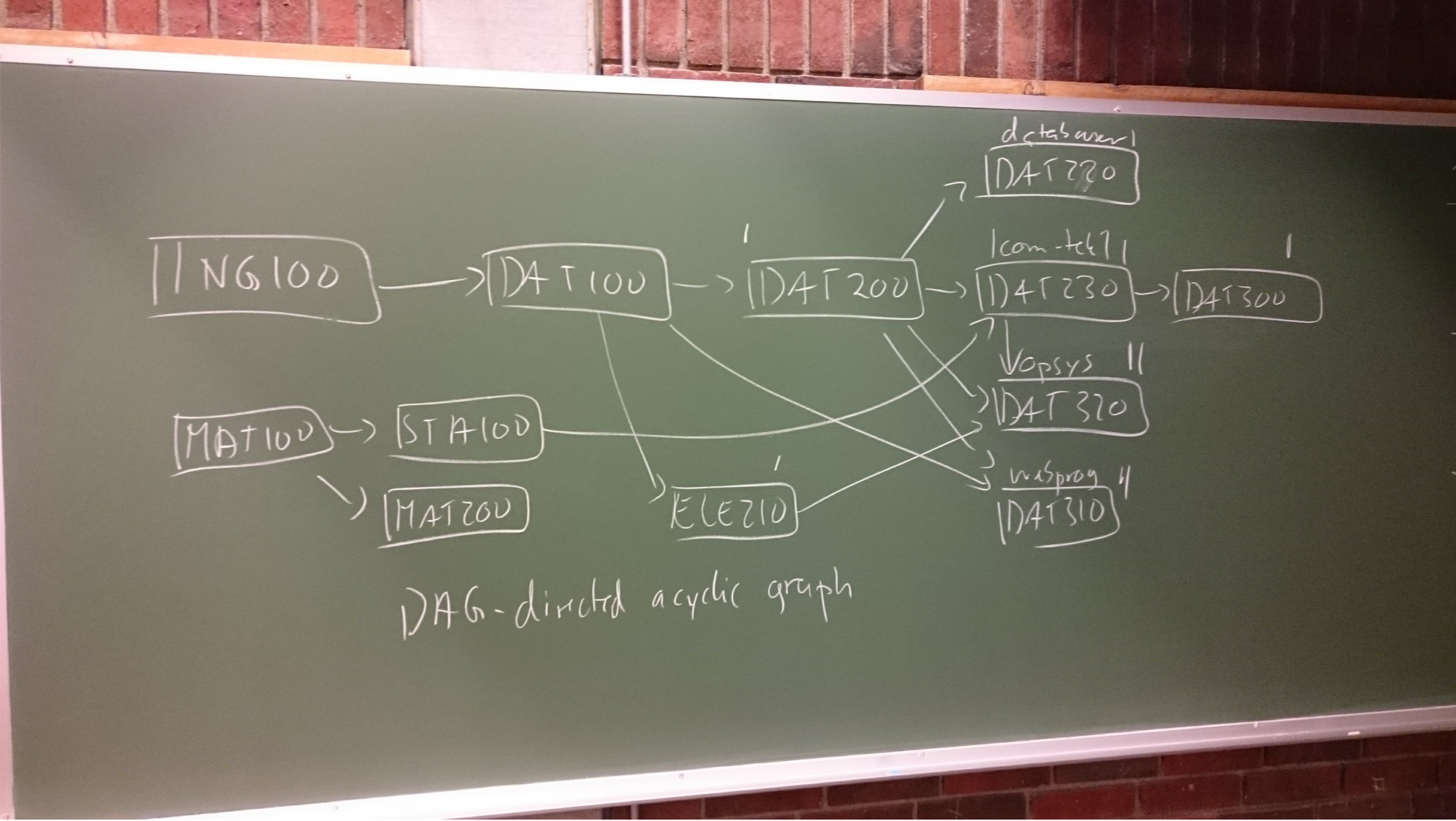
Maks poeng: 8

12 **Grafalgoritme**

Gitt en DAG over emner og hva de forutsetter. Skriv en algoritme for følgende problem: Gitt et emne, list ut alle emnene som du må ha tatt for å ta dette emnet, inkludert indirekte forutsetninger. Et eksempel på en indirekte forutsetning er hvis DAT220 forutsetter DAT200, og DAT200 forutsetter DAT100, så vil DAT220 indirekte forutsette DAT100. List ut alle emnene som det oppgitte emnet forutsetter. Rekkefølgen på emnene skal være en lovlig rekkefølge man kan ta emnene.

Du kan skrive algoritmen i pseudokode, Java kode eller Python kode. Kode for selve grafen er oppgitt under i Python og Java.

Oppgaven blir evaluert både basert på om algoritmen virker og hva kjøretida dens er i O-notasjon.



1	
---	--

Graf, Python:

```
class Node:
    def __init__(self, verdi):
        self.verdi = verdi
        self.kostnad = None
        self.forrige = None
        self.naboer = {}
# Plassbruk: Theta(V + E)
class NabolisteGraf:
    # Kjøretid O(1)
    def __init__(self):
        self.__nodeliste = []
    # Returnerer indeksen til noden i node-lista
    # O(1)
    def add_node(self, verdi):
        ny_node = Node(verdi)
        self.__nodeliste.append(ny_node)
        return len(self.__nodeliste)-1
    # Tar inn indeksene til fra-noden og til-noden
    # O(1)
    def add_edge(self, fra, til, vekt):
        self.__nodeliste[fra].naboer[til] = vekt
    # Returnerer en liste med indeksene til alle naboene.
    # Kjøretid Theta(antall naboer)
    def get_neighbours(self, node_indeks):
        naboliste = []
        for nabo in self.__nodeliste[node_indeks].naboer:
            naboliste.append(nabo)
        return naboliste
    # O(1)
    def get_weight(self, fra, til):
        if til not in self.__nodeliste[fra].naboer:
            return None
        return self.__nodeliste[fra].naboer[til]
    # O(1)
    def get_element(self, node_index):
        return self.__nodeliste[node_index].verdi
    # O(1)
    def get_kostnad(self, node_index):
        return self.__nodeliste[node_index].kostnad
    # O(1)
    def set_kostnad(self, node_index, kostnad):
        self.__nodeliste[node_index].kostnad = kostnad
    # O(V)
    def fjern_kostnader(self):
        for node in self.__nodeliste:
            node.kostnad = None
            node.forrige = None
    def get_forrige(self, node_index):
        return self.__nodeliste[node_index].forrige
    def set_forrige(self, node_index, forrige_index):
        self.__nodeliste[node_index].forrige = forrige_index
    # O(1)
    def antall_noder(self):
```

```

        return len(self.__nodeliste)
    def __len__(self):
        return self.antall_noder()

```

Graf. Java

```

public class Naboliste <E> implements KortesteVeiGraf<E> {
    private class Node {
        E element;
        HashMap<Integer, Integer> edges; // HashMap tilNodeIndex -> Vekt
        int kostnad;
        int forrige;

        public Node(E element) {
            this.element = element;
            edges = new HashMap<>();
        }
    }

    private ArrayList<Node> nodes;

    public Naboliste() {
        nodes = new ArrayList<>();
    }
    /*
    * Kjøretid O(1)
    */
    @Override
    public int addNode(E innhold) {
        Node nynode = new Node(innhold);
        nodes.add(nynode);
        nynode.kostnad = Integer.MAX_VALUE;
        nynode.forrige = KortesteVeiGraf.HAR_IKKE_FORRIGE;
        return nodes.size() - 1;
    }
    /*
    * Kjøretid O(1)
    */
    @Override
    public void addEdge(int fra, int til, int vekt) {
        if (fra >= nodes.size() || til >= nodes.size()) {
            throw new IndexOutOfBoundsException("Fra eller til er større enn antall noder");
        }
        Node fraNode = nodes.get(fra);
        fraNode.edges.put(til, vekt);
    }
    /*
    * Kjøretid O(antall naboer)
    *
    * Tett graf: Antall naboer er O(V)
    * Glissen graf: Antall naboer er O(1)
    */
    @Override
    public List<Integer> getNeighbours(int node) {
        Node fraNode = nodes.get(node);
        return new ArrayList<Integer>(fraNode.edges.keySet());
    }
    /*
    * Kjøretid O(1)
    */
    @Override
    public int getWeight(int fra, int til) {
        Node fraNode = nodes.get(fra);
        Integer vekt = fraNode.edges.get(til);
        if (vekt == null) return Graf.INGEN_KANT;
        return vekt;
    }
    /*
    * Kjøretid O(1)

```

```

    */
    @Override
    public E getElement(int node) {
        return nodes.get(node).element;
    }
    @Override
    public int getKostnad(int node) {
        return nodes.get(node).kostnad;
    }
    @Override
    public void setKostnad(int node, int kostnad) {
        Node noden = nodes.get(node);
        noden.kostnad = kostnad;
    }
    @Override
    public int getForrigePaaVeien(int node) {
        return nodes.get(node).forrige;
    }
    @Override
    public void setForrigePaaVeien(int node, int forrige) {
        Node noden = nodes.get(node);
        noden.forrige = forrige;
    }
    @Override
    public void reset() {
        for (Node noden: nodes) {
            noden.kostnad = Integer.MAX_VALUE;
            noden.forrige = KortesteVeiGraf.HAR_IKKE_FORRIGE;
        }
    }
    @Override
    public int antallNoder() {
        return nodes.size();
    }
}

```

Maks poeng: 10

13 **Dijkstra's Algoritme**

Skriv Dijkstra's Algoritme, i pseudokode, Python eler Java kode. Du kan bruke grafrepresentasjonen oppgitt under, som er lik den fra forrige oppgave.

Words: 0

Graf, Python:

```

class Node:
    def __init__(self, verdi):
        self.verdi = verdi
        self.kostnad = None
        self.forrige = None
        self.naboer = {}
# Plassbruk: Theta(V + E)
class NabolisteGraf:
    # Kjøretid O(1)
    def __init__(self):
        self.__nodeliste = []
    # Returnerer indeksen til noden i node-lista
    # O(1)
    def add_node(self, verdi):
        ny_node = Node(verdi)
        self.__nodeliste.append(ny_node)
        return len(self.__nodeliste)-1
    # Tar inn indeksene til fra-noden og til-noden
    # O(1)
    def add_edge(self, fra, til, vekt):
        self.__nodeliste[fra].naboer[til] = vekt
    # Returnerer en liste med indeksene til alle naboene.
    # Kjøretid Theta(antall naboer)
    def get_neighbours(self, node_indeks):
        naboliste = []
        for nabo in self.__nodeliste[node_indeks].naboer:
            naboliste.append(nabo)
        return naboliste
    # O(1)
    def get_weight(self, fra, til):
        if til not in self.__nodeliste[fra].naboer:
            return None
        return self.__nodeliste[fra].naboer[til]
    # O(1)
    def get_element(self, node_index):
        return self.__nodeliste[node_index].verdi
    # O(1)
    def get_kostnad(self, node_index):
        return self.__nodeliste[node_index].kostnad
    # O(1)
    def set_kostnad(self, node_index, kostnad):
        self.__nodeliste[node_index].kostnad = kostnad
    # O(V)
    def fjern_kostnader(self):
        for node in self.__nodeliste:
            node.kostnad = None
            node.forrige = None
    def get_forrige(self, node_index):
        return self.__nodeliste[node_index].forrige
    def set_forrige(self, node_index, forrige_index):
        self.__nodeliste[node_index].forrige = forrige_index
    # O(1)
    def antall_noder(self):
        return len(self.__nodeliste)

```

```
def __len__(self):
    return self.antall_noder()
```

Graf. Java

```
public class Naboliste <E> implements KortesteVeiGraf<E> {
    private class Node {
        E element;
        HashMap<Integer, Integer> edges; // HashMap tilNodeIndex -> Vekt
        int kostnad;
        int forrige;

        public Node(E element) {
            this.element = element;
            edges = new HashMap<>();
        }
    }

    private ArrayList<Node> nodes;

    public Naboliste() {
        nodes = new ArrayList<>();
    }
    /*
    * Kjøretid O(1)
    */
    @Override
    public int addNode(E innhold) {
        Node nynode = new Node(innhold);
        nodes.add(nynode);
        nynode.kostnad = Integer.MAX_VALUE;
        nynode.forrige = KortesteVeiGraf.HAR_IKKE_FORRIGE;
        return nodes.size() - 1;
    }
    /*
    * Kjøretid O(1)
    */
    @Override
    public void addEdge(int fra, int til, int vekt) {
        if (fra >= nodes.size() || til >= nodes.size()) {
            throw new IndexOutOfBoundsException("Fra eller til er større enn antall noder");
        }
        Node fraNode = nodes.get(fra);
        fraNode.edges.put(til, vekt);
    }
    /*
    * Kjøretid O(antall naboer)
    *
    * Tett graf: Antall naboer er O(V)
    * Glissen graf: Antall naboer er O(1)
    */
    @Override
    public List<Integer> getNeighbours(int node) {
        Node fraNode = nodes.get(node);
        return new ArrayList<Integer>(fraNode.edges.keySet());
    }
    /*
    * Kjøretid O(1)
    */
    @Override
    public int getWeight(int fra, int til) {
        Node fraNode = nodes.get(fra);
        Integer vekt = fraNode.edges.get(til);
        if (vekt == null) return Graf.INGEN_KANT;
        return vekt;
    }
    /*
    * Kjøretid O(1)
    */
}
```

```

    */
    @Override
    public E getElement(int node) {
        return nodes.get(node).element;
    }
    @Override
    public int getKostnad(int node) {
        return nodes.get(node).kostnad;
    }
    @Override
    public void setKostnad(int node, int kostnad) {
        Node noden = nodes.get(node);
        noden.kostnad = kostnad;
    }
    @Override
    public int getForrigePaaVeien(int node) {
        return nodes.get(node).forrige;
    }
    @Override
    public void setForrigePaaVeien(int node, int forrige) {
        Node noden = nodes.get(node);
        noden.forrige = forrige;
    }
    @Override
    public void reset() {
        for (Node noden: nodes) {
            noden.kostnad = Integer.MAX_VALUE;
            noden.forrige = KortesteVeiGraf.HAR_IKKE_FORRIGE;
        }
    }
    @Override
    public int antallNoder() {
        return nodes.size();
    }
}

```

Maks poeng: 10