Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
*«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)*

ФАКУЛЬТЕТ **ИУК «Информатика и управление»**

КАФЕДРА **ИУК4 «Программное обеспечение ЭВМ, информационные технологии»**

ЛАБОРАТОРНАЯ РАБОТА №3

«Реализация основных алгоритмов с графами»

ДИСЦИПЛИНА: «Типы и структуры данных»

Выполнил: студент гр. ИУК4-31Б _____ (Суриков Н. С.)
(подпись) (Ф.И.О.)

Проверил: _____ (Былинка М. И.)
(подпись) (Ф.И.О.)

Дата сдачи (защиты): 05.11.12

Результаты сдачи (защиты):

- Балльная оценка: 95
- Оценка:

Калуга, 2024

Цель: формирование практических навыков создания алгоритмов обработки графов.

Задачи:

1. Познакомиться со способами представления графов в памяти компьютера.
2. Изучить основные обходы графов.
3. Научиться составлять алгоритмы для нахождения кратчайших путей в графе.
4. Реализовать алгоритм согласно варианту.

Вариант 25

Формулировка задания:

Используя нерекурсивную процедуру поиска в глубину, найти города, расположенные от города B на расстоянии, большем T.

Листинг программы:

`main.cpp`

```cpp
#include "Graph.h"
#include <iostream>

int main()
{
    Graph graph("input.txt");

    char startCity;
    int threshold;
    std::cout << "Введите начальный город (A-E) и порог расстояния: ";
    std::cin >> startCity >> threshold;

    graph.findCitiesBeyondDistance(startCity, threshold);

    return 0;
}
```

Graph.cpp

```cpp
#include "Graph.h"
#include <fstream>
#include <iostream>
#include <stack>

```

```cpp
Graph::Graph(const std::string &filename)
{
    std::ifstream file(filename);
    if (!file.is_open())
    {
        std::cerr << "Ошибка открытия файла!" << std::endl;
        return;
    }

    file >> numCities;
    adjacencyMatrix.resize(numCities, std::vector<int>(numCities, 0));

    for (int i = 0; i < numCities; ++i)
    {
        for (int j = 0; j < numCities; ++j)
        {
            file >> adjacencyMatrix[i][j];
        }
    }
    file.close();
}

int Graph::cityIndex(char city)
{
    return city - 'A';
}

char Graph::indexCity(int index)
{
    return 'A' + index;
}

void Graph::dfs(int startCity, int threshold)
{
    std::vector<bool> visited(numCities, false);
    std::stack<std::pair<int, int>> stack; // (city, currentDistance)
    stack.push({startCity, 0});
    visited[startCity] = true;

    while (!stack.empty())
    {
        auto [city, currentDistance] = stack.top();
        stack.pop();

        for (int i = 0; i < numCities; ++i)
        {
            if (adjacencyMatrix[city][i] > 0)
            {
                int newDistance = currentDistance + adjacencyMatrix[city][i];

                if (newDistance > threshold && !visited[i])
                {
                    std::cout << indexCity(i) << " ";
                }

```

```
                if (!visited[i])
                {
                    visited[i] = true;
                    stack.push({i, newDistance});
                }
            }
        }
    }
}

void Graph::findCitiesBeyondDistance(char startCity, int threshold)
{
    int startIndex = cityIndex(startCity);
     std::cout << "Города на расстоянии больше " << threshold << " от города "
<< startCity << ": ";
    dfs(startIndex, threshold);
    std::cout << std::endl;
}
```

Graph.h

```
#ifndef GRAPH_H
#define GRAPH_H

#include <string>
#include <vector>

class Graph
{
public:
    Graph(const std::string &filename);
    void findCitiesBeyondDistance(char startCity, int threshold);

private:
    std::vector<std::vector<int>> adjacencyMatrix;
    int numCities;
    void dfs(int startCity, int threshold);
    int cityIndex(char city);
    char indexCity(int index);
};

#endif // GRAPH_H
```

Результат работы:

```
Матрица смежности
5
0 10 0 30 100
10 0 50 0 0
0 50 0 20 10
30 0 20 0 60
100 0 10 60 0
```

```
Введите начальный город (A-Z) и порог расстояния: A 30
Города на расстоянии больше 30 от города A: E C

Введите начальный город (A-Z) и порог расстояния: A 10
Города на расстоянии больше 10 от города A: D E C

Введите начальный город (A-Z) и порог расстояния: A 130
Города на расстоянии больше 130 от города A:
```

Вывод: в ходе работы были сформированы практические навыки создания алгоритмов обработки графов.