MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

NATIONAL TECHNICAL UNIVERSITY

«KHARKIV POLYTECHNIC INSTITUTE»

Department of Software Engineering and Management Information Technologies

Report from lab № 6

Discipline «Algorithm and Data Structures»

Executed by:

OUSSAMA BARKANE

Student Group:

1. KH201.8ei2

Checked by:

Prof M.O. Bilova

Kharkiv

2019

**Theme**: FUNDAMENTAL ALGORITHMS ON GRAPHS

AND TREES (2 PAIRS)

**Objective:** explore ways of representing graphs and gain skills of programming

algorithms that process them.

**Task :**

Develop a program that reads from the keyboard numbers N, M (1 <N, M<256) being numbers of vertices and edges of the graph, a sequence of M integerpairs being the edges. The program saves the graph and performs on it an algorithm.Representing graphs.

1 Adjacency matrix.

2 Adjacency lists.

Algorithms.

1 Depth-first search. Display the order of vertex traversal. Show time of

arrival and parent in the tree traversal for each one.

2 Depth-first search. Display the order of vertex traversal. Show time of

arrival and leaving and parent in the tree traversal for each one.

3 Topological sorting. Display the same data as for DFS and sorting results.

4 Determine whether a given graph is a tree or a forest.

5 Construct a spanning tree with Prim's algorithm.

6 Construct a spanning tree with Kruskal' algorithm.

**Theory :**

in computer science, Topological Sorting. 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. Topological Sorting for a graph is not possible if the graph is not a DAG

**Progress of the Lab**

**CODE :**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

using namespace std;

int main() {

int i, j, k, n, a[10][10], indeg[10], flag[10], count = 0;

printf("Enter the no of vertices:\n");

scanf("%d", &n);

printf("Enter the adjacency matrix:\n");

for (i = 0; i < n; i++) {

printf("Enter row %d\n", i + 1);

for (j = 0; j < n; j++)

scanf("%d", &a[i][j]);

}

for (i = 0; i < n; i++) {

indeg[i] = 0;

flag[i] = 0;

}

for (i = 0; i < n; i++)

for (j = 0; j < n; j++)

indeg[i] = indeg[i] + a[j][i];

printf("\nThe topological order is:");

while (count < n) {

for (k = 0; k < n; k++) {

if ((indeg[k] == 0) && (flag[k] == 0)) {

printf("%d ", (k + 1));

flag[k] = 1;

}

for (i = 0; i < n; i++) {

if (a[i][k] == 1)

indeg[k]--;

}

}

count++;

}

return 0;

}

**Code n 2**

#include <iostream>

#include <conio.h>

using namespace std;

struct node\_info

{

int no;

int lv\_time, st\_time;

}\*q = NULL, \*r = NULL;

struct node

{

node\_info \*pt;

node \*next;

}\*top = NULL, \*p = NULL, \*np = NULL;

int c = 0;

void push(node\_info \*ptr)

{

np = new node;

np->pt = ptr;

np->next = NULL;

if (top == NULL)

{

top = np;

}

else

{

np->next = top;

top = np;

}

}

node\_info \*pop()

{

if (top == NULL)

{

cout << "underflow\n";

}

else

{

p = top;

top = top->next;

return(p->pt);

delete(p);

}

}

void topo(int \*v, int am[][4], int i)

{

q = new node\_info;

q->no = i;

q->st\_time = c;

cout << "start time for node no " << q->no << ":" << c << endl;

push(q);

v[i] = 1;

for (int j = 0; j < 4; j++)

{

if (am[i][j] == 0 || (am[i][j] == 1 && v[j] == 1))

continue;

else if (am[i][j] == 1 && v[j] == 0)

{

c++;

topo(v, am, j);

}

}

c++;

r = pop();

cout << "leave time for " << r->no << ":" << c << endl;

return;

}

int main()

{

int v[4], am[4][4];

for (int i = 0; i < 4; i++)

v[i] = 0;

for (int i = 0; i < 4; i++)

{

cout << "enter the values for adjacency matrix row:" << i + 1 << endl;

for (int j = 0; j < 4; j++)

{

cin >> am[i][j];

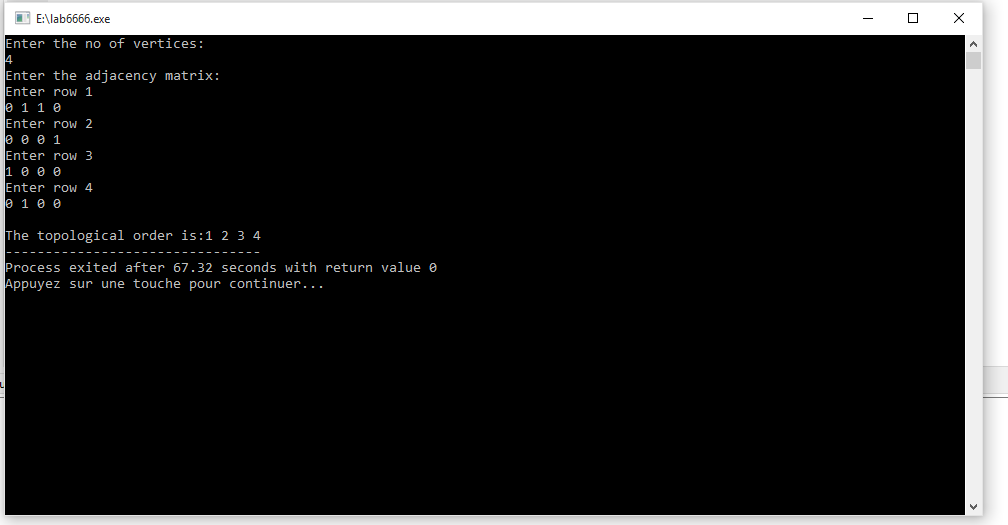
}

}

topo(v, am, 0);

\_getch();

}



**Conclusion :**

During this laboratory assignment, the knowledge of knowing and applying how to explore ways of representing graphs and gain skills of programming