Міністерство освіти і науки України

Черкаський державний технологічний університет

Кафедра програмного забезпечення автоматизованих систем

Звіт

про виконання лабораторної роботи №5

з дисципліни «Проектний практикум»

|  |  |
| --- | --- |
| Перевірив:  Асистент кафедри ПЗАС  Півень О.Б. | Виконав:  студент 3-го курсу  групи ПЗ-1644  Близнюк А. О. |

Черкаси 2017

**Тема роботи:** Графи. Пошук в ширину.

**Мета роботи**: Ознайомлення з графами та методом пошуку в ширину.

**Теоретичні відомості**

Граф — це сукупність об'єктів із зв'язками між ними.

Об'єкти розглядаються як вершини, або вузли графу, а зв'язки — як дуги, або ребра. Для різних областей використання види графів можуть відрізнятися орієнтовністю, обмеженнями на кількість зв'язків і додатковими даними про вершини або ребра.

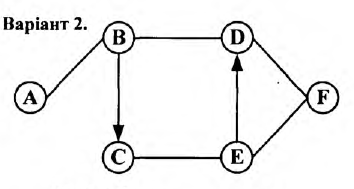
Велика кількість структур, які мають практичну цінність в [математиці](https://uk.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0) та [інформатиці](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0), можуть бути представлені графами.

Якщо задано граф G = (V, E) та початкову вершину s, алгоритм пошуку в ширину систематично обходить всі досяжні із s вершини. На першому кроці вершина s позначається, як пройдена, а в список додаються всі вершини, досяжні з s без відвідування проміжних вершин. На кожному наступному кроці всі поточні вершини списку відмічаються, як пройдені, а новий список формується із вершин, котрі є ще не пройденими сусідами поточних вершин списку. Для реалізації списку вершин найчастіше використовується [черга](https://uk.wikipedia.org/wiki/%D0%A7%D0%B5%D1%80%D0%B3%D0%B0_(%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85)). Виконання алгоритму продовжується до досягнення шуканої вершини або до того часу, коли на певному кроці в список не включається жодна вершина. Другий випадок означає, що всі вершини, доступні з початкової, уже відмічені, як пройдені, а шлях до цільової вершини не знайдений.

Алгоритм має назву пошуку в ширину, оскільки «фронт» пошуку (між пройденими та непройденими вершинами) одноманітно розширюється вздовж всієї своєї ширини. Тобто, алгоритм проходить всі вершини на відстані k перед тим як пройти вершини на відстані k+1.

**Завдання:**

Створити програму, яка реалізує пошук найкоротшого шляху між двома довільними вершинами графа згідно варіанту. Номери вершин для пошуку шляху між ними користувач повинен ввести самостійно. Також потрібно ревлізувати чергу у вигляді динамічної структури даних. Для отримання оцінки добре забезпечити знаходження шляху у випадку зваженого графа. Для отримання оцінки відмінно реалізувати граф за допомогою списква суміжності.



#include <iostream>

#include <cstdlib>

using namespace std;

const int n1 = 5;

int Graph\_metodichka[n1][n1] = {

{0,0,0,0,0},

{0,0,1,1,0},

{0,0,0,1,1},

{0,1,0,0,1},

{0,0,1,0,0}

};

const int n2 = 7;

int Graph\_my\_variant[n2][n2]={{0,0,0,0,0,0,0},//0

{0,0,1,0,0,0,0},//1

{0,1,0,1,1,0,0},//2

{0,0,0,0,0,1,0},//3

{0,0,1,0,0,0,1},//4

{0,0,0,1,1,0,1},//5

{0,0,0,0,1,1,0}};//6\*/

const int n = n2;

int Graph[n][n];

void insertG(){

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

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

Graph[i][j] = Graph\_my\_variant[i][j];

}

int mark[n]={0};

int cf = 0; //cf1=0;

int all\_path[n];

int d[n];

int q[n];

int last=0;

int last\_p=0;

int start = 0;

int finish = 6;

//int finish = 4;

void push(int x){

// if(last!=n){

if(last!=finish){

last++;

q[last] = x;

}

else cout<<"Error!\n";

}

int pop(){

if(last>-1){

return q[last--];

}

else {return -1;}

}

bool empty(){

if(last == -1) { return true;}

else {return false;}

}

void BFS(){

push(start);

d[start] = 0;

mark[start]=1;

while(!empty()){

int v = pop();

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

if(Graph[v][i] != 0){

if(mark[i] == 0){

cf++;

d[i] = d[v]+1;

all\_path[cf] = i;

cout<<"["<<d[i]<<"] ";

// cout<<"\ni = "<<i;

mark[i] = 1;

push(i);

}

}

}

}

}

void printGraph(){

for(int i=0;i<n;i++){ cout<<"\t";

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

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

}

cout<<endl;

}

}

int main(){

setlocale(LC\_ALL, "Rus");

insertG();

while(true){

cout<<"Start: "; cin>>start;

cout<<"Finish: "; cin>>finish;

if(start > finish) {

int buf\_start = start;

start = finish;

finish = buf\_start;

}

if(start<1 || start>n || finish<1 || finish>n) {

cout<<"\nNevirni koordunaty!\n\n";

continue;

}

else {

last=0;

last\_p=0;

cf = 0;

///////////////////////////// path short to finish //////////////

if((start == 2 || start == 3) && (finish == 6)) {

if(start %2 == 0) {

cout << "\n\nMin way: [" << start << "] -> [" << start+2 << "] -> [" << finish << "]";

}

else {

cout << "\n\nMin way: [" << start << "] -> [" << start+2 << "] -> [" << finish << "]";

}

}

else if((start == 4 || start == 5) && (finish == 6)) {

cout << "\n\nMin way: [" << start << "] -> [" << finish << "]";

}

else if((start == 1) && (finish == 6)) {

cout << "\n\nMin way: [" << start << "] -> [2] -> [4] -> [" << finish << "]";

}

else if((start == 1) && (finish == 2 || finish == 4)) {

if(finish == 2)

cout << "\n\nMin way: [" << start << "] -> [" << finish << "]";

else

cout << "\n\nMin way: [" << start << "] -> [2] -> [" << finish << "]";

}

else if((start == 1) && (finish == 3 || finish == 5)) {

if((finish - 2) >= 2) {

cout << "\n\nMin way: [" << start << "] -> [2] -> [" << finish-2 << "] -> [" << finish << "]";

}

else {

cout << "\n\nMin way: [" << start << "] -> [2] -> [" << finish << "]";

}

}

else if((start == 2 || start == 4 ) && (finish == 3 || finish == 5 || finish == 4)) {

if(finish == 4) {

cout << "\n\nMin way: [" << start << "] -> [3] -> [5] -> [" << finish << "]";

}

else if((start + 1) == finish)

cout << "\n\nMin way: [" << start << "] -> [" << finish << "]";

else

cout << "\n\nMin way: [" << start << "] -> [" << start+1 <<"] -> [" << finish << "]";

}

else if(start == 3 && finish == 4) {

cout << "\n\nMin way: [" << start << "] -> [5] -> [" << finish << "]";

}

else if(start!=finish) cout<<"\n\nKorotkux wlyaxiv > 1. Potribna povna variaciya.\n";

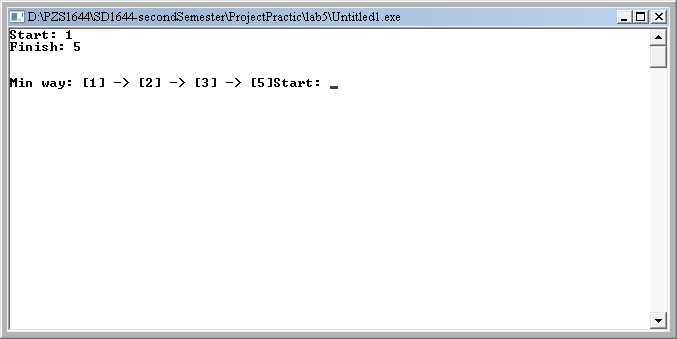
}

// cout<<"\n\ncf1 = "<<cf1<<endl;

}

return 0;

}



Висновки: Під час лабораторної роботи було освоєно навички роботи з графами та алгоритм пошуку в ширину.