**PROJECT-2**

**Revanth Malay (ITCS-6114-091)  
 Student Id: 800916718**

***Problem Statement:***The description of project  2 is given in the pdf file below. Each CSV file contains an adjacency matrix of a graph. Your program should take them as inputs and complete the tasks required for this project.

Write a program to do the following tasks: 1. Read in n, and an n by n adjacency matrix representing a directed graph. 2. Determine whether the digraph contains a directed cycle. 3. If it contains a directed cycle, print one. 4. If it is a directed acyclic graph, print a topological ordering.

**Introduction:**

Any search algorithm that considers outgoing edges of a vertex before any of the vertex’s sibilings that is, outgoing edges of the vertex's predecessor in the search.

In this procedure, extremes are searched first. This is easily implemented with recursion. An algorithm that marks all vertices in a directed graph in the order they are discovered and finished.

Topological sorting for Directed Acyclic Graph (DAG) is a linear ordering of vertices such that for every directed edge uv, vertex u comes before v in the ordering. In order to run the topological Sorting, the graph should be a directed acyclic graph.

**NOTE : This code uses .xls formatted files as input, since java libraries cannot support .csv files as their input. Java gives jxl.jar library which accepts only .xls files [ 97-2003 workbook] as inputs.  
So, I have changed the files to .xls from .csv format.**

**Code:**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Please read the execute.txt file before running the code \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**package graphalgos;**

**import static graphalgos.algo\_dcg.Depth\_first\_search;**

**import java.io.File;**

**import jxl.Cell;**

**import jxl.Sheet;**

**import jxl.Workbook;**

**class algo\_dcg{ // class which contains depth first search algorithm**

**static int ans\_pre\_vertex[]= new int[50];**

**static int color\_change[]=new int[50];**

**static int curr\_time[][]=new int[50][2];**

**static int runtime;**

**static boolean check=false;**

**static int topo\_list[]=new int[50];**

**static int high=49;**

**public static void Depth\_first\_search(int a[][]){ // Method for Depth\_first\_search**

**for(int i=0;i<50;i++){**

**color\_change[i]=1;**

**ans\_pre\_vertex[i]=-1;**

**}**

**runtime=0;**

**for(int i=0;i<50;i++){**

**if(color\_change[i]==1){**

**Depth\_first\_visit(a,i);**

**}**

**}**

**}**

**public static void Depth\_first\_visit(int arrayInput[][],int vertex){ // method for Depth\_first\_visit**

**runtime=runtime+1;**

**curr\_time[vertex][0]=runtime;**

**color\_change[vertex]=2;**

**for(int i=0;i<50;i++){**

**if(arrayInput[vertex][i]==1){**

**if(color\_change[i]==1){**

**ans\_pre\_vertex[i]=vertex;**

**Depth\_first\_visit(arrayInput,i);**

**}**

**else if(color\_change[i]==2){**

**if(i!=vertex){**

**System.out.println("One");**

**System.out.print("The cycle formed is"+"\n"+vertex+"------->");**

**int m=vertex;**

**while(ans\_pre\_vertex[m]!=i){**

**System.out.print(ans\_pre\_vertex[m]+"------->");**

**m=ans\_pre\_vertex[m];**

**}**

**System.out.print(i);**

**check=true;**

**}**

**else{**

**System.out.println("\n"+"check at "+i+" to "+i);**

**}**

**}**

**}}**

**color\_change[vertex]=3;**

**topo\_list[high]=vertex;**

**high--;**

**runtime=runtime+1;**

**curr\_time[vertex][1]=runtime;**

**}**

**}**

**public class test1 { // class which has main method**

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

**int inputArray[][]=new int[50][50];**

**File f= new File("C:\\Users\\revanth\\Desktop\\graph1.xls"); // location of file to be mentioned here**

**Workbook wb=Workbook.getWorkbook(f);**

**Sheet s=wb.getSheet(0);**

**int rows=s.getRows();**

**int columns=s.getColumns();**

**for(int i=0;i<rows;i++){**

**for(int j=0;j<columns;j++)**

**{**

**Cell cell = s.getCell(j, i);**

**inputArray[i][j]=Integer.parseInt(cell.getContents()); // values from the graph are stored here in the array**

**}**

**}**

**Depth\_first\_search(inputArray); // calling the method Depth\_first\_search in class algo\_dcg**

**try {**

**if(!(algo\_dcg.check)){**

**System.out.println("Topological ordering for the graph is : ");**

**for(int n=0;n<50;n++){**

**System.out.print(algo\_dcg.topo\_list[n]+" "); // prints the vertices for topological ordering**

**}**

**}**

**}**

**catch(Exception e){**

**System.out.println(e);**

**}**

**}**

**}**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Please read the execute.txt file before running the code \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Analysis of the code:**

The above code take’s input from the graph file and store’s the values in a n x n array. This is given to the DFS method in order to check the matrix has a directed cyclic graph or not. If the matrix has a directed cyclic graph the output will be displayed as “one”.

If the graph is not a directed cyclic one, the input array is then passed to the Topological sorting method, where the output will be an ordering of the vertices.

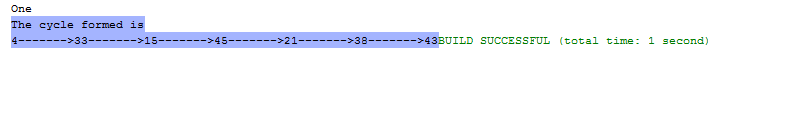
**Result:**

1. If the input is graph1.xls Output is :

**One**

**The cycle formed is**

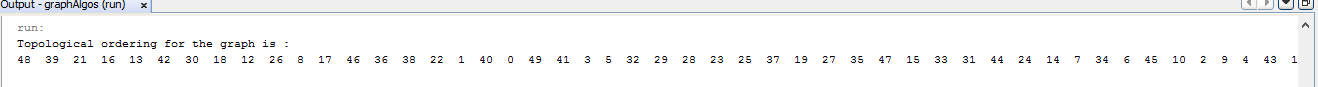
**4------->33------->15------->45------->21------->38------->43**



1. If the input is graph2.xls, Output is :

**Topological ordering for the graph is :**

**48 39 21 16 13 42 30 18 12 26 8 17 46 36 38 22 1 40 0 49 41 3 5 32 29 28 23 25 37 19 27 35 47 15 33 31 44 24 14 7 34 6 45 10 2 9 4 43 11 20.**



**Complexity of the Algorithm:**The above algorithm is simply DFS algorithm. So time complexity is same as DFS which is O(V+E) where e represents edges and v represents vertices of the graph.  
Topological sorting algorithm is also similar to the DFS with an extra stack. The time complexity is O(V+E) e represents edges and v represents vertices of the graph.

**Note: DFS algorithm has been referenced from Introduction to Algorithms 3rd edition Thomas.R.Corman**