Jobsheet 12

Algoritma dan Struktur Data

Disusun Oleh :

DHANISA PUTRI MASHILFA

NIM. 2341720212

TI-1E/07

D-IV TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

Jl. Soekarno Hatta No. 9 Jatimulyo, Kecamatan Lowokwaru, Jatimulyo, Kec. Lowokwaru, Kota Malang, Jawa Timur 6514

2.1 Percobaan Praktikum 1

2.1.1 Langkah - Langkah Percobaan Praktikum 1

→ Node07

```
public class Node07{
    int data, jarak;
    Node07 prev, next;

    Node07(Node07 prev, int data, int jarak, Node07 next){
        this.prev = prev;
        this.data = data;
        this.jarak = jarak;
        this.next = next;
    }
}
```

→ Graph07

```
public class Graph07 {
    int vertex;
    DoubleLinkedList07 list[];

    public Graph07(int v) {
        vertex = v;
        list = new DoubleLinkedList07[v];
        for (int i = 0; i < v; i++) {
            list[i] = new DoubleLinkedList07();
        }
    }

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

    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));
    }
```

```
    public void removeEdge(int asal, int tujuan) {
        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] = new DoubleLinkedList07();
        }
        System.out.println("Graph berhasil dikosongkan");
    }

    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 jalur(int asal, int tujuan) throws Exception {

        for (int i = 0; i < list[asal].size(); i++) {
            if (list[asal].get(i) == tujuan) {
                System.out.println("Gedung " + (char)('A' + asal) + " dan
Gedung " + (char)('A' + tujuan));
            }
        }
        System.out.println("Gedung " + (char)('A' + asal) + " Tidak
Bertetangga dengan " + (char)('A' + tujuan));
    }
}
```

→ DoubleLinkedList07

```
public class DoubleLinkedList07 {

    Node07 head;
    int size;

    public DoubleLinkedList07() {
        head = null;
        size = 0;
```

```
    }

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

    public void addFirst(int item, int jarak) {
        if (isEmpty()) {
            head = new Node07(null, item, jarak, null);
        } else {
            Node07 newNode07 = new Node07(null, item, jarak, head);
            head.prev = newNode07;
            head = newNode07;
        }
        size++;
    }

    public int getJarak(int index) throws Exception {
        if (isEmpty() || index >= size) {
            throw new Exception("Nilai indeks di luar batas");
        }
        Node07 tmp = head;
        for (int i = 0; i < index; i++) {
            tmp = tmp.next;
        }
        return tmp.jarak;
    }

    public void remove(int index) {
        Node07 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 void clear() {
        head = null;
        size = 0;
    }

    public int size() {
```

```
        return size;
    }

    public int get(int index) throws Exception {
        if (isEmpty() || index >= size) {
            throw new Exception("Nilai indeks di luar batas");
        }
        Node07 tmp = head;
        for (int i = 0; i < index; i++) {
            tmp = tmp.next;
        }
        return tmp.data;
    }
}
```

→ GraphMain07

```
public class GraphMain07 {
    public static void main(String[] args) throws Exception{

        Graph07 gedung = new Graph07(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();
        gedung.jalur(1, 4);
    }
}
```

2.1.2 Verifikasi Hasil Percobaan Praktikum 1

Hasil running pada 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),
```

```
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 running pada langkah 17

```
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),
```

```
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 Percobaan Praktikum 1

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

   → Method Remove dalam class DoubleLinkedList07

```java
public void remove(int index) {
        Node07 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;
        }
    }
```

2. Pada class Graph, terdapat atribut **list[]** bertipe DoubleLinkedList. Sebutkan tujuan pembuatan variabel tersebut!
   → Untuk menyimpan data yang terdapat dalam double linked list

3. Jelaskan alur kerja dari method **removeEdge**!
   1. Mencari edge yang ditentukan dalam grafik, jika tidak ditemukan, akan diberikan pengecualian yang sesuai.
   2. Menghilangkan edge dari struktur data internal grafik
   3. Memperbarui properti grafik yang terpengaruh oleh penghapusan edge.

4. Apakah alasan pemanggilan method **addFirst()** untuk menambahkan data, bukan method add jenis lain saat digunakan pada method **addEdge** pada class Graph?
   → Penggunaan method **addFirst(),** digunakan untuk menambahkan sebuat data baru ke dalam suatu double linked list, karen .next dapat digunakan untuk menyimpan edge baru.

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
```

   Menambahkan sebuah method dalam class Graph07

```
public void jalur(int asal, int tujuan) throws Exception {

        for (int i = 0; i < list[asal].size(); i++) {
            if (list[asal].get(i) == tujuan) {
                System.out.println("Gedung " + (char)('A' + asal) +
" dan Gedung " + (char)('A' + tujuan));
            }
        }
        System.out.println("Gedung " + (char)('A' + asal) + " Tidak
Bertetangga dengan " + (char)('A' + tujuan));
    }
```

```
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),

Gedung B Tidak Bertetangga dengan E
```

2.2 Percobaan Praktikum 2

2.2.1 langkah - Langkah Percobaan Praktikum 2

→ GraphMatrix07

```java
public class GraphMatrikx07 {

    int vertex;
    int [][] matriks;

    public GraphMatrikx07 (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] = -1;
    }

    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();
        }
    }
```

```
}
```

→ GraphMain07

```
public class GraphMain07 {
    public static void main(String[] args) throws Exception{

        GraphMatrikx07 gdg = new GraphMatrikx07(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();
    }
}
```

2.2.2 Verifikasi Hasil Percobaan Praktikum 2

Verifikasi hasil kompilasi kode program Anda dengan gambar berikut ini.

```
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 : Gedung A ( 0 m), Gedung B ( 50 m), Gedung C ( 0 m), Gedung D ( 0 m),  1edGedung A :
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),
```

2.2.3 Pertanyaan Percobaan Praktikum 2

1. Perbaiki kode program Anda apabila terdapat error atau hasil kompilasi kode tidak sesuai!
   → sudah sesuai, tidak ada kode program yang bermasalah

2. Apa jenis graph yang digunakan pada Percobaan 2?
   → Directed unweighted

3. Apa maksud dari dua baris kode berikut?

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

→ Kode tersebuat berisikan sebuah nilai untuk nanti menjadi subah edge yang terarah di dalam suatu graf

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

```
public void degree(int asal) {
        int inDegree = 0;
        int outDegree = 0;

        for (int j = 0; j < vertex; j++) {
            if (matriks[asal][j] != 0) {
                outDegree++;
            }
        }

        for (int i = 0; i < vertex; i++) {
            if (matriks[i][asal] != 0) {
                inDegree++;
            }
        }
        System.out.println("InDegree dari gedung " + (char) ('A' +
asal) + ": " + inDegree);
        System.out.println("OutDegree dari gedung " + (char) ('A' +
asal) + ": " + outDegree);
        System.out.println("Degree dari Gedung " + (char) ('A' +
asal)  + ": " + (inDegree + outDegree));
    }
```

```
Gedung A : Gedung A ( 0 m), Gedung B ( 50 m), Gedung C ( 0 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),
InDegree dari gedung A: 2
OutDegree dari gedung A: 1
Degree dari Gedung A: 3
```