PRAKTIKUM ALGORITMA DAN STRUKTUR DATA

JOBSHEET PERTEMUAN KE-13

NAMA : ALVINO VALERIAN D.R

KELAS : 1A

NO. ABSEN : 05

NIM : 2341720027

PROGRAM STUDI TEKNIK INFORMATIKA

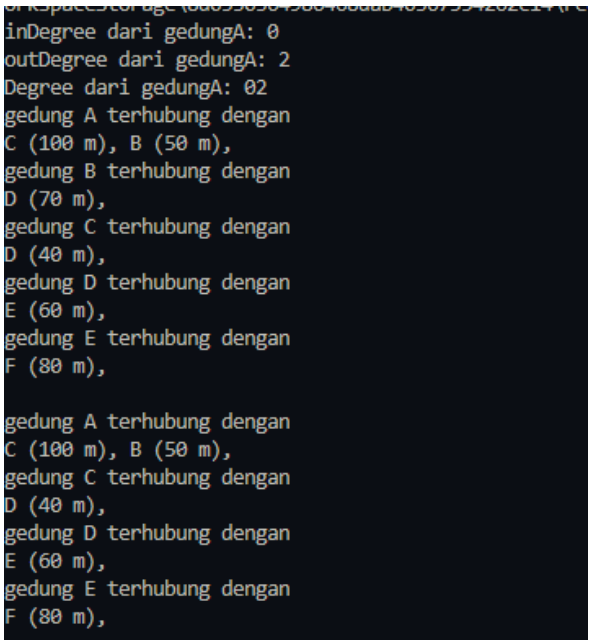
JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

2024

2.1.2 Verifikasi Hasil Percobaan

Verifikasi hasil kompilasi kode program Anda dengan gambar berikut ini.



2.1.3 Pertanyaan

1. Perbaiki kode program Anda apabila terdapat error atau hasil kompilasi kode tidak sesuai!

2. Pada class Graph, terdapat atribut **list[]** bertipe DoubleLinkedList. Sebutkan tujuan pembuatan variabel tersebut!

3. Jelaskan alur kerja dari method **removeEdge**!

4. Apakah alasan pemanggilan method **addFirst()** untuk menambahkan data, bukan method add jenis lain saat digunakan pada method **addEdge** pada class Graph?

5. Modifikasi kode program sehingga dapat dilakukan pengecekan apakah terdapat jalur antara suatu node dengan node lainnya, seperti contoh berikut (Anda dapat memanfaatkan Scanner).

JAWAB

```
Masukkan gedung asal: 2
Masukkan gedung tujuan: 3
Gedung C dan D bertetangga

Masukkan gedung asal: 2
Masukkan gedung tujuan: 5
Gedung C dan F tidak bertetangga
```

JAWAB

1.tidak ada eror

2. Variabel list[] bertipe DoubleLinkedLists05 digunakan untuk menyimpan daftar adjacency dari setiap vertex dalam graf. Setiap elemen dalam array ini merupakan linked list yang menyimpan tetangga (neighbor) dari vertex tersebut bersama dengan jaraknya.

3. Method removeEdge berfungsi untuk menghapus edge antara dua vertex. Langkah-langkahnya adalah sebagai berikut:

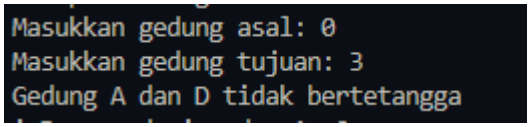
1.Method menerima dua parameter: asal dan tujuan.

2.Method memanggil remove pada linked list dari vertex asal untuk menghapus vertex tujuan.

3.Jika graf adalah undirected, method juga memanggil remove pada linked list dari vertex tujuan untuk menghapus vertex asal.

4. Method addFirst digunakan dalam addEdge untuk menambahkan data karena efisien dalam menambahkan elemen baru di awal linked list. Ini memastikan bahwa penambahan edge dilakukan dengan cepat, karena penambahan di awal linked list memiliki kompleksitas waktu O(1).

5.




```
public boolean cekJalur(int asal, int tujuan) {
        for (int i = 0; i < list[asal].size(); i++) {
            try {
                if (list[asal].get(i) == tujuan) {
                    return true;
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        return false;
    }

    public void cetakJalur(int asal, int tujuan) {
        if (cekJalur(asal, tujuan)) {
            System.out.println("Gedung " + (char) ('A' + asal) + "
dan " + (char) ('A' + tujuan) + " bertetangga");
        } else {
            System.out.println("Gedung " + (char) ('A' + asal) + "
dan " + (char) ('A' + tujuan) + " tidak bertetangga");
        }
        System.out.println("");
    }
```

```
System.out.print("Masukkan gedung asal: ");
        int asal = scanner.nextInt();

        System.out.print("Masukkan gedung tujuan: ");
        int tujuan = scanner.nextInt();

        gedung.cetakJalur(asal, tujuan);
```

2.2.2 Verifikasi Hasil Percobaan

```
Gedung A: Gedung A (0 m), Gedung B (50 m), Gedung C (0 m), Gedung D (0 m),
Gedung B: Gedung A (60 m), Gedung B (0 m), Gedung C (70 m), Gedung D (0 m),
Gedung C: Gedung A (0 m), Gedung B (80 m), Gedung C (0 m), Gedung D (40 m),
Gedung D: Gedung A (90 m), Gedung B (0 m), Gedung C (0 m), Gedung D (0 m),

hasil setelah penghapusan edge

Gedung A: Gedung A (0 m), Gedung B (50 m), Gedung C (0 m), Gedung D (0 m),
Gedung B: Gedung A (60 m), Gedung B (0 m), Gedung C (70 m), Gedung D (0 m),
Gedung C: Gedung A (0 m), Gedung C (0 m), Gedung D (40 m),
Gedung D: Gedung A (90 m), Gedung B (0 m), Gedung C (0 m), Gedung D (0 m),
D:\alvino\Semester 2\Prak algorita & struktur data\jobsheet13>
```

2.2.3 Pertanyaan

1. Perbaiki kode program Anda apabila terdapat error atau hasil kompilasi kode tidak sesuai!
2. Apa jenis graph yang digunakan pada Percobaan 2?
3. Apa maksud dari dua baris kode berikut?

```
gdg.makeEdge(1, 2, 70);
gdg.makeEdge(2, 1, 80);
```

4. Modifikasi kode program sehingga terdapat method untuk menghitung degree, termasuk inDegree dan outDegree!

JAWABAN

1.tidak adayang eror

2. Graph yang digunakan pada Percobaan 2 adalah graph berarah (directed graph). Ini terlihat dari metode makeEdge(int asal, int tujuan, int jarak) yang menambahkan edge hanya dari asal ke tujuan tanpa menambahkan edge sebaliknya secara otomatis. Jika edge antara dua vertex memiliki arah tertentu, itu berarti graph tersebut adalah graph berarah.

3. gdg.makeEdge(1, 2, 70);: Menambahkan edge dari vertex 1 ke vertex 2 dengan jarak (weight) 70. gdg.makeEdge(2, 1, 80);: Menambahkan edge dari vertex 2 ke vertex 1 dengan jarak (weight) 80. Dua baris kode ini menunjukkan bahwa terdapat dua edge yang terpisah dengan arah yang berbeda dan jarak yang berbeda antara vertex 1 dan vertex 2. Ini adalah ciri khas dari graph berarah.

4.
```
public int getInDegree(int v) {
        int inDegree = 0;
        for (int i = 0; i < vertex; i++) {
            if (matrix[i][v] != -1) {
                inDegree++;
            }
        }
        return inDegree;
    }

    public int getOutDegree(int v) {
        int outDegree = 0;
        for (int i = 0; i < vertex; i++) {
            if (matrix[v][i] != -1) {
                outDegree++;
            }
        }
        return outDegree;
    }

    public void degree(int v) {
        int inDegree = getInDegree(v);
        int outDegree = getOutDegree(v);
        System.out.println("inDegree dari gedung " + (char) ('A' +
v) + ": " + inDegree);
        System.out.println("outDegree dari gedung " + (char) ('A' +
v) + ": " + outDegree);
        System.out.println("Degree dari gedung " + (char) ('A' + v)
+ ": " + (inDegree + outDegree));
    }
}
MAIN:
System.out.println();
        gedung.degree(asal);
```

```
inDegree dari gedungB: 1
outDegree dari gedungB: 0
Degree dari gedungB: 10
```

S

3. Latihan Praktikum
Waktu percobaan: 90 menit

1. Modifikasi kode program pada class **GraphMain** sehingga terdapat menu program yang bersifat dinamis, setidaknya terdiri dari:

a) Add Edge

b) Remove Edge

c) Degree

d) Print Graph

e) Cek Edge

Pengguna dapat memilih menu program melalui input Scanner

2. Tambahkan method **updateJarak** pada Percobaan 1 yang digunakan untuk mengubah jarak antara dua node asal dan tujuan!

3. Tambahkan method **hitungEdge** untuk menghitung banyaknya edge yang terdapat di dalam graf!

JAWAB

```
1. import java.util.Scanner;


public class GraphMainTugas05 {
    public static void main(String[] args) throws Exception{
        Scanner sc = new Scanner(System.in);
        Graph05 gd = new Graph05(6);
        int pilih;
        do {
            System.out.println("=============PILIH============");
            System.out.println("1. Add edge");
            System.out.println("2. Remove Edge");
            System.out.println("3. Degree");
            System.out.println("4. Print Graph");
            System.out.println("5. Cek Edge");
            System.out.println("6. Update Jarak");
            System.out.println("7. Hitung Edge");
            System.out.println("8.Exit");
            System.out.print("Masukkan pilihan: ");
            pilih =sc.nextInt();
```

```
            System.out.println("============================");

            switch (pilih) {
                case 1:
                    System.out.println(" --------ADD EDGE--------
");
                    System.out.print("Masukkan Asal Gedung      :
");
                    int asal = sc.nextInt();
                    System.out.print("Masukkan Tujuan Gedung    :
");
                    int tujuan = sc.nextInt();
                    System.out.print("Masukkan Jarak Gedung     :
");
                    int jarak = sc.nextInt();
                    gd.addEdge(asal, tujuan, jarak);
                    break;
                case 2:
                    System.out.println(" --------REMOVE EDGE--------
");
                    System.out.print("Masukkan Asal Gedung      :
");
                    asal = sc.nextInt();
                    System.out.print("Masukkan Tujuan Gedung    :
");
                    tujuan = sc.nextInt();
                    gd.removeEdge(asal, tujuan);
                    break;
                case 3:
                    System.out.println(" --------- DEGREE ---------
");
                    System.out.print("Masukkan Asal Gedung      :
");
                    asal = sc.nextInt();
                    gd.degree(asal);
```

```
                    break;
                case 4:
                    gd.printGraph();
                    break;
                case 5:
                System.out.println(" --------- CEK EDGE ---------
");
                    System.out.print("Masukkan gedung asal: ");
                    asal = sc.nextInt();
                    System.out.print("Masukkan gedung tujuan: ");
                    tujuan = sc.nextInt();
                    if (gd.cekJalur(asal, tujuan)) {
                        System.out.println("Gedung " + (char) ('A' +
asal) + " dan " + (char) ('A' + tujuan) + " bertetangga");
                    } else {
                        System.out.println("Gedung " + (char) ('A' +
asal) + " dan " + (char) ('A' + tujuan) + " tidak bertetangga");
                    }
                    break;
                case 6:
                System.out.println(" --------- UPDATE JARAK --------
- ");
                    System.out.print("Masukkan gedung asal: ");
                    asal = sc.nextInt();
                    System.out.print("Masukkan gedung tujuan: ");
                    tujuan = sc.nextInt();
                    System.out.print("Masukkan jarak baru: ");
                    jarak = sc.nextInt();
                    gd.updateJarak(asal, tujuan, jarak);
                    break;
                case 7:
                System.out.println(" --------- HITUNG EDGE ---------
");
                    int totalEdge = gd.hitungEdge();
```

```
                    System.out.println("Total edge dalam graf: " +
totalEdge);
                    break;
                case 8:
                System.out.println("--- KELUAR ---");
                    System.exit(1);
                    break;

                default:
                    break;
            }
        } while (pilih != 8);
        sc.close();
    }
}
```

2.

Grap05

```
public void updateJarak(int asal, int tujuan, int jarak) throws
Exception {
        for (int i = 0; i < list[asal].size(); i++) {
            if (list[asal].get(i) == tujuan) {
                list[asal].updateJarak(i, jarak);
                return;
            }
        }
        throw new Exception("Edge tidak ditemukan.");
    }
```

Doublelingkedlist05

```
public void updateJarak(int index, int jarak) throws Exception {
        if (isEmpty() || index >= size) {
            throw new Exception("Nilai index di luar batas");
        }
        Node05 tmp = head;
```

```
        for (int i = 0; i < index; i++) {
            tmp = tmp.next;
        }
        tmp.jarak = jarak;
  }
```

3.

```
public int hitungEdge() {
        int count = 0;
        for (int i = 0; i < vertex; i++) {
            count += list[i].size();
        }
        return count;
    }
```

Main

```
import java.util.Scanner;

public class GraphMainTugas05 {
    public static void main(String[] args) throws Exception{
        Scanner sc = new Scanner(System.in);
        Graph05 gd = new Graph05(6);
        int pilih;
        do {
            System.out.println("=============PILIH============");
            System.out.println("1. Add edge");
            System.out.println("2. Remove Edge");
            System.out.println("3. Degree");
            System.out.println("4. Print Graph");
            System.out.println("5. Cek Edge");
            System.out.println("6. Update Jarak");
            System.out.println("7. Hitung Edge");
            System.out.println("8.Exit");
```

```
            System.out.print("Masukkan pilihan: ");
            pilih =sc.nextInt();
            System.out.println("==============================");

            switch (pilih) {
                case 1:
                    System.out.println(" --------ADD EDGE--------
");
                    System.out.print("Masukkan Asal Gedung      :
");
                    int asal = sc.nextInt();
                    System.out.print("Masukkan Tujuan Gedung    :
");
                    int tujuan = sc.nextInt();
                    System.out.print("Masukkan Jarak Gedung     :
");
                    int jarak = sc.nextInt();
                    gd.addEdge(asal, tujuan, jarak);
                    break;
                case 2:
                    System.out.println(" --------REMOVE EDGE--------
");
                    System.out.print("Masukkan Asal Gedung      :
");
                    asal = sc.nextInt();
                    System.out.print("Masukkan Tujuan Gedung    :
");
                    tujuan = sc.nextInt();
                    gd.removeEdge(asal, tujuan);
                    break;
                case 3:
                    System.out.println(" --------- DEGREE ---------
");
                    System.out.print("Masukkan Asal Gedung      :
");
                    asal = sc.nextInt();
```

```
                    gd.degree(asal);


                    break;
                case 4:
                    gd.printGraph();
                    break;
                case 5:
                System.out.println(" --------- CEK EDGE ---------
");
                    System.out.print("Masukkan gedung asal: ");
                    asal = sc.nextInt();
                    System.out.print("Masukkan gedung tujuan: ");
                    tujuan = sc.nextInt();
                    if (gd.cekJalur(asal, tujuan)) {
                        System.out.println("Gedung " + (char) ('A' +
asal) + " dan " + (char) ('A' + tujuan) + " bertetangga");
                    } else {
                        System.out.println("Gedung " + (char) ('A' +
asal) + " dan " + (char) ('A' + tujuan) + " tidak bertetangga");
                    }
                    break;
                case 6:
                System.out.println(" --------- UPDATE JARAK --------
- ");
                    System.out.print("Masukkan gedung asal: ");
                    asal = sc.nextInt();
                    System.out.print("Masukkan gedung tujuan: ");
                    tujuan = sc.nextInt();
                    System.out.print("Masukkan jarak baru: ");
                    jarak = sc.nextInt();
                    gd.updateJarak(asal, tujuan, jarak);
                    break;
                case 7:
```

```
                System.out.println(" --------- HITUNG EDGE ---------
");
                    int totalEdge = gd.hitungEdge();
                    System.out.println("Total edge dalam graf: " +
totalEdge);
                    break;
                case 8:
                System.out.println("--- KELUAR ---");
                    System.exit(1);
                    break;

                default:
                    break;
            }
        } while (pilih != 8);
        sc.close();
    }
}
```

Graph05

```
public class Graph05 {
    public static boolean cekJalur;
    int vertex;
    DoubleLingkedLists05 list[];

    public Graph05(int v){
        vertex = v;
        list= new DoubleLingkedLists05[v];
        for (int i = 0 ;i < v; i++ ) {
            list[i] = new DoubleLingkedLists05();
        }
    }
    public void addEdge(int asal, int tujuan, int jarak){
        list[asal].addFirst(tujuan, jarak);
```

```
        // undirected graph
        // list[tujuan].addFirst(asal, jarak);
    }

    public void degree(int asal) throws Exception{
        int k, totalIn =0,totalOut = 0;
        for (int i = 0 ; i < vertex; i++ ) {
            // inDegree
            for ( int j = 0; j < list[i].size(); j++) {
                if (list[i].get(j) == asal){
                    ++totalIn;
                }
            }
            // outDegree
            for(k=0 ; k<list[asal].size(); k++){
                list[asal].get(k);
            }
            totalOut =k;
        }
        System.out.println("inDegree dari gedung"+ (char)
('A'+asal)+ ": "+totalIn);
        System.out.println("outDegree dari gedung"+ (char)
('A'+asal)+ ": "+totalOut);
        System.out.println("Degree dari gedung"+ (char) ('A'+asal)+
": "+totalIn + totalOut);
        // undirected graph
        // System.out.println("Degree dari gedung"+ (char)
('A'+asal)+ ": "+list[asal].size());
    }
    public void removeEdge(int asal, int tujuan) throws Exception {
        for (int i = 0; i < vertex; i++){
            if (i == tujuan) {
                list[asal].remove(tujuan);
            }
```

```
        }
    }
    public void removeAllEdges(){
        for(int i=0 ; i<vertex;i++){
            list[i].clear();
        }
        System.out.println("graf berhasil dikosongkan");
    }
    public void printGraph() throws Exception {
        for (int i =0; i < vertex;i++){
            if (list[i].size() > 0) {
                System.out.println("gedung "+(char) ('A' +i) + "
terhubung dengan ");
                for (int j=0 ; j < list[i].size(); j++){
                    System.out.print((char) ('A'+list[i].get(j))+"
("+list[i].getJarak(j)+" m), ");
                }
                System.out.println("");
            }
        }
        System.out.println("");
    }
    public boolean cekJalur(int asal, int tujuan) {
        for (int i = 0; i < list[asal].size(); i++) {
            try {
                if (list[asal].get(i) == tujuan) {
                    return true;
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        return false;
```

```
    }


    public void cetakJalur(int asal, int tujuan) {
        if (cekJalur(asal, tujuan)) {
            System.out.println("Gedung " + (char) ('A' + asal) + "
dan " + (char) ('A' + tujuan) + " bertetangga");
        } else {
            System.out.println("Gedung " + (char) ('A' + asal) + "
dan " + (char) ('A' + tujuan) + " tidak bertetangga");
        }
        System.out.println("");
    }
    public void updateJarak(int asal, int tujuan, int jarak) throws
Exception {
        for (int i = 0; i < list[asal].size(); i++) {
            if (list[asal].get(i) == tujuan) {
                list[asal].updateJarak(i, jarak);
                return;
            }
        }
        throw new Exception("Edge tidak ditemukan.");
    }


    public int hitungEdge() {
        int count = 0;
        for (int i = 0; i < vertex; i++) {
            count += list[i].size();
        }
        return count;
    }
}
```

Output

ADD

```
============PILIH============
1. Add edge
2. Remove Edge
3. Degree
4. Print Graph
5. Cek Edge
6. Update Jarak
7. Hitung Edge
8.Exit
Masukkan pilihan: 1
=============================
 --------ADD EDGE--------
Masukkan Asal Gedung      : 1
Masukkan Tujuan Gedung    : 1
Masukkan Jarak Gedung     : 1
============PILIH============
1. Add edge
2. Remove Edge
3. Degree
4. Print Graph
5. Cek Edge
6. Update Jarak
7. Hitung Edge
8.Exit
Masukkan pilihan: 1
=============================
 --------ADD EDGE--------
Masukkan Asal Gedung      : 2
Masukkan Tujuan Gedung    : 2
Masukkan Jarak Gedung     : 2
```

CEK EDGE

```
============PILIH============
1. Add edge
2. Remove Edge
3. Degree
4. Print Graph
5. Cek Edge
6. Update Jarak
7. Hitung Edge
8.Exit
Masukkan pilihan: 5
=============================
 --------- CEK EDGE ---------
Masukkan gedung asal: 2
Masukkan gedung tujuan: 2
Gedung C dan C bertetangga
============PILIH============
1. Add edge
2. Remove Edge
3. Degree
4. Print Graph
5. Cek Edge
6. Update Jarak
7. Hitung Edge
8.Exit
Masukkan pilihan: 5
=============================
 --------- CEK EDGE ---------
Masukkan gedung asal: 1
Masukkan gedung tujuan: 2
Gedung B dan C bertetangga
```

DEGREE

```
============PILIH===========
1. Add edge
2. Remove Edge
3. Degree
4. Print Graph
5. Cek Edge
6. Update Jarak
7. Hitung Edge
8.Exit
Masukkan pilihan: 3
==============================
 --------- DEGREE ---------
Masukkan Asal Gedung      : 3
inDegree dari gedungD: 1
outDegree dari gedungD: 1
Degree dari gedungD: 11
```

REMOVE

```
============PILIH===========
1. Add edge
2. Remove Edge
3. Degree
4. Print Graph
5. Cek Edge
6. Update Jarak
7. Hitung Edge
8.Exit
Masukkan pilihan: 2
==============================
 --------REMOVE EDGE--------
Masukkan Asal Gedung      : 1
Masukkan Tujuan Gedung    : 1
============PILIH===========
1. Add edge
2. Remove Edge
3. Degree
4. Print Graph
5. Cek Edge
6. Update Jarak
7. Hitung Edge
8.Exit
Masukkan pilihan: 4
==============================
gedung B terhubung dengan
C (5 m),
gedung C terhubung dengan
C (2 m),
gedung D terhubung dengan
D (3 m),
gedung E terhubung dengan
E (4 m),
gedung F terhubung dengan
F (5 m),
```

HITUNG EDGE

```
============PILIH===========
1. Add edge
2. Remove Edge
3. Degree
4. Print Graph
5. Cek Edge
6. Update Jarak
7. Hitung Edge
8.Exit
Masukkan pilihan: 7
============================
 --------- HITUNG EDGE ---------
Total edge dalam graf: 5
============PILIH===========
```

UPDATE JARAK

```
 --------- UPDATE JARAK ---------
Masukkan gedung asal: 1
Masukkan gedung tujuan: 2
Masukkan jarak baru: 3
============PILIH===========
1. Add edge
2. Remove Edge
3. Degree
4. Print Graph
5. Cek Edge
6. Update Jarak
7. Hitung Edge
8.Exit
Masukkan pilihan: 4
============================
gedung B terhubung dengan
C (3 m),
gedung C terhubung dengan
C (2 m),
gedung D terhubung dengan
D (3 m),
gedung E terhubung dengan
E (4 m),
gedung F terhubung dengan
F (5 m),
```