LAPORAN HASIL PRAKTIKUM
ALGORITMA DAN STRUKTUR DATA
GRAPH

Oleh:

FAUZIYYAH ADELIA RAMANDA

NIM. 2341760145

SIB-1F / 10

D-IV SISTEM INFORMASI BISNIS

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

2.1 2.1 Percobaan 1: Implementasi Graph menggunakan Linked List

2.2.1 Langkah Percobaan

1. Membuat Class Node

```
Graph > Node10.java > Node10 > Node10(Node10, int, int, Node10)
1   public class Node10 {
2       int data;
3       Node10 prev, next;
4       int jarak;
5
6       Node10(Node10 prev, int data, int jarak, Node10 next) {
7           this.prev=prev;
8           this.data=data;
9           this.next=next;
10          this.jarak = jarak;
11      }
12  }
```

2. Menyalin class DoubleLinkedList

```
12  public void addFirst(int item, int jarak) {
13  if (isEmpty()){
14      head = new Node10(prev:null, item, jarak, next:null);
15  } else {
16      Node10 newNode = new Node10(prev:null, item, jarak, head);
17      head.prev = newNode;
18      head = newNode;
19  }
20  size++;
21  }
```

```
138   public int getJarak(int index) throws Exception {
139       if (isEmpty() || index >= size) {
140           throw new Exception (message:"Nilai indeks di luar batas");
141       }
142       Node10 tmp = head;
143       for (int i = 0; i < index; i++){
144           tmp = tmp.next;
145       }
```

```
public void remove (int index) {
    Node10 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;
            }
            break;
        }
        current = current.next;
    }
}
```

3. Membuat class graph dan konstruktornya

```
public Graph10(int v) {
    vertex = v;
    list = new DoubleLinkedList10[v];
    for(int i = 0; i < v; i++){
        list= new DoubleLinkedList10[v];
        for (int i = 0; i < v; i++){
            list[i] = new DoubleLinkedList10();
```

4. Membuat method addEdge

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

5. Menambahkan method addFirst

```
public void addFirst(int item, int jarak) {
if (isEmpty()){
    head = new Node10(prev:null, item, jarak, next:null);
} else {
    Node10 newNode = new Node10(prev:null, item, jarak, head);
    head.prev = newNode;
    head = newNode;
}
size++;
}
```

6. Menambahkan method degree

```
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);
}
```

7. Menambahkan method remove edge

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

8. Membuat method print graph

```
public void printGrpah10() 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.println((char) ('A' + list[i].get(j)) + "(" + list[i].getJarak(j)+ " m), ");
            }
            System.out.println("");
        }
    }
    System.out.println("");
```

9. Membuat main class

```
Graph > GraphMain10.java > GraphMain10 > main(String[])
1   public class GraphMain10 {
        Run | Debug
2       public static void main(String[] args) throws Exception {
3   Graph10 gedung = new Graph10(v:6);
4   gedung.addEdge(asal:0, tujuan:1, jarak:50);
5   gedung.addEdge(asal:0, tujuan:2, jarak:100);
6   gedung.addEdge(asal:1, tujuan:3, jarak:70);
7   gedung.addEdge(asal:2, tujuan:3, jarak:40);
8   gedung.addEdge(asal:3, tujuan:4, jarak:60);
9   gedung.addEdge(asal:4, tujuan:5, jarak:80);
10  gedung.degree(asal:0);
11  gedung.printGrpah10();
12
13  }
14  }
```

2.1.3 Langkah Percobaan

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 pembuatanvariabel tersebut!

    - Untuk menyimpan daftar tetangga (adjency list) dari stiap vertex dalam graph

3. Jelaskan alur kerja dari method **removeEdge**!
    1. Validasi : Pastikan node yang ingin dihapus ada dan terhubung
    2. Penghapusan Koneksi : Hapus node dari daftar tetangga masing-masing, lalu perbarui degree node, yang trakhir hapus edge (jika graf terarah)
    3. Pembaruan Struktur Graf : Perbarui struktur data internal graf, perbarui matriks adjasensi (jika ada), hapus entri daftar adjasensi (jika ada)
    4. Penanganan Kasus Khusus : Perbarui bobot graf(jika graf tertimbang), hapus node terisolasi (jika ada)
    5. Pengembangan Nilai : Kembalikan status operasi (berhasil/gagal)
4. Apakah alasan pemanggilan method **addFirst()** untuk menambahkan data, bukan method addjenis lain saat digunakan pada method **addEdge** pada class Graph?

    Karena efisiensi addFirst() lebih cepat karna tidak menggeser elemen lain, dan representasi koneksi addFirst() dapat mewakili urutan edge diproses, lalu kemudahan implementasi addFirst() menyederhanakan implementasi addEdge

5. Modifikasi kode program sehingga dapat dilakukan pengecekan apakah terdapat jalur antarasuatu 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
```

2.2 Percobaan 2: Implementasi Graph menggunakan Matriks

2.2.1 Langkah Percobaan

1. Membuat file GraphMatriks.java dan menambahlan konstruktor

```
Graph > GraphMatriks10.java > GraphMatriks10 > printGraph()
public class GraphMatriks10 {
    int vertex;
    int [][] matriks;

    public GraphMatriks10(int v) {
        vertex = v;
        matriks = new int [v] [v] ;
    }
```

2. Membuat method makeEdge()

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

3. Membuat removeEdge()

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

4. Tambahkan method printGraph()

```
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.println("Gedung " + (char) ('A' + j) + "("+ matriks[i][j] + " m).");
            }
        }
        System.out.println();
```

5.Menambahkan kode pada file GraphMain.java

```
GraphMatriks10 gdg = new GraphMatriks10(v:4);
gedung.makeEdge(i:0, j:1, k:50);
gedung.makeEdge(i:1, j:0, k:60);
gedung.makeEdge(i:1, j:2, k:70);
gedung.makeEdge(i:2, j:1, k:80);
gedung.makeEdge(i:2, j:3, k:40);
gedung.makeEdge(i:3, j:0, k:90);

gedung.printGraph10();

System.out.println(x:"Hasil setelah penghapusan edge");
gedung.removeEdge(asal:2, tujuan:1);

gedung.printGraph10();

}
}
```

2.2.1 Pertanyaan

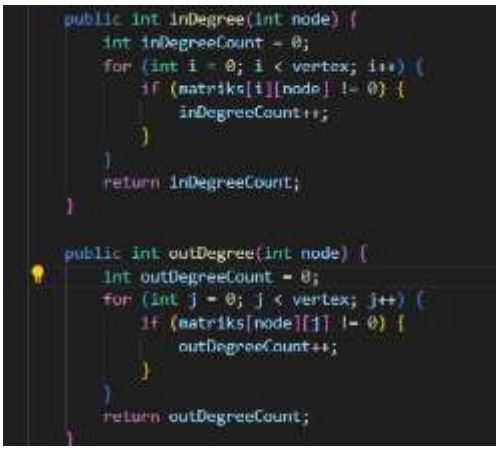
1. Perbaiki kode program Anda apabila terdapat error atau hasil kompilasi kode tidak sesuai!
2. Apa jenis graph yang digunakan pada Percobaan 2?

   Graf matriks, ialah cara unruk merepresentasikan struktur graf dengan menggunakan matriks, yaitu susunan bilangan yang tersusun dalam baris dan kolom
3. Apa maksud dari dua baris kode berikut?

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

Untuk menambahkan edge diantara 2 node dalam graf baris matriks

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

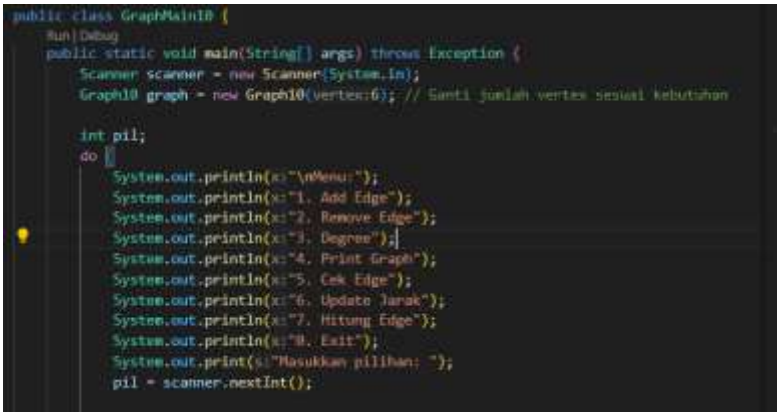


Latihan Praktikum

Waktu percobaan: 90 menit

1. Modifikasi kode program pada class **GraphMain** sehingga terdapat menu program yang bersifatdinamis, 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 jarakantara dua node asal dan tujuan!

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


```
Menu:
1. Add Edge
2. Remove Edge
3. Degree
4. Print Graph
5. Cek Edge
6. Update Jarak
7. Hitung Edge
0. Exit
Masukkan pilihan: 1
Masukkan gedung asal: 1
Masukkan gedung tujuan: 2
Masukkan jarak: 40
```

```
Menu:
1. Add Edge
2. Remove Edge
3. Degree
4. Print Graph
5. Cek Edge
6. Update Jarak
7. Hitung Edge
0. Exit
Masukkan pilihan: 4
Gedung B terhubung dengan: C(40m),
Gedung C terhubung dengan: B(40m),

Menu:
1. Add Edge
2. Remove Edge
3. Degree
4. Print Graph
5. Cek Edge
6. Update Jarak
7. Hitung Edge
0. Exit
Masukkan pilihan: 5
Masukkan gedung asal: 1
Masukkan gedung tujuan: 4
Gedung B dan Gedung E tidak bertetangga
```