**Assignment 0**

**Program Code:**

//\*Using DFS\*

//Program Code with DFS

**import** java.io.\*;

**import** java.util.Arrays;

**import** java.util.Scanner;

**public** **class** Assignment\_0 {

**private** **int** v,e;

**void** adjRead() **throws** IOException{

**int** i,j,l,k;

String m;

// Reading a file

FileReader fp = **new** FileReader("Graph.txt");

Scanner rd=**new** Scanner(System.*in*);

Scanner sc=**new** Scanner(fp);

m=sc.nextLine();

e=Integer.*parseInt*(m.substring(0,1));

v=Integer.*parseInt*(m.substring(2,3));

System.*out*.println("No. of edge: "+e);

System.*out*.println("No. of vertices: "+v);

**int** arr[][]=**new** **int**[v][v];

**boolean** nodeCovered[]=**new** **boolean**[v];

//Initializing the adjacency matrix with 0

**for**(i=0;i<v;i++) {

**for**(j=0;j<v;j++) {

arr[i][j]=0;

}

}

//Constructing the adjacency node

**while** (sc.hasNextLine()) {

String data = sc.nextLine();

arr[Integer.*parseInt*(data.substring(0,1))][Integer.*parseInt*(data.substring(2,3))]=1;

}

//Display the adjacency node

System.*out*.println("The adjacency matrix is: ");

**for**(i=0;i<v;i++) {

**for**(j=0;j<v;j++) {

System.*out*.print(" "+arr[i][j]);

}

System.*out*.println();

}

System.*out*.println("Enter the initiator vertex: ");

l=rd.nextInt();

System.*out*.println("The DFS Traversal is: ");

//Calling DFS function

dfs(l,nodeCovered,arr);

System.*out*.println();

k=0;

**for**(i=0;i<v;i++){

**if**(nodeCovered[i] == **true**){

//Counting the number of vertices reached from a vertex

k++;

}

}

//checking whether all the vertices are reached

**if**(v-k == 0){

System.*out*.println("Vertex "+l+" can be considered as initiator");

}

**else** {

System.*out*.println("Vertex "+l+" cannot be considered as initiator");

}

//Filling the nodeCovered array with False to

//calculate the nodes visited from the next vertex

Arrays.*fill*(nodeCovered, **false**);

sc.close();

}

**void** dfs(**int** start,**boolean**[] visited, **int** arr[][])

{

// Print the current node

System.*out*.print(start + " ");

// Set current node as visited

visited[start] = **true**;

// For every node of the graph

**for** (**int** i = 0; i < arr[start].length; i++) {

// If some node is adjacent to the current node

// and it has not already been visited

**if** (arr[start][i] == 1 && (!visited[i])) {

dfs(i, visited,arr);

}

}

}

**public** **static** **void** main(String[] args) {

Scanner sc=**new** Scanner(System.*in*);

**try**{

Assignment\_0 g1=**new** Assignment\_0();

g1.adjRead();

sc.close();

}

**catch**(IOException e1){

System.*out*.println(e1);

}

}

}

**Output:**

**Set 1:**

No. of edge: 5

No. of vertices: 5

The adjacency matrix is:

0 1 1 0 0

0 0 0 1 0

0 0 0 0 0

0 0 0 0 1

0 0 1 0 0

Enter the initiator vertex:

2

The DFS Traversal is:

2

Vertex 2 cannot be considered as initiator

**Set 2:**

No. of edge: 5

No. of vertices: 5

The adjacency matrix is:

0 1 1 0 0

0 0 0 1 0

0 0 0 0 0

0 0 0 0 1

0 0 1 0 0

Enter the initiator vertex:

0

The DFS Traversal is:

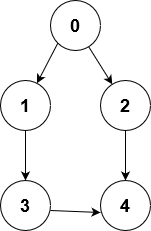
0 1 3 4 2

Vertex 0 can be considered as initiator

**Input:**

Here we are using a text file as an input to our program. The first line of the text file contains the number of edges and vertex. The next lines of the text file contains the connectivity between two vertices. If the vertex 2 and vertex 3 is connected the we write 2 3 in the text file as a source destination pair.

**Graph1 (Diagram) :**



**Graph1 (Text file representation):**

5 5

0 1

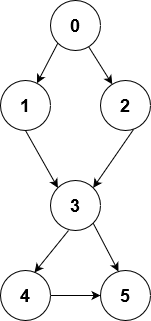
0 2

1 3

3 4

4 2

**Graph 2 (Diagram):**



**Graph1 (Text file representation):**

8 6

0 1

0 2

1 2

3 1

3 2

3 4

3 5

4 5