

JOBSHEET 15

Nama : Farid Aziz Wicaksono

Kelas : TI-1C

Absen : 14

A. Praktikum

1. Praktikum 1

No	Node.java
1	package Jobsheet15;
2	
3	public class Node {
4	
5	int data;
6	Node next;
7	
8	public Node(int data, Node next) {
9	this.data = data;
10	this.next = next;
11	}
12	}

No	LinkedList.java
1	package jobsheet15;
2	
3	public class LinkedList {
4	
5	Node head;
6	int size;
7	
8	public LinkedList() {
9	head = null;
10	size = 0;
11	}
12	
13	public boolean isEmpty() {
14	return head == null;
15	}
16	
17	public void addFirst(int item) {
18	head = new Node(item, head);
19	size++;
20	}
21	
22	public void add(int item, int index) throws Exception {
23	if (index < 0 index > size) {
24	throw new Exception("Nilai index di luar batas");
25	}

```

26     if (isEmpty() || index == 0) {
27         addFirst(item);
28     } else {
29         Node tmp = head;
30         for (int i = 1; i < index; i++) {
31             tmp = tmp.next;
32         }
33         Node next = (tmp == null) ? null : tmp.next;
34         tmp.next = new Node(item, next);
35     }
36     size++;
37 }
38
39 public void addLast(int item) {
40     if (isEmpty()) {
41         addFirst(item);
42     } else {
43         Node tmp = head;
44         while (tmp.next != null) {
45             tmp = tmp.next;
46         }
47         tmp.next = new Node(item, null);
48     }
49     size++;
50 }
51
52 public int getFirst() throws Exception {
53     if (isEmpty()) {
54         throw new Exception("LinkedList Kosong");
55     }
56     return head.data;
57 }
58
59 public int getLast() throws Exception {
60     if (isEmpty()) {
61         throw new Exception("LinkedList Kosong");
62     }
63     Node tmp = head;
64     while (tmp.next != null) {
65         tmp = tmp.next;
66     }
67     return tmp.data;
68 }
69
70 public int get(int index) throws Exception {
71     if (isEmpty() || index >= size) {

```

```

72         throw new Exception("Nilai index di luar batas");
73     }
74     Node tmp = head;
75     for (int i = 0; i < index; i++) {
76         tmp = tmp.next;
77     }
78     return tmp.data;
79 }
80
81 public void remove(int index) throws Exception {
82     if (isEmpty() || index >= size) {
83         throw new Exception("Nilai index di luar batas");
84     }
85     if (index == 0) {
86         removeFirst();
87     } else {
88         Node prev = head;
89         Node cur = head.next;
90         for (int i = 1; i < index; i++) {
91             prev = cur;
92             cur = prev.next;
93         }
94         prev.next = cur.next;
95         size--;
96     }
97 }
98
99 public void removeFirst() {
100     head = head.next;
101     size--;
102 }
103
104 public void clear() {
105     head = null;
106     size = 0;
107 }
108
109 public void print() {
110     if (!isEmpty()) {
111         Node tmp = head;
112         while (tmp != null) {
113             System.out.print(tmp.data + "\t");
114             tmp = tmp.next;
115         }
116         System.out.println();
117     } else {

```

119	System.out.println("LinkedList Kosong");
120	}
121	}
122	
123	public int size() {
124	return size;
125	}
126	}

No	Graph.java
1	public class Graph {
2	int vertex;
3	LinkedList list[];
4	
5	public Graph(int vertex) {
6	this.vertex = vertex;
7	list = new LinkedList[vertex];
8	for (int i = 0; i < vertex; i++) {
9	list[i] = new LinkedList();
10	}
11	}
12	
13	public void addEdge(int source, int destination) {
14	list[source].addFirst(destination);
15	list[destination].addFirst(source);
16	}
17	
18	public void degree(int source) throws Exception {
19	System.out.println("degree vertex " + source + " : " + list[source].size());
20	int k, totalIn = 0, totalOut = 0;
21	for (int i = 0; i < vertex; i++) {
22	for (int j = 0; j < list[i].size(); j++) {
23	if (list[i].get(j) == source) {
24	++totalIn;
25	}
26	}
27	for (k = 0; k < list[source].size(); k++) {
28	list[source].get(k);
29	}
30	totalOut = k;
31	}
32	System.out.println("Indegree dari vertex " + source + " : " + totalIn);
33	System.out.println("Outdegree dari vertex " + source + " : " + totalOut);
34	System.out.println("degree vertex " + source + " : " + (totalIn + totalOut));
35	}
36	

```

37 public void removeEdge(int source, int destination) throws Exception {
38     for (int i = 0; i < vertex; i++) {
39         if (i == destination) {
40             list[source].remove(destination);
41         }
42     }
43 }
44
45 public void removeAllEdges() {
46     for (int i = 0; i < vertex; i++) {
47         list[i].clear();
48     }
49     System.out.println("Graph berhasil dikosongkan");
50 }
51
52 public void printGraph() throws Exception {
53     for (int i = 0; i < vertex; i++) {
54         if (list[i].size() > 0) {
55             System.out.println("Vertex " + i + " terhubung dengan: ");
56             for (int j = 0; j < list[i].size(); j++) {
57                 System.out.println(list[i].get(j) + " ");
58             }
59             System.out.println("");
60         }
61     }
62     System.out.println(" ");
63 }
64 }
65
66 class graphMain {
67     public static void main(String[] args) throws Exception {
68         Graph graph = new Graph(6);
69         graph.addEdge(0, 1);
70         graph.addEdge(0, 4);
71         graph.addEdge(1, 2);
72         graph.addEdge(1, 3);
73         graph.addEdge(1, 4);
74         graph.addEdge(2, 3);
75         graph.addEdge(3, 4);
76         graph.addEdge(3, 0);
77         graph.printGraph();
78         graph.degree(2);
79         graph.removeEdge(1, 2);
80         graph.printGraph();
81     }
82 }

```

Output :

```
Run:
Vertex 0 terhubung dengan : 3 4 1
Vertex 1 terhubung dengan : 4 3 2 0
Vertex 2 terhubung dengan : 3 1
Vertex 3 terhubung dengan : 0 4 2 1
Vertex 4 terhubung dengan : 3 1 0

degree vertex2 : 2
Indegree dari vertex 2 : 2
Outdegree dari vertex 2 : 0
Degree vertex 2 : 2
Vertex 0 terhubung dengan : 3 4 1
Vertex 1 terhubung dengan : 4 3 0
Vertex 2 terhubung dengan : 3 1
Vertex 3 terhubung dengan : 0 4 2 1
Vertex 4 terhubung dengan : 3 1 0

BUILD SUCCESSFUL (total time: 0 seconds)
```

2. Praktikum 2

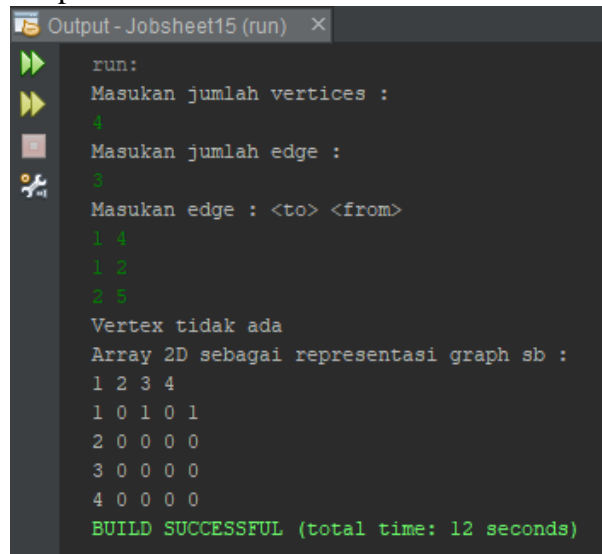
No	graphArray.java
1	package jobsheet15;
2	import java.util.*;
3	public class graphArray {
4	private final int vertices;
5	private int[][] twoD_array;
6	
7	public graphArray(int v){
8	vertices = v;
9	twoD_array = new int[vertices + 1][vertices + 1];
10	}
11	
12	public void makeEdge(int to, int from, int edge){
13	try{
14	twoD_array[to][from] = edge;
15	}
16	catch (ArrayIndexOutOfBoundsException index){
17	System.out.println("Vertex tidak ada");
18	}
19	}
20	
21	public int getEdge(int to, int from){
22	try{
23	return twoD_array[to][from];
24	}
25	catch (ArrayIndexOutOfBoundsException index){
26	System.out.println("Vertex tidak ada");
27	}
28	return -1;
29	}

```

30
31 public static void main(String[] args) {
32     int v, e, count = 1, to = 0, from = 0;
33     Scanner sc = new Scanner(System.in);
34     Scanner sa = new Scanner(System.in);
35     graphArray graph;
36     try {
37         System.out.println("Masukan jumlah vertices : ");
38         v = sc.nextInt();
39         System.out.println("Masukan jumlah edge : ");
40         e = sa.nextInt();
41         graph = new graphArray(v);
42         System.out.println("Masukan edge : <to> <from>");
43         while (count <= e) {
44             to = sc.nextInt();
45             from = sa.nextInt();
46             graph.makeEdge(to, from, 1);
47             count++;
48         }
49         System.out.println("Array 2D sebagai representasi graph sb : ");
50         System.out.print("");
51         for (int i = 1; i <= v; i++) {
52             System.out.print(i + " ");
53         }
54         System.out.println();
55         for (int i = 1; i <= v; i++) {
56             System.out.print(i + " ");
57             for (int j = 1; j <= v; j++) {
58                 System.out.print(graph.getEdge(i, j) + " ");
59             }
60             System.out.println();
61         }
62     } catch (Exception E) {
63         System.out.println("Error. Silakan cek kembali");
64     }
65     sc.close();
66 }
67 }

```

Output :



```
run:
Masukan jumlah vertices :
4
Masukan jumlah edge :
3
Masukan edge : <to> <from>
1 4
1 2
2 5
Vertex tidak ada
Array 2D sebagai representasi graph sb :
1 2 3 4
1 0 1 0 1
2 0 0 0 0
3 0 0 0 0
4 0 0 0 0
BUILD SUCCESSFUL (total time: 12 seconds)
```

B. Pertanyaan

1. Mengapa graph diimplementasikan dengan double linked list, bukan single linked list?

Jawab :

karena graph membutuhkan sebuah node yang digunakan untuk indegree dan outdegree.

2. Jelaskan masing-masing pada dua jenis looping yang terdapat dalam metod printgraph()!

Jawab :

looping yang pertama digunakan untuk menampilkan vertex ke berapa looping yang kedua digunakan untuk mencari nomor yang terhubung ke vertex

3. Apakah perbedaan graph dengan binary tree pada implementasi nya menggunakan linked list?

Jawab :

jika dalam binary tree, node yang terkecil akan diletakkan di left, sedangkan node terbesar dari parent akan diletakkan di bagian kanan jika dalam graph, vertex akan diletakkan di bagian kiri, sedangkan di bagian kanan akan memasukkan lintasan dari vertex.

4. Jelaskan dengan contoh perbedaan antara edge dan path pada graph!

Jawab :

edge adalah lintasan penghubung antar simpul dalam graph, sedangkan path adalah urutan dari lintasan (edge). Contohnya lintasan j ke p, yang dapat kita sebut sebagai lintasan (path) j b c p adalah lintasan (path) dari simpul j ke p.

5. Sebutkan beberapa contoh (minimal 3) implementasi graph dalam permasalahan yang membutuhkan representasi internal dalam memori komputer untuk suatu struktur data!

Jawab :

digunakan untuk pembuatan website, digunakan untuk mendesain web dalam mengalokasi memori computer, pengaplikasian dalam membuat suatu program oleh para programmer

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

Jawab :

- Algoritma djikstra merupakan salah satu varian bentuk algoritma populer dalam pemecahan persoalan terkait masalah optimasi pencarian lintasan terpendek sebuah lintasan yang mempunyai panjang minimum dari verteks a ke z dalam graph berbobot, bobot tersebut adalah bilangan positif jadi tidak dapat dilalui oleh node negative
- Algoritma greedy merupakan jenis algoritma yang menggunakan pendekatan penyelesaian masalah dengan mencari nilai maksimum sementara pada setiap langkahnya
- Algoritma d'satur merupakan algoritma pewarnaan terurut dari derajat simpul tertinggi hingga derajat terendah. Derajat saturasi suatu simpul v_i adalah derajat simpul v_i dikurangi banyaknya warna berbeda yang sudah muncul pada tetangga v_i

7. Pada class graph terdapat array bertipe linkedlist, yaitu linkedlist list[]. Apakah tujuan pembuatan variabel tersebut?

Jawab :

array digunakan untuk vertex

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

Jawab :

undirected : graf yang semua edge miliknya bersifat bi-directional. Dengan kata lain edge, tersebut tidak memiliki arah tertentu. Directed: directed graph adalah graf yang semua edge miliknya bersifat uni-directional, dengan kata lain edge tersebut menuju ke arah tertentu.

9. apakah alasan pemanggilan method addfirst() untuk menambahkan data, bukan method add jenis lain pada linked list ketika digunakan pada method addedge pada class graph?

Jawab :

karena pada graph tidak memasukkan index, jadi menggunakan addfirst().

10. Bagaimana cara mendeteksi prev pointer pada saat akan melakukan penghapusan suatu edge pada graph?

Jawab :

akan dilakukan proses looping jika sudah sesuai maka akan dilakukan proses remove

11. kenapa uji coba praktikum bagian 1 poin 12 untuk menghapus path yang bukan merupakan lintasan pertama kali menghasilkan output error atau salah?bagaimana solusinya?

Jawab :

```
graph.removeEdge(1, 2);  
graph.PrintGraph();
```

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

Jawab :

Karena vertex dimasukkan dalam matrix sedangkan indeks pada matrix dimulai dari angka 0, jadi supaya hasil vertex sama dengan yang aslinya maka harus di +1.

```
public graphArray(int v) {  
    vertices = v;  
    twoD_array = new int[vertices + 1][vertices + 1];  
}
```

13. Apakah kegunaan method getEdge(); ?

Jawab :

Untuk menampilkan suatu lintasan diperlukan pembuatan method getEdge().

14. Termasuk jenis graph apakah uji coba pada praktikum bagian 2?

Jawab :

Uji coba graph bagian 2 menggunakan array 2 dimensi (Adjacency Matrix) sebagai representasi graph.

C. Tugas

1. Ubahlah lintasan pada praktikum bagian 1 menjadi inputan!
2. Tambahkan method `graphType` dengan tipe boolean yang akan membedakan graph termasuk directed atau undirected graph. Kemudian update seluruh method yang berelasi dengan method `graphType` tersebut (hanya menjalankan statement sesuai dengan jenis graph) pada praktikum bagian 1
3. Modifikasi class `Graph` sehingga terdapat pilihan menu pada praktikum bagian 1:
 - 1) Jenis Graph (directed/undirected)
 - 2) Input jumlah vertex
 - 3) `addEdge`
 - 4) `removeEdge`
 - 5) `removeAllEdges`
 - 6) KeluarSetiap perubahan sesuai dengan menu yang dipilih otomatis akan mencetak kondisi graph ter-update
4. Modifikasi method `removeEdge()` pada praktikum bagian 1 agar tidak menghasilkan output yang salah untuk path selain path pertama kali!
5. Buatlah class untuk membuat, menghapus dan menampilkan weighted graph menggunakan representasi adjacency list!
6. Ubahlah tipe data vertex pada seluruh graph pada praktikum bagian 1 dan 2 dari Integer menjadi String. Misalkan saja setiap vertex yang pada awalnya berupa angka 0,1,2,3, dst. Akan di rubah menjadi suatu nama daerah Malang, Surabaya, Gresik, Bandung, dst.