Tugas Praktikum Mata Kuliah

Alogaritma dan Struktur Data

Jobsheet 15 : Graph

Dosen Pembimbing : Triana Fatmawati, S.T,M.T

Eka Putri Natalya Kabelen

2341760107

SIB 1E

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

Tahun Ajaran 2024

2. Praktikum

2.1 Percobaan 1: Implementasi Graph menggunakan Linked List

2.1.1 Langkah-langkah Percobaan

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

A. Class Node

Node11.java > Node11 > Node11(Node11, int, int, Node11)

```
1   import org.w3c.dom.Node;
2   public class Node11 {
3
4   int data;
5   Node11 prev, next;
6   int jarak;
7
8   Node11(Node11 prev, int data, int jarak, Node11 next) {
9       this.prev = prev;
10      this.data = data;
11      this.next = next;
12      this.jarak = jarak;
13  }
14  }
```

B. Class DoubleLinkedList

```java
public class DoubleLinkedLists11 {

    Node11 head;
    int size;

    public void DoubleLinkedList11(){
        head = null;
        size = 0;
    }

    public boolean isEmpty(){
        return head == null;
    }

    public void addFirst(int item, int jarak){
        if(isEmpty()){
            head = new Node11(null, item, jarak, null);
        }else{
            Node11 newNode = new Node11(null, item, jarak, head);
            head.prev = newNode;
            head = newNode;
        }
        size++;
    }

    public int getJarak(int index) throws Exception{
        if (isEmpty() || index >= size) {
            throw new Exception("Nilai indeks diluar batas.");
        }
        Node11 tmp = head;
        for (int i = 0; i < index; i++) {
            tmp = tmp.next;
        }
        return tmp.jarak;
    }

    public void remove(int index) {
        Node11 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;
        }
    }

    public int size() {
        return size;
    }

    public int get(int index) throws Exception {
        if (isEmpty() || index >= size) {
            throw new Exception("Nilai indeks di luar batas");
        }
        Node11 tmp = head;
        for (int i = 0; i < index; i++) {
            tmp = tmp.next;
        }
        return tmp.data;
    }

    public void clear(){
        head = null;
        size = 0;
    }
}
```

C. Class Graph

2. Buat file baru, beri nama Graph.java Lengkapi class Graph dengan atribut yang telah digambarkan di dalam pada class diagram, yang terdiri dari atribut vertex dan DoubleLinkedList.

```
Graph11.java > Graph11
1  public class Graph11 {
2
3      int vertex;
4      DoubleLinkedLists11 list[];
5
```

3. Tambahkan konstruktor default untuk menginisialisasi variabel vertex dan menambahkan perulangan jumlah vertex sesuai dengan panjang array yang telah ditentukan.

```
6      public Graph11(int v) {
7          vertex = v;
8          list = new DoubleLinkedLists11[v];
9          for (int i = 0; i < v; i++) {
10             list[i] = new DoubleLinkedLists11();
11         }
12     }
```

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

```
14     public void addEdge(int asal, int tujuan, int jarak) {
15         list[asal].addFirst(tujuan, jarak);
16         // list[tujuan].addFirst(tujuan, jarak);
17     }
```

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

```
19  public void degree(int asal) throws Exception {
20      int k, totalIn = 0, totalOut = 0;
21      for(int i = 0; i < vertex; i++) {
22          //inDegree
23          for(int j = 0; j < list[i].size(); j++) {
24              if (list[i].get(j) == asal) {
25                  ++totalIn;
26              }
27          }
28          //outDegree
29          for(k = 0; k < list[asal].size(); k++) {
30              list[asal].get(k);
31          }
32          totalOut = k;
33      }
34      System.out.println("InDegree dari Gedung " + (char) ('A' + asal) + ": " + totalIn);
35      System.out.println("OutDegree dari Gedung " + (char) ('A' + asal) + ": " + totalOut);
36      System.out.println("Degree dari Gedung " + (char) ('A' + asal) + ": " + (totalIn + totalOut));
37      // System.out.println("Degree dari Gedung " + (char) ('A' + asal) + ": " + list[asal].size());
38  }
```

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

```
40      public void removeEdge(int asal, int tujuan) throws Exception {
41          for(int i = 0; i < vertex; i++) {
42              if (i == tujuan) {
43                  list[asal].remove(tujuan);
44              }
45          }
46      }
```

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

```
48      public void removeAllEdges() {
49          for ( int i = 0; i < vertex; i++) {
50              list[i].clear();
51          }
52          System.out.println(x:"Graf berhasil dikosongkan");
53      }
```

8. Tambahkan method printGraph() untuk mencetak graf.

```
55      public void printGraph() throws Exception {
56          for(int i = 0; i < vertex; i++) {
57              if (list[i].size() > 0) {
58                  System.out.println("Gedung " + (char) ('A' + i) + " terhubung dengan ");
59                  for(int j = 0; j < list[i].size(); j++) {
60                      System.out.print((char) ('A' + list[i].get(j)) + " (" + list[i].getJarak(j) + " m), ");
61                  }
62                  System.out.println(x:"");
63              }
64          }
65          System.out.println(x:"");
66      }
67  }
```

D. Class Utama

9. Buat file baru, beri nama GraphMain.java
10. Tuliskan struktur dasar bahasa pemrograman Java yang terdiri dari fungsi main 1
11. Di dalam fungsi main, lakukan instansiasi object Graph bernama gedung dengan nilai parameternya adalah 6.
12. Tambahkan beberapa edge pada graf, tampilkan degree salah satu node, kemudian tampilkan grafnya.

```
GraphMain11.java > ...
1   import java.util.Scanner;
2   public class GraphMain11 {
3
        Run | Debug
4       public static void main(String[] args) throws Exception {
5           Graph11 gedung = new Graph11(v:6);
6           gedung.addEdge(asal:0, tujuan:1, jarak:50);
7           gedung.addEdge(asal:0, tujuan:2, jarak:100);
8           gedung.addEdge(asal:1, tujuan:3, jarak:70);
9           gedung.addEdge(asal:2, tujuan:3, jarak:40);
10          gedung.addEdge(asal:3, tujuan:4, jarak:60);
11          gedung.addEdge(asal:4, tujuan:5, jarak:80);
12          gedung.degree(asal:0);
13          gedung.printGraph();
```

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

```
15          gedung.removeEdge(asal:1,tujuan:3);
16          gedung.printGraph();
17      }
18  }
```

14. Hasil run code program

```
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),

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),
```

2.1.3 Pertanyaan

1. Perbaiki kode program Anda apabila terdapat error atau hasil kompilasi kode tidak sesuai!
   Jawab :
   Terjadi error di saat running percobaan 1 pada bagaian exception dan untuk mengatasi masalah ini saya membenarkan kode pada class DoubleLinkedList method Remove() lalu menambahkan size--.

2. Pada class Graph, terdapat atribut list[] bertipe DoubleLinkedList. Sebutkan tujuan pembuatan variabel tersebut!
   Jawab :
   Pada class Graph, atribut list[] yang bertipe DoubleLinkedList digunakan untuk merepresentasikan graf sebagai kumpulan dari beberapa list.

3. Jelaskan alur kerja dari method removeEdge!
   Jawab :
   Method removeEdge dalam kelas Graph berfungsi untuk menghapus sebuah edge antara dua simpul dalam graf. Alur kerja method ini melibatkan penghapusan entri yang sesuai dari daftar adjacency (linked list) untuk simpul asal (asal). Metode removeEdge bertujuan untuk menghapus sebuah edge (sisi) yang menghubungkan dua node dalam graf.

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

   Jawab :

   Class Graph

```
67      public boolean ifTrue(int asal, int tujuan) throws Exception {
68          for(int i = 0; i < list[asal].size(); i++) {
69              if (list[asal].get(i) == tujuan) {
70                  return true;
71              }
72          }
73          return false;
74      }
75  }
```

   Class GraphMain

```
21          Scanner sc11 = new Scanner(System.in);
22          int asal, tujuan;
23
24          System.out.print(s:"Masukkan inputan: ");
25          int input = sc11.nextInt();
26
27          for (int i = 0; i < input; i++) {
28              System.out.print(s:"Masukkan gedung asal: ");
29              asal = sc11.nextInt();
30              System.out.print(s:"Masukkan gedung tujuan: ");
31              tujuan = sc11.nextInt();
32              if (gedung.ifTrue(asal, tujuan)) {
33                  System.out.println("Gedung " + (char) ('A' + asal) + " dan " + (char) ('A' + tujuan) + " bertetangga");
34              } else {
35                  System.out.println("Gedung " + (char) ('A' + asal) + " dan " + (char) ('A' + tujuan) + " tidak bertetangga");
36              }
37              System.out.println();
38          }
39          sc11.close();
40
41      }
42  }
```

Hasil run code modifikasi

```
Masukkan inputan: 2
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
```

2.2 Percobaan 2: Implementasi Graph menggunakan Matriks

Langkah-langkah Percobaan

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

```
1  public class GraphMatriks11 {
2
3      int vertex;
4      int[][] matriks;
```

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

```
6      public GraphMatriks11(int v) {
7          vertex = v;
8          matriks = new int[v][v];
9      }
```

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

```
11     public void makeEdge(int asal, int tujuan, int jarak) {
12         matriks[asal][tujuan] = jarak;
13     }
```

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

```
15     public void removeEdge(int asal, int tujuan) {
16         matriks[asal][tujuan] = 0;
17     }
```

6. Tambahkan method printGraph() untuk mencetak graf.

```
43      public void printGraph() {
44          for (int i = 0; i < vertex; i++) {
45              System.out.print("Gedung " + (char) ('A' + i) + ": ");
46              for (int j = 0; j < vertex; j++) {
47                  if (matriks[i][j] != -1) {
48                      System.out.print("Gedung " + (char) ('A' + j) + " (" + matriks[i][j] + " m), ");
49                  }
50              }
51              System.out.println();
52          }
53      }
```

7. Tambahkan kode berikut pada file GraphMain.java yang sudah dibuat pada Percobaan 1.

```
6           GraphMatriks11 gdg = new GraphMatriks11(v:4);
7           gdg.makeEdge(asal:0, tujuan:1, jarak:50);
8           gdg.makeEdge(asal:1, tujuan:0, jarak:60);
9           gdg.makeEdge(asal:1, tujuan:2, jarak:70);
10          gdg.makeEdge(asal:2, tujuan:1, jarak:80);
11          gdg.makeEdge(asal:2, tujuan:3, jarak:40);
12          gdg.makeEdge(asal:3, tujuan:0, jarak:90);
13          gdg.printGraph();
14          System.out.println(x:"Hasil setelah penghapusan edge");
15          gdg.removeEdge(asal:2, tujuan:1);
16          gdg.printGraph();
```

8. 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 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),
PS C:\Users\ASUS\Documents\SEMESTER 2\Alogaritma&StrukturData\JOBSHEET 15>
```

2.2.3 Pertanyaan

1. Perbaiki kode program Anda apabila terdapat error atau hasil kompilasi kode tidak sesuai!
   Jawab :
   Tidak terjadi error didalam kode program ini

2. Apa jenis graph yang digunakan pada Percobaan 2?
   Jawab :
   Jenis graph yang digunakan yaitu graph berarah dan graph berbobot. Ini terlihat di cara pengaturan nilai pada matriks adjacency dalam metode makeEdge, di mana nilai jarak (yang merupakan bobot) disimpan pada posisi matriks[asal][tujuan]. Sedangkan dalam metode removeEdge untuk mengindikasikan penghapusan edge juga mendukung bahwa graph ini berarah, karena hanya arah dari asal ke tujuan yang dihapus, bukan sebaliknya.

3. Apa maksud dari dua baris kode berikut?

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

Jawab :

Maksut dari dua bari tersbeut yaitu pada baris pertama 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!

Jawab :

Class GraphMatriks

```
55      public int outDegree(int vertex) {
56          int degree = 0;
57          for (int i = 0; i < this.vertex; i++) {
58              if (matriks[vertex][i] != 0) {
59                  degree++;
60              }
61          }
62          return degree;
63      }
64
65      public int inDegree(int vertex) {
66          int degree = 0;
67          for (int i = 0; i < this.vertex; i++) {
68              if (matriks[i][vertex] != 0) {
69                  degree++;
70              }
71          }
72          return degree;
73      }
74  }
```

Class Main

```
19          for (int i = 0; i < 4; i++) {
20          System.out.println("Gedung " + (char) ('A' + i) + ":");
21          System.out.println("  inDegree: " + gdg.inDegree(i));
22          System.out.println("  outDegree: " + gdg.outDegree(i));
23      }
24  }
25  }
```

Hasil run kode program

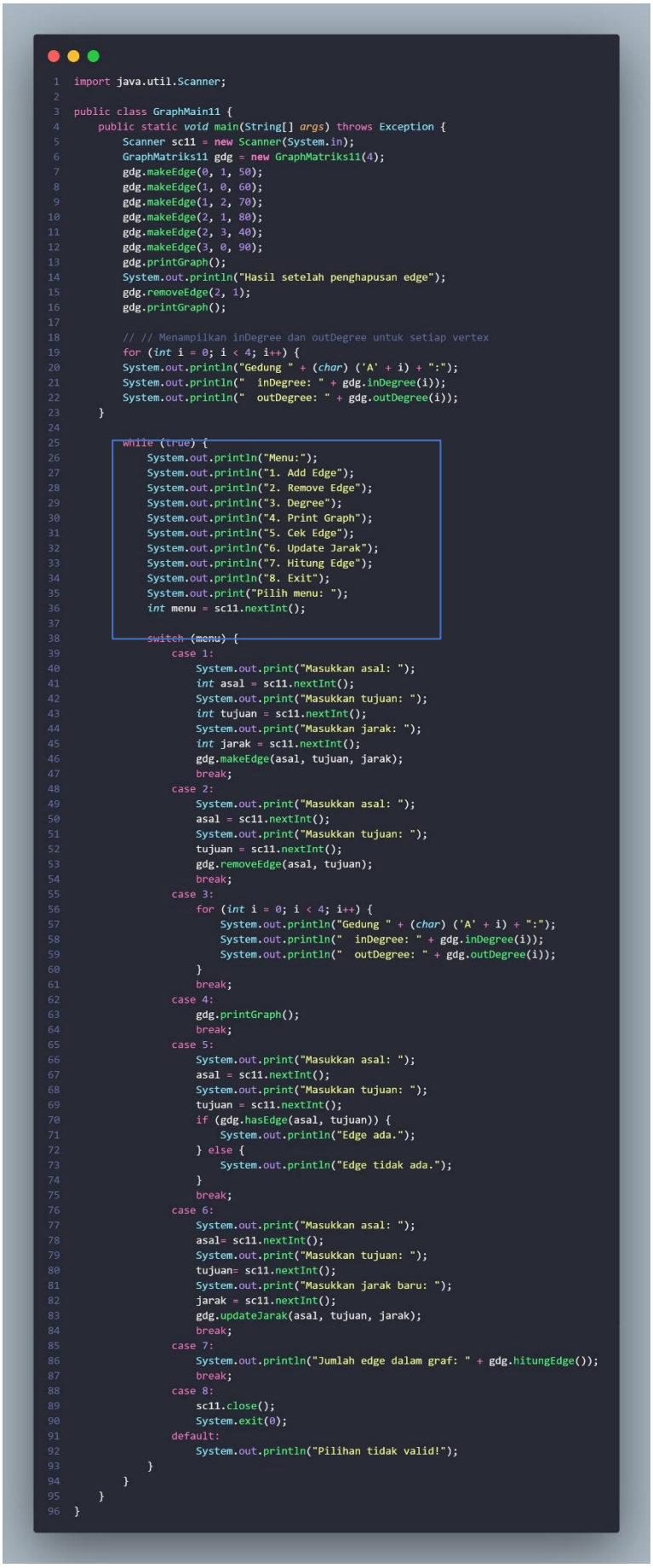
```
Gedung A: Gedung A (0 m), Gedung B (50 m), Gedung C (0 m), Gedung D (0 m),
Gedung B: Gedung A (60 m), Gedung B (0 m), Gedung C (70 m), Gedung D (0 m),
Gedung C: Gedung A (0 m), Gedung B (80 m), Gedung C (0 m), Gedung D (40 m),
Gedung D: Gedung A (90 m), Gedung B (0 m), Gedung C (0 m), Gedung D (0 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 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
PS C:\Users\ASUS\Documents\SEMESTER 2\Alogaritma&StrukturData\JOBSHEET 15>
```

3. 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!

JAWAB ;

ClassGraphMain



```java
import java.util.Scanner;

public class GraphMain11 {
    public static void main(String[] args) throws Exception {
        Scanner sc11 = new Scanner(System.in);
        GraphMatriks11 gdg = new GraphMatriks11(4);
        gdg.makeEdge(0, 1, 50);
        gdg.makeEdge(1, 0, 60);
        gdg.makeEdge(1, 2, 70);
        gdg.makeEdge(2, 1, 80);
        gdg.makeEdge(2, 3, 40);
        gdg.makeEdge(3, 0, 90);
        gdg.printGraph();
        System.out.println("Hasil setelah penghapusan edge");
        gdg.removeEdge(2, 1);
        gdg.printGraph();

        // // Menampilkan inDegree dan outDegree untuk setiap vertex
        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("Menu:");
            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("Pilih menu: ");
            int menu = sc11.nextInt();

            switch (menu) {
                case 1:
                    System.out.print("Masukkan asal: ");
                    int asal = sc11.nextInt();
                    System.out.print("Masukkan tujuan: ");
                    int tujuan = sc11.nextInt();
                    System.out.print("Masukkan jarak: ");
                    int jarak = sc11.nextInt();
                    gdg.makeEdge(asal, tujuan, jarak);
                    break;
                case 2:
                    System.out.print("Masukkan asal: ");
                    asal = sc11.nextInt();
                    System.out.print("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("Masukkan asal: ");
                    asal = sc11.nextInt();
                    System.out.print("Masukkan tujuan: ");
                    tujuan = sc11.nextInt();
                    if (gdg.hasEdge(asal, tujuan)) {
                        System.out.println("Edge ada.");
                    } else {
                        System.out.println("Edge tidak ada.");
                    }
                    break;
                case 6:
                    System.out.print("Masukkan asal: ");
                    asal= sc11.nextInt();
                    System.out.print("Masukkan tujuan: ");
                    tujuan= sc11.nextInt();
                    System.out.print("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(0);
                default:
                    System.out.println("Pilihan tidak valid!");
            }
        }
    }
}
```

ClassGraphMatriks

```java
public class GraphMatriks11 {

    int vertex;
    int[][] matriks;

    public GraphMatriks11(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 boolean hasEdge(int asal, 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("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 code program

```
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),
```

```
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
```