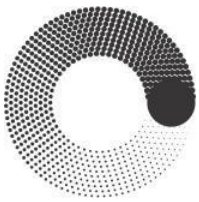
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ



МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет Информационных технологий
Кафедра Информатики и информационных технологий

направление подготовки
09.03.02 «Информационные системы и технологии»

ЛАБОРАТОРНАЯ РАБОТА № 8

Дисциплина: «Основы современных алгоритмов»

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

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 глубину).

Цель:

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

Постановка задачи:

1) Написать программу поиска в глубину с использованием стека, реализующую приведенный алгоритм поиска пути на графе между двумя вершинами. Результат выдавать перечислением номеров вершин.
2) Написать программу поиска в глубину с использованием рекурсии, реализующую приведенный алгоритм поиска пути на графе между двумя вершинами. Результат выдавать перечислением номеров вершин.

Поиск в глубину через стек.

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


class Matrix
{
private:
	int	graph[SIZE][SIZE]; //граф
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];
	}
```

```
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];
}




vector <int> DFS(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;
	stack <int> s1;
	visited[_start] = 1;
	ways[_start] = -1;
	s1.push(_start);
	while (!s1.empty())
	{
		int unit = s1.top();
		s1.pop();
		visited[unit] = 1;
		for (int i = 0; i < SIZE; i++)
		{
			if ((visited[i] == 0) && (graph[unit][i] == 1))
			{
				s1.push(i);
				ways[i] = unit;
			}
		}
	}
	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;


}


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.DFS(start - 1, end - 1));
        }
        return 0;
}
```

```
Будете вводить собственную матриц(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 ра?считать маршрут?: 7
Путь: 3 9 10 8 7
Из какой вершины начинается путь?:
```

Поиск в глубину через рекурсию

```
#include <iostream>

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


class Matrix
{
private:
	int	graph[SIZE][SIZE]; //граф
	int ways[SIZE]; //Массив родительских вершин
	bool visited[SIZE]; //Массив пройденных врешин
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];
		for (int i = 0; i < SIZE; i++)
			ways[i] = 0;
		for (int i = 0; i < SIZE; i++)
			visited[i] = 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 DFSoriginal(int _val)
{
	visited[_val] = 1;
	for (int i = 0; i < SIZE; i++)
	{
		if ((visited[i] == 0) && (graph[_val][i] == 1))
		{
			ways[i] = _val;
			DFSoriginal(i);
		}
	}
}


void ClearVisitedAndWays()
{
	for (int i = 0; i < SIZE; i++)
		ways[i] = 0;
	for (int i = 0; i < SIZE; i++)
		visited[i] = 0;
}


vector <int> DFS(int _start, int _end)
{
	ways[_start] = -1;
	DFSoriginal(_start);
	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--;

	ClearVisitedAndWays();


	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;


}


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.DFS(start - 1, end - 1));
	}
	return 0;
}
```

```
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?: 7
Путь: 3 2 4 1 5 6 7
Из какой вершины начинается путь?:
```