PRAKTIKUM ALGORITMA DAN STRUKTUR DATA

JOBSHEET 12

NAMA : DIMAS ADI BAYU SAMUDRA

KELAS : 1A

NO. ABSEN : 08

NIM : 2341720169

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

2024

Praktikum

Percobaan 1

Kode Node08

```
public class Node08 {
    int data;
    Node08 prev,next;
    int jarak;

    Node08 (Node08 prev, Node08 next, int data, int jarak){
        this.prev = prev;
        this.next = next;
        this.data = data;
        this.jarak = jarak;
    }
}
```

Kode DoubleLinkedList08

```
public class DoubleLinkedList08 {
    Node08 head;
    int size;

    public DoubleLinkedList08() {
        head = null;
        size = 0;
    }

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

    public void addFirst(int item, int jarak) {
        if (isEmpty()) {
            head = new Node08(null, null, item, jarak);
        } else {
            Node08 newNode08 = new Node08(null, head, item, jarak);
            head.prev = newNode08;
            head = newNode08;
        }
        size++;
    }

    public int getJarak(int index) throws Exception {
        if (isEmpty() || index >= size) {
            throw new Exception("Nilai indeks di luar batas");
        }
        Node08 tmp = head;
        for (int i = 0; i < index; i++) {
            tmp = tmp.next;
        }
        return tmp.jarak;
```

```
    }

    public int get(int index) throws Exception {
        if (isEmpty() || index >= size) {
            throw new Exception("Nilai indeks di luar batas");
        }
        Node08 tmp = head;
        for (int i = 0; i < index; i++) {
            tmp = tmp.next;
        }
        return tmp.data;
    }

    public void remove(int index) throws Exception {
        if (index < 0 || index >= size) {
            throw new Exception("Nilai indeks di luar batas");
        }
        Node08 current = head;
        for (int i = 0; i < index; i++) {
            current = current.next;
        }
        if (current.prev != null) {
            current.prev.next = current.next;
        } else {
            head = current.next;
        }
        if (current.next != null) {
            current.next.prev = current.prev;
        }
        size--;
    }

    public int size() {
        return size;
    }

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

Kode Graph08

```
public class Graph08 {
    int vertex;
    DoubleLinkedList08 list[];

    public Graph08(int v) {
        vertex = v;
        list = new DoubleLinkedList08[v];
        for (int i = 0; i < v; i++) {
            list[i] = new DoubleLinkedList08();
        }
```

```
    }

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

    public void degree(int asal) throws Exception {
        int totalIn = 0, totalOut = list[asal].size();

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

        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));
    }

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

    public void removeAllEdge() {
        for (int i = 0; i < vertex; i++) {
            list[i].clear();
        }
    }

    public void printGraph() throws Exception {
        for (int i = 0; i < vertex; i++) {
            if (list[i].size() > 0) {
                System.out.print("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("");
    }
}
```

Kode GraphMain08

```
public class GraphMain {
    public static void main(String[] args) throws Exception {
        Graph08 gedung = new Graph08(6);
        gedung.addEdge(0, 1, 50);
        gedung.addEdge(0, 2, 100);
        gedung.addEdge(1, 3, 70);
        gedung.addEdge(2, 3, 40);
        gedung.addEdge(3, 4, 60);
        gedung.addEdge(4, 5, 80);
        gedung.degree(0);
        gedung.printGraph();
        gedung.removeEdge(1,3);
        gedung.printGraph();
    }
}
```

Hasil Run langkah 14

```
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),
```

Hasil Run langkah 17

```
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),
```

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).

```
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

2. - Menyimpan tetangga setiap vertex

- efisiensi memori

- memudahkan saat ingin mengubah pada vertex

3. method removeEdge menerima parameter yaitu asal sebagai vertex asal dan tujuan sebagai vertex tujuan, lalu di looping ditentukan oleh nilai vertex dan pengecekan jika I = tujuan maka menghapus node yang memiliki nilai data tujuan dari linked list yang dihubungkan dengan asal .

4. Efesiensi waktu dan lebih sederhana dalam implementasi

5. Kode tambahan pada Graph08

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

Kode tambahan pada main

```
Scanner sc = new Scanner(System.in);
        System.out.print("Asal : ");
        int asal  = sc.nextInt();
        System.out.print("Tujuan : ");
        int tujuan = sc.nextInt();
        gedung.cekEdge(asal, tujuan);
```

Hasil run

```
Asal : 2
Tujuan : 3
Gedung C Dan D Tidak Bertetangga
```

Gedung C Dan D Bertetangga

Percobaan 2

Kode GraphMatrix08

```
public class GraphMatriks08 {
    int vertex;
    int[][] matriks;

    public GraphMatriks08(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 void printGraph(){
        for (int i = 0; i < vertex; i++) {
            System.out.print("Gedung " + (char) ('A' + i) + " : ");
            for (int j = 0; j < vertex; j++) {
                System.out.print("Gedung " + (char)('A' + j) + " (" +
matriks[i][j] + " m),");
            }
            System.out.println();
        }
    }
}
```

Kode MainGraphMatrix08

```
public class GraphMainMatrix08 {
    public static void main(String[] args) {
        GraphMatriks08 gedung = new GraphMatriks08(4);
        gedung.makeEdge(0, 1, 50);
```

```
        gedung.makeEdge(1, 0, 60);
        gedung.makeEdge(1, 2, 70);
        gedung.makeEdge(2, 1, 80);
        gedung.makeEdge(2, 3, 40);
        gedung.makeEdge(3, 0, 90);
        gedung.printGraph();
        System.out.println("Hasil setelah penghapusan edge");
        gedung.removeEdge(2, 1);
        gedung.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 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),
```

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. sudah

2. directed graph (graf berarah).

3. Baris pertama membuat sebuah edge dari vertex 0 ke vertex 1 dengan jarak 50.Baris kedua membuat sebuah edge dari vertex 1 ke vertex 0 dengan jarak 60.

4. Kode tambahan GraphMatrix08

```
public int outDegree(int v) {
        int outDegree = 0;
        for (int i = 0; i < vertex; i++) {
            if (matriks[v][i] != 0) {
                outDegree++;
```

```
            }
        }
        return outDegree;
    }

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

    public int degree(int v) {
        return inDegree(v) + outDegree(v);
    }
```

Kode tambahan main

```
        System.out.println("OutDegree of vertex 2: " + gedung.outDegree(2));
        System.out.println("InDegree of vertex 2: " + gedung.inDegree(2));
        System.out.println("Degree of vertex 2: " + gedung.degree(2));
```

Hasil Run

```
OutDegree of vertex 2: 1
InDegree of vertex 2: 1
Degree of vertex 2: 2
```

Tugas

Kode Graph08

```
public class Graph08 {
    int vertex;
    DoubleLinkedList08 list[];
    int edge;

    public Graph08(int v) {
        vertex = v;
        list = new DoubleLinkedList08[v];
        for (int i = 0; i < v; i++) {
            list[i] = new DoubleLinkedList08();
        }
    }
    public int edge(){
        return edge;
    }
    public void addEdge(int asal, int tujuan, int jarak) {
        list[asal].addFirst(tujuan, jarak);
    }
```

```
    public void degree(int asal) throws Exception {
        int totalIn = 0, totalOut = list[asal].size();

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

        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));
    }

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

    public void removeAllEdge() {
        for (int i = 0; i < vertex; i++) {
            list[i].clear();
        }
    }

    public void printGraph() throws Exception {
        for (int i = 0; i < vertex; i++) {
            if (list[i].size() > 0) {
                System.out.print("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 void cekEdge(int asal, int tujuan) throws Exception {
```

```
        boolean found = false;
        for (int i = 0; i < list[asal].size(); i++) {
            if (list[asal].get(i) == tujuan) {
                found = true;
                break;
            }
        }
        if (found) {
            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");
        }
    }

    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].head.jarak = jarak;
                return;
            }
        }
        throw new Exception("Edge tidak ditemukan.");
    }

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

Kode DoubleLinkedList08

```
public class DoubleLinkedList08 {
    Node08 head;
    int size;

    public DoubleLinkedList08() {
        head = null;
        size = 0;
    }

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

    public void addFirst(int item, int jarak) {
        if (isEmpty()) {
            head = new Node08(null, null, item, jarak);
```

```
        } else {
            Node08 newNode08 = new Node08(null, head, item, jarak);
            head.prev = newNode08;
            head = newNode08;
        }
        size++;
    }

    public int getJarak(int index) throws Exception {
        if (isEmpty() || index >= size) {
            throw new Exception("Nilai indeks di luar batas");
        }
        Node08 tmp = head;
        for (int i = 0; i < index; i++) {
            tmp = tmp.next;
        }
        return tmp.jarak;
    }

    public int get(int index) throws Exception {
        if (isEmpty() || index >= size) {
            throw new Exception("Nilai indeks di luar batas");
        }
        Node08 tmp = head;
        for (int i = 0; i < index; i++) {
            tmp = tmp.next;
        }
        return tmp.data;
    }

    public void remove(int index) throws Exception {
        if (index < 0 || index >= size) {
            throw new Exception("Nilai indeks di luar batas");
        }
        Node08 current = head;
        for (int i = 0; i < index; i++) {
            current = current.next;
        }
        if (current.prev != null) {
            current.prev.next = current.next;
        } else {
            head = current.next;
        }
        if (current.next != null) {
            current.next.prev = current.prev;
        }
        size--;
    }

    public int size() {
        return size;
    }
```

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

    public void updateJarak(int index, int jarak) throws Exception {
        if (isEmpty() || index >= size) {
            throw new Exception("Nilai Index Di luar batas");


        }
        Node08 tmp = head;
        for (int i = 0; i < index; i++) {
            tmp = tmp.next;
        }
        tmp.jarak = jarak;
        }
}
```

Kode Node08

```
public class Node08 {
    int data;
    Node08 prev,next;
    int jarak;

    Node08 (Node08 prev, Node08 next, int data, int jarak){
        this.prev = prev;
        this.next = next;
        this.data = data;
        this.jarak = jarak;
    }
}
```

Kode Main

```
import java.util.Scanner;

public class GraphMain08 {
    public static void main(String[] args) throws Exception {
        Scanner scanner = new Scanner(System.in);
        Graph08 gedung = new Graph08(6);

        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 choice = scanner.nextInt();
```

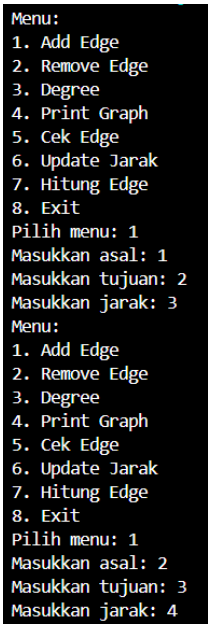
```
            switch (choice) {
                case 1:
                    System.out.print("Masukkan asal: ");
                    int asal = scanner.nextInt();
                    System.out.print("Masukkan tujuan: ");
                    int tujuan = scanner.nextInt();
                    System.out.print("Masukkan jarak: ");
                    int jarak = scanner.nextInt();
                    gedung.addEdge(asal, tujuan, jarak);
                    break;
                case 2:
                    System.out.print("Masukkan asal: ");
                    asal = scanner.nextInt();
                    System.out.print("Masukkan tujuan: ");
                    tujuan = scanner.nextInt();
                    gedung.removeEdge(asal, tujuan);
                    break;
                case 3:
                    System.out.print("Masukkan vertex: ");
                    asal = scanner.nextInt();
                    gedung.degree(asal);
                    break;
                case 4:
                    gedung.printGraph();
                    break;
                case 5:
                    System.out.print("Masukkan asal: ");
                    asal = scanner.nextInt();
                    System.out.print("Masukkan tujuan: ");
                    tujuan = scanner.nextInt();
                    if (cekEdge(gedung, asal, tujuan)) {
                        System.out.println("Edge ada");
                    } else {
                        System.out.println("Edge tidak ada");
                    }
                    break;
                case 6:
                    System.out.print("Masukkan asal: ");
                    asal = scanner.nextInt();
                    System.out.print("Masukkan tujuan: ");
                    tujuan = scanner.nextInt();
                    System.out.print("Masukkan jarak baru: ");
                    jarak = scanner.nextInt();
                    gedung.updateJarak(asal, tujuan, jarak);
                    break;
                case 7:
                    System.out.println("Jumlah edge: " +
gedung.hitungEdge());
                    break;
                case 8:
                    System.out.println("Keluar dari program.");
                    scanner.close();
```

```
                    return;
                default:
                    System.out.println("Pilihan tidak valid.");
            }
        }
    }

    public static boolean cekEdge(Graph08 graph, int asal, int tujuan)
throws Exception {
        for (int i = 0; i < graph.list[asal].size(); i++) {
            if (graph.list[asal].get(i) == tujuan) {
                return true;
            }
        }
        return false;
    }
}
```

Hasil run

1. Add Edge



2. remove Edge

```
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: 1
Masukkan tujuan: 3
```

3. Degree

```
Masukkan vertex: 1
InDegree dari Gedung B : 0
OutDegree dari Gedung B : 1
Degree dari Gedung B : 1
```

4. Print Graph

```
Gedung B terhubung dengan C (3 m),
Gedung C terhubung dengan D (4 m),
```

5. Cek Edge

```
Pilih menu: 5
Masukkan asal: 1
Masukkan tujuan: 2
Edge ada
```

6. Update Jarak

```
Pilih menu: 6
Masukkan asal: 1
Masukkan tujuan: 2
Masukkan jarak baru: 4
```

7. Hitung Edge

```
Pilih menu: 7
Jumlah edge: 2
```