ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет Информационных технологий
Кафедра Информатики и информационных технологий

направление подготовки
09.03.02 «Информационные системы и технологии»

ЛАБОРАТОРНАЯ РАБОТА № 7

Дисциплина: «Основы современных алгоритмов»

Тема: «Поиск кратчайшего расстояния на не взвешенном графе (поиск в ширину)»

Выполнил: студент группы 211-723

Сергеев Станислав Олегович

Дата, подпись_____ _____
(Дата) (Подпись)

Проверил:_____ _____
(Фамилия И.О., степень, звание) **(Оценка)**

Дата, подпись_____ _____
(Дата) (Подпись)

Замечания: _____

_____

_____

Москва

2022

Поиск кратчайшего расстояния на не взвешенном графе (поиск в ширину)

Цель:

Получить знания и практические навыки в решении задач обхода графа и поиска кратчайшего расстояния средствами языка С.

Постановка задачи:

Написать программу поиска в ширину, реализующую следующий алгоритм поиска пути на графе между двумя вершинами. Результат выдавать перечислением номеров вершин.

Поиск в ширину

```
#include <iostream>
#include <iomanip>
#include <vector>
#define SIZE 10 //Количество вершин в графе
using namespace std;

struct item { //Структура элемента
	int data;
	item* next;
	item* prev;
	item() : data(NULL), next(nullptr), prev(nullptr){}
};

class Matrix
{
private:
	int    graph[SIZE][SIZE]; //граф
	item* first; //Адрес первого элемента списка
	item* last; //Адрес последнего элемента списка
	int count; //Количество элементов в списке

public:
	Matrix()
	{
		int b[SIZE][SIZE] = {
		{0, 1, 1, 0, 1, 0, 0, 0, 1, 1},
		{0, 0, 1, 1, 0, 1, 0, 0, 1, 0},
		{0, 1, 0, 0, 1, 0, 0, 0, 1, 0},
		{1, 0, 1, 0, 1, 1, 1, 1, 0, 0},
		{0, 1, 0, 0, 0, 1, 0, 0, 0, 0},
		{0, 0, 0, 1, 1, 0, 1, 1, 0, 1},
		{1, 1, 0, 1, 0, 0, 0, 0, 1, 0},
		{0, 1, 0, 1, 0, 1, 1, 0, 1, 0},
		{0, 1, 1, 0, 1, 0, 0, 0, 0, 1},
		{0, 0, 1, 1, 0, 1, 1, 1, 0, 0},
		};
		for (int i = 0; i < SIZE; i++)
			for (int j = 0; j < SIZE; j++)
				graph[i][j] = b[i][j];
		last = nullptr;
		first = nullptr;
		count = 0;
	}

	void AddToMatrix(int _i, int _j, int _temp)
	{
		graph[_j][_i] = _temp;
		if (_i == _j) graph[_j][_i] = 0;
```

```
}

int GetFromMatrix(int _i, int _j)
{
	return graph[_i][_j];
}

void pushBack(int a) //Добавление элемента в конец списка
{
	item* newitem = new item;
	newitem->data = a;
	if (first == nullptr)
		first = newitem;
	newitem->next = nullptr;
	newitem->prev = last;
	if (last != nullptr)
		last->next = newitem;
	last = newitem;
	count++;
}

void del_first() //Удаление елемента из начала очереди
{
	item* place = first;
	if (place != last) //Если элемент первый, но НЕ последний
	{
		place->next->prev = nullptr;
		first = place->next;
	}
	else //Если элемент первый и последний
	{
		first = nullptr;
		last = nullptr;
	}
	delete place;
	count--;
}

bool empty() //Возвращает true если очередь пустая и false если нет
{
	bool empt;
	if (first == nullptr)
		empt = true;
	else
		empt = false;
	return empt;
}

int first_data() //Возвращает значение из начала очереди
{
	return first->data;
}

vector <int> BFS(int _start, int _end)
{
	bool visited[SIZE]; //Массив пройденных врешин
	int ways[SIZE]; //Массив родительских вершин
	for (int i = 0; i < SIZE; i++)
		visited[i] = 0;
	pushBack(_start);

	visited[_start] = 1;
	ways[_start] = -1;

	while (!empty())
	{
		int unit = first_data();
		del_first();
		for (int i = 0; i < SIZE; i++)
```

```
                        {
                                if ((visited[i] == 0) && (graph[unit][i] == 1))
                                {
                                        pushBack(i);
                                        ways[i] = unit;
                                        visited[i] = 1;
                                }
                        }
                }
                int dne = _end;
                int _count = 0;
                do {
                        _count++;
                        dne = ways[dne];
                } while (dne != ways[_start]);
                int* rev = new int[_count];
                int i = 0;
                rev[i] = _end;
                i++;
                do {
                        rev[i] = ways[_end];
                        i++;
                        _end = ways[_end];
                } while (_end != _start);
                i--;

                vector <int> ForOutput;
                while (i >= 0)
                {
                        ForOutput.push_back(rev[i]);
                        i--;
                }
                return ForOutput;
        }
};

Matrix CreateMatrix(Matrix m)
{
        cout << "\nВведите матрицу 10 на 10: " << endl;
        int temp = 0;
        for (int j = 0; j < SIZE; j++)
                for (int i = 0; i < SIZE; i++)
                {
                        cin >> temp;
                        m.AddToMatrix(i, j, temp);
                }
        return m;
}

void OutputMatrix(Matrix m)
{
        for (int i = 0; i < SIZE; i++)
        {
                for (int j = 0; j < SIZE; j++)
                        cout << setfill(' ') << setw(5) << m.GetFromMatrix(i, j);
                cout << endl;
        }
}

void OutputWay(vector <int> _vector)
{
        cout << "Путь: ";
        for (unsigned i = 0; i < _vector.size(); i++)
        {
                cout << _vector[i]+1 << " ";
        }
        cout << endl;

}
```

```cpp
int main()
{
	setlocale(LC_ALL, "");
	Matrix M;
	int start, end, choice;
	cout << "Будете вводить собственную матрицу(1) или взять готовую(0)?: ";
	cin >> choice;
	if (choice)
		M = CreateMatrix(M);
	OutputMatrix(M);
	while (true)
	{
		cout << "Из какой вершины начинается путь?: ";
		cin >> start;
		cout << "До какой точки рассчитать маршрут?: ";
		cin >> end;
		if ( (start == end) || (start > SIZE) || (end > SIZE) ) cout << "Ошибка!\n";
		else
			OutputWay(M.BFS(start - 1, end - 1));
	}
	return 0;
}
```

```
D:\МПУ\VisualStudio\Поиск в ширину\Debug\Поиск в ширину.exe
Будете вводить собственную матрицу(1) или взять готовую(0)?: 0
    0    1    1    0    1    0    0    0    1    1
    0    0    1    1    0    1    0    0    1    0
    0    1    0    0    1    0    0    0    1    0
    1    0    1    0    1    1    1    1    0    0
    0    1    0    0    0    1    0    0    0    0
    0    0    0    1    1    0    1    1    0    1
    1    1    0    1    0    0    0    0    1    0
    0    1    0    1    0    1    1    0    1    0
    0    1    1    0    1    0    0    0    0    1
    0    0    1    1    0    1    1    1    0    0
Из какой вершины начинается путь?: 3
До какой точки рассчитать маршрут?: 9
Путь: 3 9
Из какой вершины начинается путь?: 4
До какой точки рассчитать маршрут?: 9
Путь: 4 1 9
Из какой вершины начинается путь?: 5
До какой точки рассчитать маршрут?: 3
Путь: 5 2 3
Из какой вершины начинается путь?:
```