Laporan Praktikum Algoritma & Struktur Data

Jobsheet 15 – Graph

Nama: Aqil Rahmat Alifiandi

NIM: 2341760099

Prodi: D-IV Sistem Informasi Bisnis

JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG
2023/2024

PERCOBAAN 1

Buka text editor. Buat class Node.java dan class DoubleLinkedList.java sesuai dengan praktikum Double Linked List.

A. Class Node

Kode program yang terdapat pada class Node belum dapat mengakomodasi kebutuhan pembuatan graf berbobot, sehingga diperlukan sedikit modifikasi. Setelah Anda menyalin kode program dari class Node pada praktikum Double Linked List, tambahkan atribut jarak bertipe int untuk menyimpan bobot graf

```
public class Node05 {

    int data;
    Node05 prev, next;
    int jarak;

    Node05(Node05 prev, int data, int jarak, Node05 next) {
        this.prev = prev;
        this.data = data;
        this.jarak = jarak;
        this.next = next;
    }
}
```

B. Class DoubleLinkedList

Setelah Anda menyalin kode program dari class DoubleLinkedList pada praktikum Double Linked List, lakukan modifikasi pada method addFirst agar dapat menerima parameter jarak dan digunakan saat instansiasi Node

```
public void addFirst (int item, int jarak) {
    if (isEmpty()) {
        head = new Node05(prev:null, item, jarak,  next:null);
    } else {
        Node05 newNode05 = new Node05(prev:null, item, jarak, head);
        head.prev = newNode05;
        head = newNode05;
    }
    size++;
}
```

Selanjutnya buat method getJarak (hampir sama seperti method get) yang digunakan untuk mendapatkan nilai jarak edge antara dua node.

```
public int getJarak(int index) throws Exception {
    if (isEmpty() || index >= size) {
        throw new Exception(message:"Nilai indeks diluar batas.");
    }
    Node05 tmp = head;
    for (int i = 0; i < index; i++) {
        tmp = tmp.next;
    }
    return tmp.jarak;
}
```

Modifikasi method remove agar dapat melakukan penghapusan edge sesuai dengan node asal dan tujuan pada graf. Pada praktikum Double Linked List, parameter index digunakan untuk menghapus data sesuai posisi pada indeks tertentu, sedangkan pada Graf ini, penghapusan didasarkan pada data node tujuan, sehingga modifikasi kode diperlukan untuk menghindari index out of bound.

```
public void remove(int index) {
    Node05 current = head;
    while (current != null) {
        if (current.data == index) {
            if (current.prev != null) {
                current.prev.next = current.next;
            }else {
                head = current.next;
            }
            if (current.next != null) {
                current.next.prev = current.prev;
            }
            size--;
            break;
        }
        current = current.next;
    }
}
```

C. Class Graph

Lengkapi class Graph dengan atribut yang telah digambarkan di dalam pada class diagram, yang terdiri dari atribut vertex dan DoubleLinkedList dan Tambahkan konstruktor default untuk menginisialisasi variabel vertex dan menambahkan perulangan jumlah vertex sesuai dengan panjang array yang telah ditentukan.

```
public class Graph05 {

    int vertex;
    DoubleLinkedLists05[] list;

    public Graph05(int v) {
        vertex = v;
        list = new DoubleLinkedLists05[v];
        for (int i = 0; i < v; i++) {
            list[i] = new DoubleLinkedLists05();
        }
    }
```

Tambahkan method addEdge() untuk menghubungkan dua node. Baris kode program berikut digunakan untuk graf berarah (directed).

```
public void addEdge(int asal, int tujuan, int jarak) {
    list[asal].addFirst(tujuan, jarak);

}
```

Tambahkan method degree() untuk menampilkan jumlah derajat lintasan pada setiap vertex. Kode program berikut digunakan untuk menghitung degree pada graf berarah

```
public void degree(int asal) throws Exception {
    int k, totalIn = 0, totalOut = 0;
    for (int i = 0; i < vertex; i++) {

        for (int j = 0; j < list[i].size(); j++) {
            if (list[i].get(j) == asal) {
                totalIn++;
            }
        }

        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));

}
```

Tambahkan method removeEdge() untuk menghapus lintasan pada suatu graph. Penghapusan membutuhkan 2 parameter yaitu node asal dan tujuan.

```
public void removeEdge(int asal, int tujuan) throws Exception {
    for(int i = 0; i < vertex; i++) {
        if (i == tujuan) {
            list[asal].remove(tujuan);
        }
    }
}
```

Tambahkan method removeAllEdges() untuk menghapus semua vertex yang ada di dalam graf.

```
public void removeAllEdges() {
    for (int i = 0; i < vertex; i++) {
        list[i].clear();
    }
    System.out.println(x:"Graf berhasil dikosongkan");
}
```

Tambahkan method printGraph() untuk mencetak graf.

```
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();
}
```

D. Class Utama

Di dalam fungsi main, lakukan instansiasi object Graph bernama gedung dengan nilai parameternya adalah 6.

```
public static void main(String[] args) throws Exception {
    Graph05 gedung = new Graph05(v:6);
```

Tambahkan beberapa edge pada graf, tampilkan degree salah satu node, kemudian tampilkan grafnya.

```
gedung.addEdge(asal:0, tujuan:1, jarak:50);
gedung.addEdge(asal:0, tujuan:2, jarak:100);
gedung.addEdge(asal:1, tujuan:3, jarak:70);
gedung.addEdge(asal:2, tujuan:3, jarak:40);
gedung.addEdge(asal:3, tujuan:4, jarak:60);
gedung.addEdge(asal:4, tujuan:5, jarak:80);
gedung.degree(asal:0);
gedung.printGraph();
```

Hasil Run:

```
InDegree dari Gedung A: 0
OutDegree dari Gedung A: 2
Degree dari Gedung A: 2
Gedung A terhubung dengan:
C (100 m), B (50 m),
Gedung B terhubung dengan:
D (70 m),
Gedung C terhubung dengan:
D (40 m),
Gedung D terhubung dengan:
E (60 m),
Gedung E terhubung dengan:
F (80 m),
```

Tambahkan pemanggilan method removeEdge(), kemudian tampilkan kembali graf tersebut.

```
gedung.removeEdge(asal:1, tujuan:3);
gedung.printGraph();
```

Hasil Run:

```
Gedung A terhubung dengan:
C (100 m), B (50 m),
Gedung C terhubung dengan:
D (40 m),
Gedung D terhubung dengan:
E (60 m),
Gedung E terhubung dengan:
F (80 m),
```

PERTANYAAN

1. Perbaiki kode program Anda apabila terdapat error atau hasil kompilasi kode tidak sesuai!
   Jawaban: Terjadi error pada class DoubleLinkedList dan tepatnya pada method remove, perbaikan dengan menambahkan "size—"
   ```
   size--;
   ```
2. Pada class Graph, terdapat atribut list[] bertipe DoubleLinkedList. Sebutkan tujuan pembuatan variabel tersebut!
   Jawaban: untuk merepresentasikan graf sebagai kumpulan dari beberapa list.
3. Jelaskan alur kerja dari method removeEdge!
   Jawaban:
   - Metode removeEdge menerima dua parameter, asal dan tujuan, yang mewakili dua simpul (vertex) dalam graf.
   - Looping melalui semua simpul di graf menggunakan indeks i dari 0 hingga jumlah vertex (vertex).
   - Jika indeks i sama dengan tujuan, maka metode akan mencoba menghapus simpul tujuan dari daftar simpul yang terhubung dengan simpul asal.
4. Apakah alasan pemanggilan method addFirst() untuk menambahkan data, bukan method add jenis lain saat digunakan pada method addEdge pada class Graph?
   Jawaban: Pemanggilan addFirst dalam metode addEdge dipilih karena alasan efisiensi, kemudahan implementasi, dan konsistensi dalam pengelolaan adjacency list. Dengan addFirst, penambahan edge dilakukan dalam waktu konstan, menjaga performa graf tetap optimal
   tanpa perlu traversal yang tidak perlu atau penanganan urutan khusus.
5. Modifikasi kode program sehingga dapat dilakukan pengecekan apakah terdapat jalur antara suatu node dengan node lainnya, seperti contoh berikut (Anda dapat memanfaatkan Scanner)

```
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
```

Jawaban:

Menambahkan kode pada class Graph01 yaitu method ifTrue bertipe boolean untuk pengecekan apakah terdapat jalur antara uatu node dengan node lainnya.

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

Menambahkan kode pada class GraphMain01 berikut ini yaitu sacnner berupa inputan variabel asal dan tujuan yang berguna agar pengguna dapat memasukkan data

```
    System.out.print(s:"Masukkan inputan: ");
    int input = sc05.nextInt();

    for (int i = 0; i < input; i++) {
        System.out.print(s:"Masukkan gedung asal: ");
        asal = sc05.nextInt();
        System.out.print(s:"Masukkan gedung tujuan: ");
        tujuan = sc05.nextInt();
        if (gedung.ifTrue(asal, tujuan)) {
            System.out.println("Gedung " + (char) ('A' + asal) + " dan " + (char) ('A' + tujuan) + " bertetanga");
    }else {
        System.out.println("Gedung " + (char) ('A' + asal) + " dan " + (char) ('A' + tujuan) + " tidak bertetangga");
    }
    System.out.println();
}
sc05.close();
}
```

Hasil Run:

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

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

PERCOBAAN 2

Buat file baru, beri nama GraphMatriks05.java. Lengkapi class GraphMatriks dengan atribut vertex dan matriks.

```
public class GraphMatriks05 {
    int vertex;
    int[][] matriks;
```

Tambahkan konstruktor default untuk menginisialisasi variabel vertex dan menginstansiasi panjang array dua dimensi yang telah ditentukan.

```
public GraphMatriks05(int v) {
    vertex = v;
    matriks = new int[v][v];
}
```

Untuk membuat suatu lintasan yang menghubungkan dua node, maka dibuat method makeEdge() sebagai berikut.

```
public void makeEdge(int asal, int tujuan, int jarak) {
    matriks[asal][tujuan] = jarak;
}
```

Tambahkan method removeEdge() untuk menghapus lintasan pada suatu graf.

```
public void removeEdge(int asal, int tujuan) {
    matriks[asal][tujuan] = -1;
}
```

Tambahkan method printGraph() untuk mencetak graf.

```
public void printGraph() {
    for (int i = 0; i < vertex; i++) {
        System.out.print("Gedung " + (char) ('A' + i) + ": ");
        for (int j = 0; j < vertex; j++) {
            if (matriks[i][j] != -1) {
                System.out.print("Gedung " + (char) ('A' + j) + " (" + matriks[i][j] + " m), ");
            }
        }
        System.out.println();
    }
}
```

Tambahkan kode berikut pada file GraphMain<NoAbsen>.java yang sudah dibuat pada Percobaan 1.

```
public class GraphMain05 {

    Run | Debug
    public static void main(String[] args) throws Exception {

        GraphMatriks05 gdg = new GraphMatriks05(v:4);

        gdg.makeEdge(asal:0, tujuan:1, jarak:50);
        gdg.makeEdge(asal:1, tujuan:0, jarak:60);
        gdg.makeEdge(asal:1, tujuan:2, jarak:70);
        gdg.makeEdge(asal:2, tujuan:1, jarak:80);
        gdg.makeEdge(asal:2, tujuan:3, jarak:40);
        gdg.makeEdge(asal:3, tujuan:0, jarak:90);
        gdg.printGraph();

        System.out.println(x:"Hasil setengah penghapusan edge");
        gdg.removeEdge(asal:2, tujuan:1);
        gdg.printGraph();
```

Hasil Run:

```
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 setengah 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),
```

PERTANYAAN

1. Perbaiki kode program Anda apabila terdapat error atau hasil kompilasi kode tidak sesuai!
   Jawaban: berikut adalah yang salah:

```
Gedung C: Gedung A (0 m), Gedung C (0 m), Gedung D (40 m),
```

   Pembenaran:
   Dari -1 diubah menjadi -0

```
public void removeEdge(int asal, int tujuan) {
    matriks[asal][tujuan] = -0;
```

   Hasil Run yg benar:

```
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 setengah 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 B (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),
```

2. Apa jenis graph yang digunakan pada Percobaan 2?
   Jawaban: yang digunakan adalah graph berarah dan graph berbobot.
3. Apa maksud dari dua baris kode berikut?

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

   Jawaban: Maksud daripada baris pertama yaitu untuk menambahkan sebuah edge dari vertex 1 ke vertex 2 dengan bobot (jarak) 70. Ini berarti ada sebuah koneksi dari node 1 menuju node 2 dengan jarak atau bobot sebesar 70.
   Sedangkan pada baris kedua di maksudkan untuk menambahkan sebuah edge dari vertex 2 ke vertex 1 dengan bobot (jarak) 80. Ini berarti ada sebuah koneksi dari node 2 menuju node 1 dengan jarak atau bobot sebesar 80.
4. Modifikasi kode program sehingga terdapat method untuk menghitung degree, termasuk inDegree dan outDegree!
   Jawaban:

   Class GraphMatriks

```
public int outDegree(int vertex) {
    int degree = 0;
    for (int i = 0; i < this.vertex; i++) {
        if (matriks[vertex][i] != 0) {
            degree++;
        }
    }
    return degree;
}

public int inDegree(int vertex) {
    int degree = 0;
    for (int i = 0; i < this.vertex; i++) {
        if (matriks[i][vertex] != 0) {
            degree++;
        }
    }
    return degree;
}
```

   Class Main

```
for (int i = 0; i < 4; i++) {
    System.out.println("Gedung " + (char) ('A' + i) + ":");
    System.out.println("  inDegree: " + gdg.inDegree(i));
    System.out.println("  outDegree: " + gdg.outDegree(i));
}
```

Hasil Run:

```
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 setengah 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 B (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),
Gedung A:
  inDegree: 2
  outDegree: 1
Gedung B:
  inDegree: 1
  outDegree: 2
Gedung C:
  inDegree: 1
  outDegree: 1
Gedung D:
  inDegree: 1
  outDegree: 1
```

Latihan Praktikum

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!

Jawaban:

ClassGraphMain:

```java
import java.util.Scanner;

public class GraphMain05 {
    Run | Debug
    public static void main(String[] args) throws Exception {
        Scanner sc11 = new Scanner(System.in);
        GraphMatriks05 gdg = new GraphMatriks05(v:4);
        gdg.makeEdge(asal:0, tujuan:1, jarak:50);
        gdg.makeEdge(asal:1, tujuan:0, jarak:60);
        gdg.makeEdge(asal:1, tujuan:2, jarak:70);
        gdg.makeEdge(asal:2, tujuan:1, jarak:80);
        gdg.makeEdge(asal:2, tujuan:3, jarak:40);
        gdg.makeEdge(asal:3, tujuan:0, jarak:90);
        gdg.printGraph();
        System.out.println(x:"Hasil setelah penghapusan edge");
        gdg.removeEdge(asal:2, tujuan:1);
        gdg.printGraph();

        for (int i = 0; i < 4; i++) {
        System.out.println("Gedung " + (char) ('A' + i) + ":");
        System.out.println("  inDegree: " + gdg.inDegree(i));
        System.out.println("  outDegree: " + gdg.outDegree(i));
    }

        while (true) {
            System.out.println(x:"Menu:");
            System.out.println(x:"1. Add Edge");
            System.out.println(x:"2. Remove Edge");
            System.out.println(x:"3. Degree");
            System.out.println(x:"4. Print Graph");
            System.out.println(x:"5. Cek Edge");
            System.out.println(x:"6. Update Jarak");
            System.out.println(x:"7. Hitung Edge");
            System.out.println(x:"8. Exit");
            System.out.print(s:"Pilih menu: ");
            int menu = sc11.nextInt();

            switch (menu) {
                case 1:
                    System.out.print(s:"Masukkan asal: ");
                    int asal = sc11.nextInt();
                    System.out.print(s:"Masukkan tujuan: ");
                    int tujuan = sc11.nextInt();
                    System.out.print(s:"Masukkan jarak: ");
                    int jarak = sc11.nextInt();
                    gdg.makeEdge(asal, tujuan, jarak);
                    break;
                case 2:
                    System.out.print(s:"Masukkan asal: ");
                    asal = sc11.nextInt();
                    System.out.print(s:"Masukkan tujuan: ");
                    tujuan = sc11.nextInt();
                    gdg.removeEdge(asal, tujuan);
                    break;
                case 3:
                    for (int i = 0; i < 4; i++) {
                        System.out.println("Gedung " + (char) ('A' + i) + ":");
                        System.out.println("  inDegree: " + gdg.inDegree(i));
                        System.out.println("  outDegree: " + gdg.outDegree(i));
                    }
                    break;
                case 4:
                    gdg.printGraph();
                    break;
                case 5:
                    System.out.print(s:"Masukkan asal: ");
                    asal = sc11.nextInt();
                    System.out.print(s:"Masukkan tujuan: ");
                    tujuan = sc11.nextInt();
                    if (gdg.hasEdge(asal, tujuan)) {
                        System.out.println(x:"Edge ada.");
                    } else {
                        System.out.println(x:"Edge tidak ada.");
                    }
                    break;
                case 6:
                    System.out.print(s:"Masukkan asal: ");
                    asal= sc11.nextInt();
                    System.out.print(s:"Masukkan tujuan: ");
                    tujuan= sc11.nextInt();
                    System.out.print(s:"Masukkan jarak baru: ");
                    jarak = sc11.nextInt();
                    gdg.updateJarak(asal, tujuan, jarak);
                    break;
                case 7:
                    System.out.println("Jumlah edge dalam graf: " + gdg.hitungEdge());
                    break;
                case 8:
                    sc11.close();
                    System.exit(status:0);
                default:
                    System.out.println(x:"Pilihan tidak valid!");
            }
        }
    }
}
```

ClassGraphMatriks:

```java
public class GraphMatriks05 {

    int vertex;
    int[][] matriks;

    public GraphMatriks05(int v) {
        vertex = v;
        matriks = new int[v][v];
    }

    public void makeEdge(int asal, int tujuan, int jarak) {
        matriks[asal][tujuan] = jarak;
    }

    public void removeEdge(int asal, int tujuan) {
        matriks[asal][tujuan] = 0;
    }

    public bool int[][] matriks al, int tujuan) {
        return matriks[asal][tujuan] != 0;
    }

    public void updateJarak(int asal, int tujuan, int jarak) {
        if (matriks[asal][tujuan] != 0) {
            matriks[asal][tujuan] = jarak;
        } else {
            System.out.println(x:"edge tidak ditemukan.");
        }
    }

    public int hitungEdge() {
        int jumlah = 0;
        for (int i = 0; i < vertex; i++) {
            for (int j = 0; j < vertex; j++) {
                if (matriks[i][j] != 0) {
                    jumlah++;
                }
            }
        }
        return jumlah;
    }

    public void printGraph() {
        for (int i = 0; i < vertex; i++) {
            System.out.print("Gedung " + (char) ('A' + i) + ": ");
            for (int j = 0; j < vertex; j++) {
                if (matriks[i][j] != -1) {
                    System.out.print("Gedung " + (char) ('A' + j) + " (" + matriks[i][j] + " m), ");
                }
            }
            System.out.println();
        }
    }

    public int outDegree(int vertex) {
        int degree = 0;
        for (int i = 0; i < this.vertex; i++) {
            if (matriks[vertex][i] != 0) {
                degree++;
            }
        }
        return degree;
    }

    public int inDegree(int vertex) {
        int degree = 0;
        for (int i = 0; i < this.vertex; i++) {
            if (matriks[i][vertex] != 0) {
                degree++;
            }
        }
        return degree;
    }
}
```

Hasil Run:

```
Menu:
1. Add Edge
2. Remove Edge
3. Degree
4. Print Graph
5. Cek Edge
6. Update Jarak
7. Hitung Edge
8. Exit
Pilih menu: 1
Masukkan asal: 0
Masukkan tujuan: 2
Masukkan jarak: 50
Menu:
1. Add Edge
2. Remove Edge
3. Degree
4. Print Graph
5. Cek Edge
6. Update Jarak
7. Hitung Edge
8. Exit
Pilih menu: 4
Gedung A: Gedung A (0 m), Gedung B (50 m), Gedung C (50 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 (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),
```

```
Menu:
1. Add Edge
2. Remove Edge
3. Degree
4. Print Graph
5. Cek Edge
6. Update Jarak
7. Hitung Edge
8. Exit
Pilih menu: 2
Masukkan asal: 2
Masukkan tujuan: 2
```

```
Pilih menu: 3
Gedung A:
  inDegree: 2
  outDegree: 2
Gedung B:
  inDegree: 1
  outDegree: 2
Gedung C:
  inDegree: 2
  outDegree: 1
Gedung D:
  inDegree: 1
  outDegree: 1
```