Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра інформаційних систем та мереж

ЗВІТ

про виконання лабораторних робіт №8

**“Графи. Обхід графу. Пошук”**

з дисципліни "Алгоритми та структури даних ”

В-15

Виконала студентка групи СА-32

Катеринчук Тетяна Андріївна

Прийняв доцент

Щербак С.С.

*2017*

**Мета роботи:** набуття практичних навичок програмування графів.

**Завдання:**

Розробити алгоритм знаходження зв’язних підграфів заданого графа.

**Текст програми**

Main.cpp

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <stdio.h>

#include <string>

#include <vector>

using namespace std;

bool EntrancTopGraph(vector<vector<int>> boundSubgraph, int k);

void MatrixAdjacencySupgraph(vector<vector<int>> matrixAdjacency, vector<vector<int>> boundSubgraph, int n);

void OutputMatrixAdjacencySupgraph(vector<vector<int>> boundSubgraph, vector<vector<int>> tempSubgraph, int nSubgraph);

void InputMatrixAdjacency(vector<vector<int>>& matrixAdjacency,int n) {

cout << "Введiть матрицю сумiжностi\n\n";

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

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

int arc;

cin >> arc;

matrixAdjacency[i].push\_back(arc);

}

matrixAdjacency.push\_back(vector<int>());

}

}

void OutputMatrixAdjacency(vector<vector<int>> matrixAdjacency, int n) {

cout << "Матриця сумiжностi: "<<endl<<" ";

for (int i = 1; i <= n; i++) {

cout <<" "<< i;

}

cout << endl<<" ";

for (int i = 1; i <= n; i++) {

cout << "\_\_";

}

cout << endl;

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

cout << i + 1 << " |";

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

cout <<matrixAdjacency[i][j]<<" ";

}

cout <<"|"<< endl;

}

cout << " ";

for (int i = 1; i <= n; i++) {

cout << "\_\_";

}

cout << endl;

}

void SearchBoundSubgraph(vector<vector<int>> matrixAdjacency, int n) {

vector<vector<int>> boundSubgraph;

int nSupgraph = 0;

boundSubgraph.push\_back(vector<int>());

boundSubgraph[nSupgraph].push\_back(0);

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

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

if (i != j) {

if (matrixAdjacency[i][j] > 0) {

if (EntrancTopGraph(boundSubgraph, i)) {

if(!EntrancTopGraph(boundSubgraph, j))

boundSubgraph[nSupgraph].push\_back(j);

}

else {

nSupgraph++;

boundSubgraph.push\_back(vector<int>());

boundSubgraph[nSupgraph].push\_back(j);

}

}

}

}

}

MatrixAdjacencySupgraph(matrixAdjacency, boundSubgraph, n);

}

void MatrixAdjacencySupgraph(vector<vector<int>> matrixAdjacency, vector<vector<int>> boundSubgraph,int n) {

for (int i = 0; i < boundSubgraph.size(); i++) {

for (int repeat\_counter = 0; repeat\_counter < boundSubgraph[i].size(); repeat\_counter++)

{

int temp = boundSubgraph[i][0];

for (int element\_counter = repeat\_counter + 1; element\_counter < boundSubgraph[i].size(); element\_counter++)

{

if (boundSubgraph[i][repeat\_counter] > boundSubgraph[i][element\_counter])

{

temp = boundSubgraph[i][repeat\_counter];

boundSubgraph[i][repeat\_counter] = boundSubgraph[i][element\_counter];

boundSubgraph[i][element\_counter] = temp;

}

}

}

}

cout << endl;

for (int i = 0; i < boundSubgraph.size(); i++) {

vector<vector<int>> tempSubgraph;

for (int j = 0; j < boundSubgraph[i].size(); j++) {

tempSubgraph.push\_back(vector<int>());

for (int k = 0; k < boundSubgraph[i].size();k++) {

tempSubgraph[j].push\_back(matrixAdjacency[boundSubgraph[i][j]][boundSubgraph[i][k]]); }

}

OutputMatrixAdjacencySupgraph(boundSubgraph, tempSubgraph, i);

cout << endl;

}

}

void OutputMatrixAdjacencySupgraph(vector<vector<int>> boundSubgraph, vector<vector<int>> tempSubgraph, int nSubgraph) {

cout << "Матриця сумiжностi "<<nSubgraph+1<<" зв'язного пiдграфа даного графа: \n\n ";

for (int i = 0; i < tempSubgraph.size(); i++) {

cout << " " << boundSubgraph[nSubgraph][i]+1;

}

cout << endl << " ";

for (int i = 0; i < tempSubgraph.size(); i++) {

cout << "\_\_";

}

cout << endl;

for (int i = 0; i < tempSubgraph.size(); i++) {

cout << boundSubgraph[nSubgraph][i] + 1 << " |";

for (int j = 0; j < tempSubgraph.size(); j++) {

cout << tempSubgraph[i][j] << " ";

}

cout << "|" << endl;

}

cout << " ";

for (int i = 0; i < tempSubgraph.size(); i++) {

cout << "\_\_";

}

cout << endl;

}

bool EntrancTopGraph(vector<vector<int>> boundSubgraph,int k) {

for (int i = 0; i < boundSubgraph.size(); i++) {

for (int j = 0; j < boundSubgraph[i].size(); j++) {

if (boundSubgraph[i][j] == k) {

return true;

}

}

}

return false;

}

int main() {

setlocale(LC\_CTYPE, "ukr");

int n=6;

cout << "Введiть кiлькiсть вершин графа: "; cin >> n;

vector<vector <int>> matrixAdjacency(n);

InputMatrixAdjacency(matrixAdjacency,n);

OutputMatrixAdjacency(matrixAdjacency,n);

SearchBoundSubgraph(matrixAdjacency, n);

system("pause");

return 0;

}

**Результат програми**

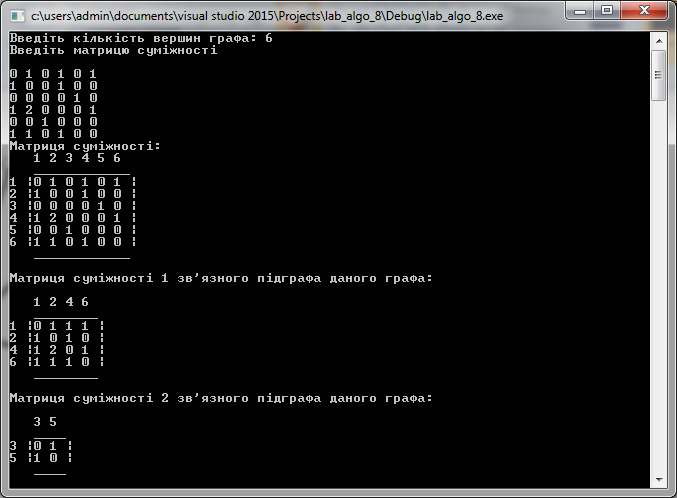


Рис.1 Розбиття графа на зв’язні підграфи

Граф, по якому вручну будувалася матриця суміжності у даній реалізації програми

**6 2 3**

**1 4 5**

**Висновок:** При виконання даної лабораторної роботи я набула навичок програмування графів. Створила програму яка по заданій матриці суміжності будує матриці суміжності зв’язних підграфів.