Introduksjon til DAT200 eksamen

EKSAMEN I: DAT200 Algoritmer og Datastrukturer

DATO: 07.12 2018
TID FOR EKSAMEN: 4 timer
TILLATTE HJELPEMIDDEL: Ingen trykte og håndskrevne hjelpemidler, standard enkel kalkulator

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

For oppgavene som krever programmering: Det viktigste er at du viser at du forstår hvordan en løsning kan programmeres i Java. 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 C++-lik syntaks 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 Dominante ledd (4% total)

Hva er de dominante leddet i følgende funksjoner?

f(n) = 2*n + 6*log(n) + 25
Velg ett alternativ

- 2*n
- 6*log(n)
- 25

f(n) = n^3 + 2*n + 6 + 1,5^n

Velg ett alternativ

- n^3
- 2*n
- 6
- 1,5^n

Maks poeng: 4

2 O-notasjon (8%)

a) Hva er O-notasjon og hvorfor bruker man O-notasjon for å angi kjøretidene til algoritmer?

b) Hva er Omega og Theta notasjon? Beskriv forskjellen mellom O-notasjon og disse to notasjonene.

Skriv ditt svar her...

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

Words: 0

Maks poeng: 8

3 Kjøretid, enkel metode (4%)

Hva er kjøretida til følgende metode i O-notasjon?

```
public static int[] reverser(int[] array) {
    int[] nyListe = new int[array.length];
    for (int i=0;i<array.length;i++) {
        nyListe[i] = array[array.length - i - 1];
    }
    return nyListe;
}
```

Velg ett alternativ

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

Maks poeng: 4

4 Kjøretid, rekursiv metode (6%)

Hva er kjøretida til følgende metode i O-notasjon?

```
public static int rekursivMetode(List<Integer> liste) {
    if (liste.size() == 0) return 0;
    if (liste.size() == 1) return liste.get(0);
```

```
        if (liste.size() == 2) return liste.get(0) + liste.get(1);
        return rekursivMetode(liste.subList(0, liste.size()/2)) + rekursivMetode(liste.subList(liste.size()/2,
liste.size()));
    }
```

Velg ett alternativ

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

Maks poeng: 6

5 Rekursjon, Filsøking (8%)

Skriv en rekursiv Java metode som leiter etter ei fil med et bestemt navn i en katalogstruktur. Metoden skal ta en File referanse til katalogen den skal starte å leite i samt en String med navnet på fila den leiter etter. Metoden skal returnere en File referanse til fila den leiter etter, eller null hvis den ikke finner fila. Oppgi også kjøretida til programmet ditt i O-notasjon.

Skriv ditt svar her...

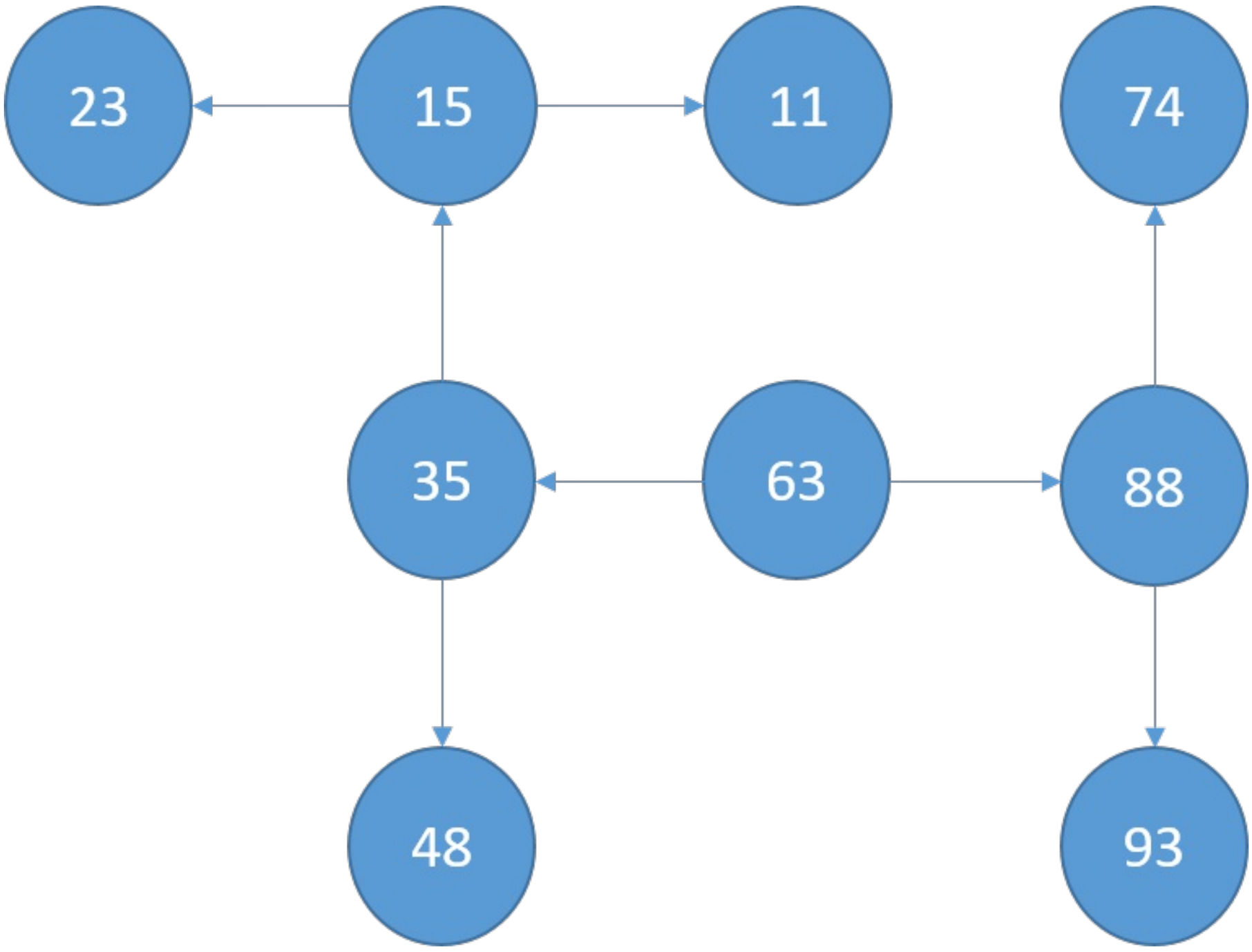
```
1
```

Metodenavn til bruk i denne oppgaven:
- file.isDirectory() spør ei fil om den er ei mappe (true) eller ei vanlig fil (false)
- file.listFiles() returnerer en File[] hvis den kalles på ei mappe
- file.getName() returnerer navnet på fila

Maks poeng: 8

6 Identifiser datastrukturen (4%)

Hvilken datastruktur er tegnet opp på bildet under? Desto mer spesifikk du er desto høyere scorer du på oppgaven.



Skriv ditt svar her...

Maks poeng: 4

7 Lenkete Lister (9%)

Beskriv datastrukturen "lenket liste" med tekst og gjerne en scannet figur. Hvordan er dataene strukturert i en lenket liste?

Svar i tillegg på følgende spørsmål:

- Beskriv algoritmene for operasjonene addLast, getFirst og get(index) på ei lenket liste, med pseudokode eller Java kode
- Hva er kjøretida til addLast, getFirst og get(index) og remove(index) operasjonene på ei lenket liste?

Skriv ditt svar her...

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

Words: 0

Maks poeng: 9

8 Kjøretid for sorteringsalgoritmer (6% total, 1 pr. riktig svar)

Oppgi kjøretidene for de ulike sorteringsalgoritmene i tabellen under.

Finn de som passer sammen

	O(n*n)	O(n)	O(1)	O(n*log(n))	O(log(n))
Insertion Sort, best case	○	○	○	○	○
Quicksort, best case	○	○	○	○	○
Quicksort, worst case	○	○	○	○	○
Merge Sort, best case	○	○	○	○	○
Insertion Sort, worst case	○	○	○	○	○
Merge Sort, Worst Case	○	○	○	○	○

Maks poeng: 6

9 In-place sortering (6%)

Hva vil det si at en sorteringsalgoritme er in-place? Hva er fordelene og eventuelt ulempene ved in-

place sorteringsalgoritmer sammenliknet med de andre? Hvilke av sorteringsalgoritmene fra pensum (insertion sort, shellsort, merge sort, quicksort, counting sort) er in-place?

Skriv ditt svar her...

Format

Words: 0

Maks poeng: 6

10 Hashtabell, innsetting (9%)

Sett inn elementene 45, 23, 35, 7, 1, 5, 27 i hashtabellen under i den oppgitte rekkefølgen. Bruk modulo som hashfunksjon. Bruk kvadratisk prøving i tilfelle kollisjoner. Marker i tabellen under hvor hvert tall havner.

Finn de som passer sammen

	Tom	1	5	7	23	27	35	45
0								
1								
2								
3								
4								
5								
6								
7								
8								

Maks poeng: 9

11 Hashfunksjoner (6%)

11 Hashfunksjoner (6%)

Hvilken egenskaper skal en god hashfunksjon ha og hvorfor?

Skriv ditt svar her...

Format

Words: 0

Maks poeng: 6

12 Valg av datastrukturer (10%)

Gitt følgende oppgave:

Du skal lage et system for håndtering av ordre for kunder i en nettbutikk. Du har tre klasser, ordre, kunde og vare. Du ønsker å gjøre følgende operasjoner:

1: Finn kunde basert på brukernavn

2: For en kunde, finn alle ordre sortert etter tidsfrist

3: Finn varer basert på kategori

4: Legg inn en ny ordre

5: Finn ordren med kortest tidsfrist igjen (uansett kunde). Denne skal behandles først og skal derfor normalt sett fjernes snart etter den er funnet.

6: Finn vare basert på navn, ofte et delvis navn. Du har et søkefelt hvor bruker skal starte å skrive et navn og så skal systemet foreslå personer som matcher så lenge det er mindre enn 10 personer.

Hvilke datastrukturer ville du ha valgt for å implementere disse operasjonene? Begrunn svaret ditt. Merk at samme struktur muligens kan brukes for flere operasjoner.

Skriv ditt svar her...

Format

Words: 0

Egenskapene til de tre klassene:
Kunde (brukernavn, gateadresse, postnummer, poststed, telefon, epost, ordre datastruktur)
Vare (ID, navn, kategori, beskrivelse, bilde, pris)
Ordre (ID, lagt inn dato, tidsfrist, Kunde, List<Vare>)

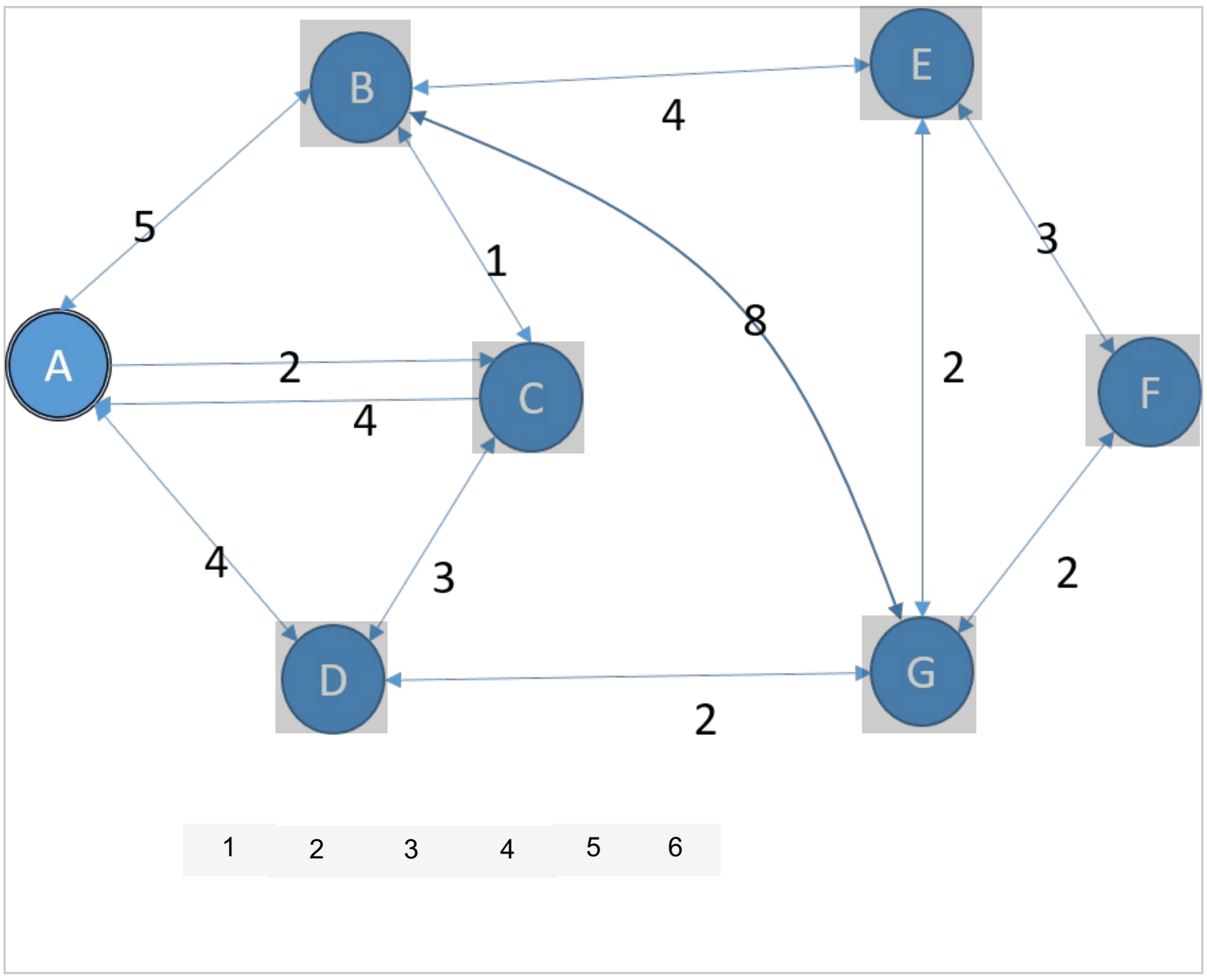
Maks poeng: 10

13 Dijkstra's Algoritme (6%)

Gitt følgende graf, og at node A er startnoden til algoritmen:

Oppgi hvilken rekkefølge nodene i grafen blir besøkt av Dijkstra's Algoritme gjennom å trekke de seks tallene oppgitt nederst inn i riktig node i grafen. Så hvis du mener at node G blir besøkt først, trekk tallet 1 inn i node G.

Velg ett alternativ



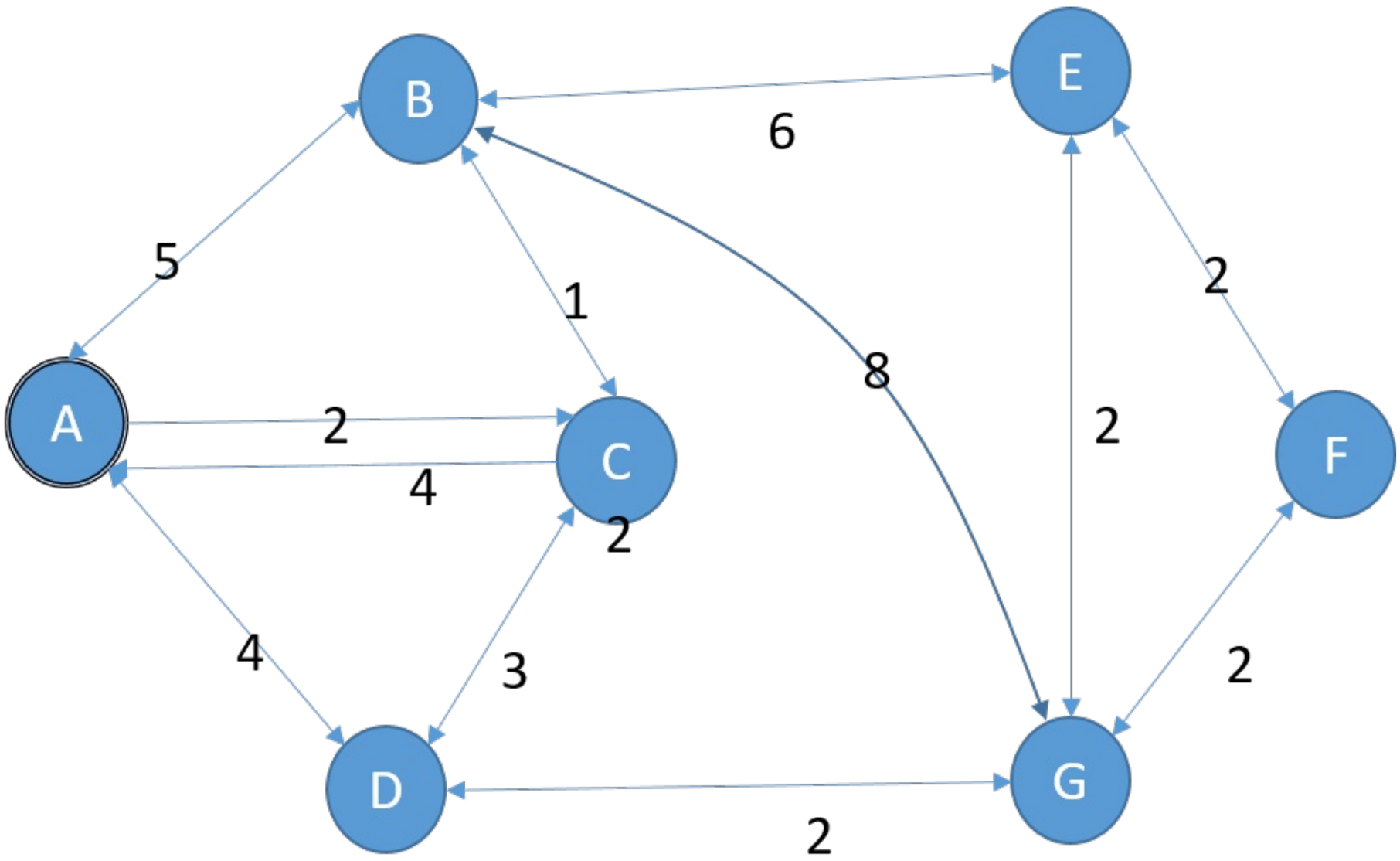
Maks poeng: 6

14 Variant graf (8%)

Gitt en graf hvor det kan være vekter i nodene i tillegg til i kantene, beskriv hvordan grafen kan endres slik at veifinningsalgoritmene fra pensum likevel kan brukes på grafen. Vekten i en node er en ekstra kostnad ved å bevege seg ut av noden.

Eksempel: Hvis grafen representerer et veifinningssystem så kan noen noder representere lyskryss. Kostnaden til en slik node er forventet tid du må vente på grønt lys.

Svar på oppgaven ved å beskrive hva som må gjøres i feltet under, samt tegn området rundt node C i eksempelgrafen oppgitt her på papir slik det blir seende ut etter din endring.



Skriv ditt svar her...

Format

Words: 0

Maks poeng: 8

15 Nabomatrise som korteste vei graf (6%)

I de utdelte filene er det to måter å representere grafer på, nabolister og nabomatriser. Så ble det laget en korteste vei graf som inneholder noen flere metoder, og naboliste representasjonen ble endret for å oppfylle den nye kontrakten.

Beskriv hva som må gjøres med nabomatrise representasjonen for at den skal kunne brukes som en korteste vei graf. Det vil si, hva må gjøres med klassen Nabomatrise oppgitt under for at den skal implementere KortesteVeiGraf grensesnittet. Du kan bruke tekst eller Java kode. Java kode for nabomatrise samt korteste vei graf grensesnittet er oppgitt under.

Skriv ditt svar her...

```
public interface Graf <E> {

    public static final int INGEN_KANT = Integer.MAX_VALUE;
```

```
    /*
* Returnerer en ID for noden
*/
    public int addNode(E innhold);
    public void addEdge(int fra, int til, int vekt);
    public List<Integer> getNeighbours(int node);
    public int getWeight(int fra, int til);
    public E getElement(int node);
    public int antallNoder();
}

public interface KortesteVeiGraf <E> extends Graf<E> {

    public static final int HAR_IKKE_FORRIGE = -1;

    public int getKostnad(int node);
    public void setKostnad(int node, int kostnad);
    public int getForrigePaaVeien(int node);
    public void setForrigePaaVeien(int node, int forrige);

    // Resetter kostnader of forrige-referanser slik at man kan kjøre flere
    // tester på samme graf.
    public void reset();
}

public class Nabomatrise <E> implements Graf<E> {

    int[][] vekter;
  ArrayList<E> elementer;

    public Nabomatrise(int antallNoder) {
        vekter = new int[antallNoder][antallNoder];
        for (int i=0;i<antallNoder;i++) {
            for (int j=0;j<antallNoder;j++) {
                vekter[i][j] = Graf.INGEN_KANT;
    }
  }
        elementer = new ArrayList<>();
}

    @Override
    public int addNode(E innhold) {
        if (elementer.size() >= vekter.length) {
```

```
                    throw new IndexOutOfBoundsException("Har lagt til flere elementer enn det er i
arrayen!");
        }
            elementer.add(innhold);
            return elementer.size() - 1;
    }

        @Override
        public void addEdge(int fra, int til, int vekt) {
            vekter[fra][til] = vekt;
}

    @Override
    public List<Integer> getNeighbours(int node) {
        ArrayList<Integer> naboer = new ArrayList<>();
        for (int j=0;j<elementer.size();j++) {
            if (vekter[node][j] != Graf.INGEN_KANT) {
                naboer.add(j);
    }
    }
        return naboer;
}

    @Override
    public int getWeight(int fra, int til) {
        return vekter[fra][til];
 }

    @Override
    public E getElement(int node) {
        return elementer.get(node);
}

    @Override
    public int antallNoder() {
        return vekter.length;
    }
}
```

Maks poeng: 6