Nama : Alfina Salsabilla

NIM : 2141720044

Kelas / Absen : TI-1G / 03

Tugas :

JOBSHEET 15 GRAPH!

12.2.3 Pertanyaan Percobaan

1. Sebutkan beberapa jenis (minimal 3) algoritma yang menggunakan dasar Graph, dan apakah kegunaan algoritma-algoritma tersebut?

   JAWABAN 1. Pencarian Melebar (Breadth First Search atau BFS) Untuk Mencari Pencarian Mendalam (Depth First Search atau DFS) Untuk Mencari 3. Algoritma Bellman-Ford Untuk mencari lintasan terpendek 4. Algoritma Boruvska Untuk menentukan pohon penjangkau minimum
2. Pada class Graph terdapat array bertipe LinkedList, yaitu LinkedList list[]. Apakah tujuan pembuatan variabel tersebut ?

   JAWABAN Untuk memanggil objek Linkedlist dan mengubahnya menjadi objek array yang mana nantinya akan di isi oleh vertex
3. Apakah alasan pemanggilan method addFirst() untuk menambahkan data, bukan method add jenis lain pada linked list ketika digunakan pada method addEdge pada class Graph?

   JAWABAN Alasannya adalah untuk dapat mengenalkan data dari depan
4. Bagaimana cara mendeteksi prev pointer pada saat akan melakukan penghapusan suatu edge pada graph ?

   JAWABAN Dengan cara looping edge, jika vertex lebih besar dari I dan destination sama dengan I maka edge akan otomatis di hapus
5. Kenapa pada praktikum 12.2 langkah ke-12 untuk menghapus path yang bukan merupakan lintasan pertama kali menghasilkan output yang salah ? Bagaimana solusinya ?

   JAWABAN Output tidak error tapi data vertex yang di keluarkan mengalami perubahan path / lintasan

12.3.3 Pertanyaan Percobaan

1. Apakah perbedaan degree/derajat pada directed dan undirected graph?

   JAWABAN Directed gaph degree mempunyai nilai yang berbeda karena terdapat in out Sedangkan undirected degree mempunyai nilai yang sama karena tidak terdapat in out

2. Pada implementasi graph menggunakan adjacency matriks. Kenapa jumlah vertices harus ditambahkan dengan 1 pada indeks array berikut ?

   JAWABAN Karena pada matriks index di mulai dari 0, agar vertex dalam matrix bernilai sama dengan vertex yang seharusnya maka harus di tambah +1

3. Apakah kegunaan method getEdge() ?

   JAWABAN Untuk menampilkan apakah vertex tersedia atau tidak

4. Termasuk jenis graph apakah uji coba pada praktikum 12.3 ?

   JAWABAN Jenis directed graph

5. Mengapa pada method main harus menggunakan try-catch Exception ?

   JAWABAN Agar program yang error dapat tetap berjalan dan tidak terhent

```
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package jobsheet14;

/**
 *
 * @author HP
 */
public class Graph {
    int vertex;
    DoubleLinkedLists list[];
    Node right;

    Graph(int vertex){
        this.vertex = vertex;
        list = new DoubleLinkedLists[vertex];
        for(int i=0; i < vertex; i++){
            list[i] = new DoubleLinkedLists();
        }
    }

    public void addEdge(int source, int destination){
        list[source].addFirst(destination);
        list[destination].addFirst(source);
    }

    public void degree(int source) throws Exception{
        System.out.println("Degree vertex " + source + " : " + list[source].size());
```

```
    public void degree(int source) throws Exception{
        System.out.println("Degree vertex " + source + " : " + list[source].size());
        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)==source)
                    ++totalIn;
            }
            for (k = 0; k < list[source].size(); k++){
                list[source].get(k);
            }
            totalOut = k;
            System.out.println("Indegree dari vertex " + source + " : " + totalIn);
            System.out.println("Outdegree dari vertex " + source + " : " + totalOut);
            System.out.println("Degree vertex " + source + " : " + (totalIn + totalOut));
        }
    }

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

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

```
                System.out.println(" ");
                for(int i = 1; i <= v; i++){
                    System.out.print(i + " ");
                }
                System.out.println();

                for(int i = 1; i <= v; i++){
                    System.out.print(i + " ");
                    for (int j = 1; j <= v; j++){
                        System.out.print(graph.getEdge(i, j) + " ");
                    }
                    System.out.println();
                }
            }catch(Exception E){
                System.out.println("error. silakan cek kembali " + E.getMessage());
                sc.close();
            }
        }
    }

```

```
run:
vertex 0 Terhubung dengan : 0 1 2
vertex 1 Terhubung dengan : 0 1 2 3
vertex 2 Terhubung dengan : 0 1
vertex 3 Terhubung dengan : 0 1 2 3
vertex 4 Terhubung dengan : 0 1 2

Degree vertex 2 : 2
Indegree dari vertex 2 : 1
Outdegree dari vertex 2 : 2
Degree vertex 2 : 3
Indegree dari vertex 2 : 2
Outdegree dari vertex 2 : 2
Degree vertex 2 : 4
Indegree dari vertex 2 : 2
Outdegree dari vertex 2 : 2
Degree vertex 2 : 4
Indegree dari vertex 2 : 3
Outdegree dari vertex 2 : 2
Degree vertex 2 : 5
Indegree dari vertex 2 : 4
Outdegree dari vertex 2 : 2
Degree vertex 2 : 6
Indegree dari vertex 2 : 4
Outdegree dari vertex 2 : 2
Degree vertex 2 : 6
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
run:
Masukkan jumlah vertices :
5
Masukkan jumlah edges :
6
Masukkan edges : <to> <from>
1 2
1 5
2 3
2 4
2 5
3 4
ARRAY 2D Sebagai representasi graph sbb:

1 2 3 4 5
1 0 1 0 0 1
2 0 0 1 1 1
3 0 0 0 1 0
4 0 0 0 0 0
5 0 0 0 0 0
BUILD SUCCESSFUL (total time: 43 seconds)
```

```
            }
        }

        public void removeAllEdges(){
            for(int i =0; i < vertex; i++){
                list[i].clear();
            }
            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("vertex " + i + " Terhubung dengan : ");
                    for ( int j = 0; j < list[i].size(); j++){
                        System.out.print(list[i].get(j) + " ");
                    }
                    System.out.println(" ");
                }
            }
            System.out.println(" ");
        }


    }
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package jobsheet14;

/**
 *
 * @author HP
 */
public class GraphMain {
    public static void main(String[] args) throws Exception{
        Graph graph = new Graph(6);
        graph.addEdge(0,1);
        graph.addEdge(0,4);
        graph.addEdge(1,2);
        graph.addEdge(1,3);
        graph.addEdge(1,4);
        graph.addEdge(2,3);
        graph.addEdge(3,4);
        graph.addEdge(3,0);
        graph.printGraph();
        graph.degree(2);
    }
}
```

```
 * @author HP
 */
public class GraphArray {
    private final int vertices;
    private final int[][] twoD_array;

    GraphArray(int v){
        vertices = v;
        twoD_array = new int[vertices + 1][vertices + 1];
    }

    GraphArray() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | T
    }

    public void makeEdge(int to, int from, int edge){
        try {
          twoD_array[to] [from] = edge;
        }catch(ArrayIndexOutOfBoundsException index){
            System.out.println("Vertex Tidak ada");
        }
    }

    public int getEdge(int to, int from){
        try{
            return twoD_array[to][from];
        }catch(ArrayIndexOutOfBoundsException index){
            System.out.println("vertex tidak ada");
        }
```

```
 */
public class GraphArrayMain {
    public static void main(String args []){
        int v, e, count = 1, to = 0, from = 0;
        Scanner sc = new Scanner(System.in);
        GraphArray graph;
        try{
            System.out.println("Masukkan jumlah vertices : ");
            v = sc.nextInt();
            System.out.println("Masukkan jumlah edges :");
            e = sc.nextInt();

            graph = new GraphArray(v);

            System.out.println("Masukkan edges : <to> <from>");
            while (count <= e){
              to = sc.nextInt();
              from = sc.nextInt();
              graph.makeEdge(to, from, 1);
              count++;
            }
            System.out.println("ARRAY 2D Sebagai representasi graph sbb: ");
            System.out.println(" ");
            for(int i = 1; i <= v; i++){
                System.out.print(i + " ");
            }
            System.out.println();

            for(int i = 1; i <= v; i++){
                System.out.print(i + " ");
```