Name : Ramzy Izza Wardhana
Class : IUP CS 1
NIM : 21/472698/PA/20322

Assignment 6 Lab ASD

Complete Source Code :

https://drive.google.com/file/d/1HkYH5xPwpy4kuy28aSEKjq2wq-cPrAAS/view?usp=sharing

1. **Try implementing the adjacency matrix-version for the graph representation**

Screenshot of Code:

```java
public class AdjacencyMatrix {

    int matrix[][];//create new 2d array to represent matrix
    int node; //act as a vertex
    int bound;
    LinkedList<Integer> q = new LinkedList<Integer>(); //bfs method

    //create constructor to initialize the 2d array & vertex
    AdjacencyMatrix(int vertex){
        node = vertex;
        matrix = new int[vertex][vertex]; //n x n size
        bound = matrix.length;
    }
    //method to add path/edge from x node to y node
    void addPath(int from, int to){
        //we declare 1 as it is unweighted graph
        matrix[from][to] = 1;
        matrix[to][from] = 1;
        //declare both from and to vice versa for undirected graph
    }
    void printArray(){
        //print out the 2d array
        for(int u = 0; u < node; u++){
            for(int v  = 0; v < node; v++){
                System.out.print(matrix[u][v] + " ");
            }
            System.out.println( x: " ");
        }
    }
```

Output:

```
Input size of 2D array matrix: 6
Enter amount of value that needs to be inserted: 7

Enter the edge [from,to] that correspond to each vertex seperated by spaces:
0 1
0 2
1 3
2 3
2 4
3 5
4 5

The Graph representation in Ajdacency matrix is:
0 1 1 0 0 0
1 0 0 1 0 0
1 0 0 1 1 0
0 1 1 0 0 1
0 0 1 0 0 1
0 0 0 1 1 0
```

2. Modify the method bfs() so that it works for the adjacency matrix-version.

Screenshot of Code:

```
37      //bfs implementation on adjacency matrix
38      void bfs(int begin){
39          //create array of boolean to insert the start and visisted value
40          boolean visit[] = new boolean[bound];
41          visit[begin] = true; //declare that it is visited
42          q.add(begin);  //enqueue the start value
43          //iterate until queue empty
44          while(q.size() !=0){
45              int head = q.remove(); //pop the head of queue
46              System.out.print(head + " ");
47              //check if the value of head
48              for (int j = 1; j < bound; j++){
49                  if((matrix[head][j] == 1) && (!visit[j])){
50                    q.add(j);
51                    visit[j]=true;
52                  }
53              }
54          }
55      }
```

Output :

```
BFS Algorithm for this graph: 0 1 2 3 4 5
PS C:\Users\themi\Documents\VSCode\DFS\DFS>
```

3. Add a method named dfs() that performs Depth First Search to the graph.

Screenshot of Code:

```java
    //dfs implementation on adjacency matrix
    void dfs(boolean[] visit, int begin){
        //set the head value to true visited
        visit[begin] = true;
        //push head to the queue
        q.add(begin);
        //itterate using recursive
        for (int i = 0; i < matrix[begin].length; i++) {
            //check if it haven't visited yet & has a direct connection (edge)
            if (matrix[begin][i] == 1 && (!visit[i]))
                dfs(visit, i); //recursion
        }
        //printout the queue until queue is empty
        while(!q.isEmpty()){
            System.out.print(q.pop() + " ");
        }
    }
}
```

Output :

```
DFS Algorithm for this graph: 0 1 3 2 4 5
PS C:\Users\themi\Documents\VSCode\DFS\DFS>
```

Main.Java Screenshot:

```java
public class Main {

    Run | Debug
    public static void main(String[] args){
        //user determine the size of the adjacency matrix
        System.out.print( s: "Input size of 2D array matrix: ");
        Scanner sc = new Scanner(System.in);
        int size = sc.nextInt();
        boolean[] visited = new boolean[size];

        //create the object and declare the size of the graph
        AdjacencyMatrix matrix = new AdjacencyMatrix(size);
        //user want could insert n desired amount of data
        System.out.print( s: "Enter amount of value that needs to be inserted: ");
        int insert = sc.nextInt();

        //user input one by one for matrix u and v until n size
        System.out.println( x: "\nEnter the edge [from,to] that correspond to each vertex seperated by spaces: ");
        for(int i = 0; i <= size; i++){
            int u = sc.nextInt();
            int v = sc.nextInt();
            matrix.addPath(u,v); //insert to the array
        }
        System.out.println( );

        System.out.println( x: "The Graph representation in Adjacency matrix is: ");
        matrix.printArray();

        System.out.print( s: "\nBFS Algorithm for this graph: ");
        matrix.bfs( begin: 0);

        System.out.print( s: "\nDFS Algorithm for this graph: ");
        matrix.dfs(visited, begin: 0);
    }
}
```

Program Output Screenshot:

```
Input size of 2D array matrix: 6
Enter amount of value that needs to be inserted: 7

Enter the edge [from,to] that correspond to each vertex seperated by spaces:
0 1
0 2
1 3
2 3
2 4
3 5
4 5

The Graph representation in Ajdacency matrix is:
0 1 1 0 0 0
1 0 0 1 0 0
1 0 0 1 1 0
0 1 1 0 0 1
0 0 1 0 0 1
0 0 0 1 1 0

BFS Algorithm for this graph: 0 1 2 3 4 5
DFS Algorithm for this graph: 0 1 3 2 4 5
PS C:\Users\themi\Documents\VSCode\DFS\DFS>
```