Nama : Andyan Yogawardhana

NIM : 21/482180/PA/21030

Kelas : KOMB1

Tugas 5 – Graph

Source Code

```java
package Adjacency;

import java.util.LinkedList;
import java.util.Queue;
import java.util.Stack;

public class AdjacencyMatrix {
    public static void main(String[] args) {
        AdjMatrix graph = new AdjMatrix(6);

        graph.addEdge(0, 1);
        graph.addEdge(0, 2);
        graph.addEdge(1, 3);
        graph.addEdge(2, 3);
        graph.addEdge(2, 4);
        graph.addEdge(3, 5);
        graph.addEdge(4, 5);

        graph.printGraph();

        graph.bfs(1);
        graph.dfs(1);
        graph.bfs(2);
        graph.dfs(2);
        graph.bfs(3);
        graph.dfs(3);
        graph.bfs(4);
        graph.dfs(4);
        graph.bfs(5);
        graph.dfs(5);
    }
}

// 1. Implementasi Adjacency Matrix
class AdjMatrix {
    private int vertices;
    private int[][] matrix;

```

```java
    // constructor dengan parameter jumlah vertex dalam graph
    public AdjMatrix(int v) {
        this.vertices = v;
        // matrix 2 dimensi untuk menampung edge (0 jika tidak berhubungan, 1 jika berhubungan)
        matrix = new int[v][v];
    }

    public void addEdge(int from, int to) {
        // mengatur edge antara vertex "from" dan vertex "to" menjadi 1 karena telah terhubung
        matrix[from][to] = 1;
        matrix[to][from] = 1;
    }

    public void printGraph() {
        System.out.println("    Matrices Adjacency");   // judul tabel
        System.out.print("  | ");
        for(int j = 0; j < vertices; j++) {
            System.out.print(j + "   ");    // judul kolom
        }
        System.out.println();

        System.out.print("--+");
        for(int j = 0; j < vertices; j++) {
            System.out.print("----");   // batas tabel
        }
        System.out.println();

        for(int i = 0; i < vertices; i++) {
            System.out.print(i + " | ");    // judul baris
            for(int k = 0; k < vertices; k++) {
                System.out.print(matrix[i][k] + "   ");     // nilai adjacency antara vertex kolom dan vertex baris
            }
            System.out.println();
        }
        System.out.println();
    }

    // 2. BFS versi Adjacency Matrix
    public void bfs(int start) {
        boolean visited[] = new boolean[vertices];  // menampung visited value suatu vertex
        Queue<Integer> queue = new LinkedList<>();

```

```
81          // mengubah value pada vertex awal menjadi true karena pertama kali dikunjungi
82          visited[start] = true;
83          // memasukkan vertex awal ke queue untuk dijadikan head bfs
84          queue.add(start);
85
86          System.out.print("BFS (Head = " + start + ") : ");
87          while(queue.size() != 0) {
88              // mengubah nilai variabel start menjadi nilai index pertama queue lalu melakukan dequeue
89              start = queue.poll();
90              // print vertex dari variabel start
91              System.out.print(start + " ");
92
93              // melakukan perulangan sebanyak jumlah vertex kali untuk mencari rute bfs
94              for(int i = 0; i < vertices; i++) {
95                  // jika vertex ke-i belum dikunjungi dan terdapat edge antara matrix "start" dan "i"
96                  if(!visited[i] && matrix[start][i] == 1) {
97                      visited[i] = true;  // kunjungi dan ubah nilai visited matrix i
98                      queue.add(i);       // masukkan vertex i ke queue
99                  }
100                   }   // perulangan dilanjutkan hingga semua vertex selesai dikunjungi
101             }
102             System.out.println();
103         }
104
105         // 3. Method DFS
106         public void dfs(int start) {
107             boolean visited[] = new boolean[vertices];  // menampung visited value suatu vertex
108             Stack<Integer> stack = new Stack<>();
109
110             // mengubah value pada vertex awal menjadi true karena pertama kali dikunjungi
111             visited[start] = true;
112             // memasukkan vertex awal ke stack untuk dijadikan head bfs
113             stack.push(start);
114
115             System.out.print("DFS (Head = " + start + ") : ");
116             while(stack.size() != 0) {
```

```
                // mengubah nilai variabel start menjadi nilai index teratas dari stack lalu mengeluarkan nilai tersebut dari stack
                start = stack.pop();
                // print vertex dari variabel start
                System.out.print(start + " ");

                // melakukan perulangan sebanyak jumlah vertex kali untuk mencari rute dfs
                // perulangan dilakukan dari index terbesar agar urutan dfs sesuai dengan besar valuenya
                for(int i = vertices - 1; i >= 0; i--) {
                    // jika vertex ke-i belum dikunjungi dan terdapat edge antara matrix "start" dan "i"
                    if(!visited[i] && matrix[start][i] == 1) {
                        visited[i] = true;  // kunjungi dan ubah nilai visited matrix i
                        stack.add(i);       // masukkan vertex i ke stack
                    }
                }   // perulangan dilanjutkan hingga semua vertex selesai dikunjungi
            }
            System.out.println();
        }
    }
```

Output Terminal

```
    Matrices Adjacency
  | 0   1   2   3   4   5
--+-----------------------
0 | 0   1   1   0   0   0
1 | 1   0   0   1   0   0
2 | 1   0   0   1   1   0
3 | 0   1   1   0   0   1
4 | 0   0   1   0   0   1
5 | 0   0   0   1   1   0

BFS (Head = 2) : 2 0 3 4 1 5
DFS (Head = 2) : 2 0 1 3 5 4
BFS (Head = 3) : 3 1 2 5 0 4
DFS (Head = 3) : 3 1 0 2 4 5
BFS (Head = 4) : 4 2 5 0 3 1
DFS (Head = 4) : 4 2 0 1 3 5
BFS (Head = 5) : 5 3 4 1 2 0
DFS (Head = 5) : 5 3 1 0 2 4
```