|  |  |
| --- | --- |
| https://www.mirea.ru/bitrix/templates/unlimtech/images/logo.png | |
| МИНОБРНАУКИ РОССИИ | |
| Федеральное государственное бюджетное образовательное учреждениевысшего образования"МИРЭА - Российский технологический университет"РТУ МИРЭА | |
| Институт информационных технологий | |
| Кафедра математического обеспечения и стандартизации ИТ | |

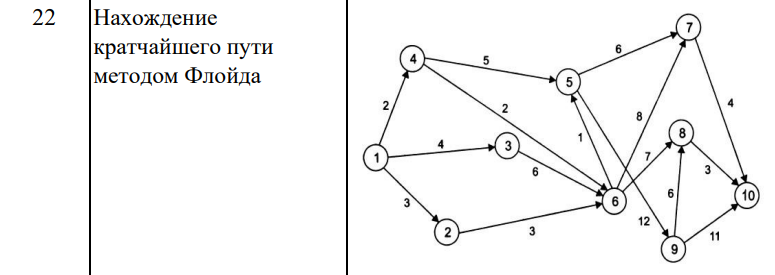
|  |  |
| --- | --- |
| **ОТЧЕТ**  **ПО ПРАКТИЧЕСКОЙ РАБОТЕ №2.5: Основные алгоритмы работы с графами** | |
| **ПО ДИСЦИПЛИНЕ** | |
| **«**СТРУКТУРЫ И АЛГОРИТМЫ ОБРАБОТКИ ДАННЫХ**»** | |
|  | |
| Выполнил студент группы ИКБО-01-21 | Кузнецов А.А. |
| Принял старший преподаватель | Туманова М.Б. |
|  |  |

Москва 2022

**Цель:** Составить программу создания графа и реализовать процедуру для работы с графом, определенную индивидуальным вариантом задания.

# Ход работы

**Индивидуальный вариант** (22):



**Модель решения:**

Алгоритм Флойда-Уоршелла представляет собой простой перебор всех путей и выбор из них наименьшего. Перебор осуществляется по так называемой «матрице смежности» размера NxN, где N — количество вершин графа.

На пересечении i-ой строки и j-го столбца матрицы стоит значение веса ребра из вершины i в вершину j. Главная диагональ матрицы смежности — это всегда нули, потому что ребер из i-ой вершины в i-ую в графе быть не должно. В том случае, когда между вершинами нет ребра, в матрице будет стоять условная бесконечность (INF).

Реализуется поиск наименьшего расстояния последовательным перебором всех путей (функция – floydWarshall). В три цикла, внешний выбирает вершину, через которую мы пробуем найти лучший путь. Внутренние два перебирают пары вершин, между которыми мы ищем минимальное расстояние.

Матрицу смежности удобнее всего хранить в файле в следующем виде: первой строкой количество вершин, затем непосредственно сама матрица. В качестве бесконечно большого расстояния (INF, когда вершины не связаны ребром) взять конкретную константу. В моей реализации она равна 101, это значит, что максимальный вес ребра не может быть больше 100.

**Сложность алгоритма:**

Данная реализация работает за время Θ(n3), но требует Θ(n2) памяти. В целом, алгоритм Флойда очень прост, и, поскольку в нем используются только простые операции, константа, скрытая в определении Θ весьма мала.

**Код программы с комментариями**

#include <iostream>  
#include <algorithm>  
#include <fstream>  
  
//Максимальное значение веса = 100  
#define INF 101  
  
using namespace std;  
  
void printMatrix(int\*\* matrix, int numberOfVert) {  
 for(int i = 0; i < numberOfVert; i++) {  
 for(int j = 0; j < numberOfVert; j++) {  
 if(matrix[i][j] == INF) {  
 cout << "INF" << " ";  
 }  
 else {  
 cout << matrix[i][j] << " ";  
 }  
 }  
 cout << endl;  
 }  
 cout << endl;  
}  
  
//matrix - матрица смежности  
void floydWarshall(int \*\*matrix, int numberOfVert) {  
 //Пробегаемся по всем вершинам и ищем более короткий путь  
 //через вершину k  
 for(int k = 0; k < numberOfVert; k++) {  
 for(int i = 0; i < numberOfVert; i++) {  
 for(int j = 0; j < numberOfVert; j++) {  
 //Новое значение ребра равно минимальному между старым  
 //и суммой ребер i <-> k + k <-> j (если через k пройти быстрее)  
 matrix[i][j] = min(matrix[i][j], matrix[i][k] + matrix[k][j]);  
 }  
 }  
 }  
}  
  
int main(int argc, char\*\* argv) {  
 int menu;  
 string path;  
  
 cout << "How do you want to open the graph?" << endl  
 << "1 - From file by path," << endl  
 << "2 - Write it here." << endl  
 << "Enter:";  
 cin >> menu;  
 cout << endl;  
  
 int numberOfVert;  
 //Матрица смежности с весами ребер графа(101 - ребра нет, 0 ребро в себя)  
 int \*\*matrix = (int\*\*)malloc(sizeof(int)\*numberOfVert);  
 for(int i = 0; i < numberOfVert; i++) {  
 matrix[i] = (int \*) malloc(sizeof(int) \* numberOfVert);  
 }  
  
 if (menu == 1) {  
 cout << "Enter path:";  
 cin >> path;  
 //C:/Users/BadHard/CLionProjects/untitled1/matrix.txt  
 ifstream file(path);  
 file >> numberOfVert;  
 cout << numberOfVert << endl;  
  
 //Считываем матрицу весов ребер  
 for(int i = 0; i < numberOfVert; i++)  
 for(int j = 0; j < numberOfVert; j++)  
 file >> matrix[i][j];  
 file.close();  
 } else if (menu == 2) {  
 cout << "Use 101 as INF" << endl;  
 cout << "Enter a number of nodes:";  
 cin >> numberOfVert;  
 cout << endl;  
  
 //Считываем матрицу весов ребер  
 for(int i = 0; i < numberOfVert; i++) {  
 for(int j = 0; j < numberOfVert; j++) {  
 cout << j+1 << " length for the " << i+1 << " node:";  
 cin >> matrix[i][j];  
 cout << endl;  
 }  
 }  
 } else {  
 cout << "Wrong answer";  
 exit(0);  
 }  
  
 cout << "Old matrix:" << endl;  
 printMatrix(matrix, numberOfVert);  
  
 floydWarshall(matrix, numberOfVert);  
  
 cout << "New matrix:" << endl;  
  
 printMatrix(matrix, numberOfVert);  
  
 return 0;  
}

**Тестирование**

При запуске программа предлагает пользователю ввести данные длин дорог непосредственно в программе, либо загрузить данные из файла по введенному пути, рисунок 1. На данном этапе проверим ручной ввод с клавиатуры.

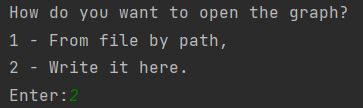


Рисунок 1. Выбор варианта ввода матрицы длин дорог

Далее программа предлагает ввести нам длины, рисунки 2-3.

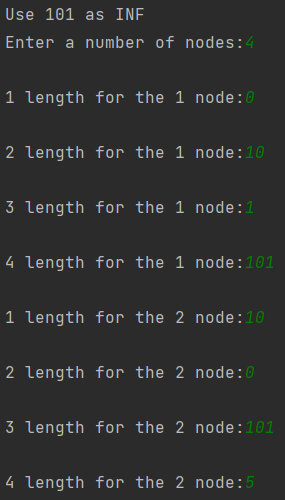


Рисунок 2. Ввод длин дорог

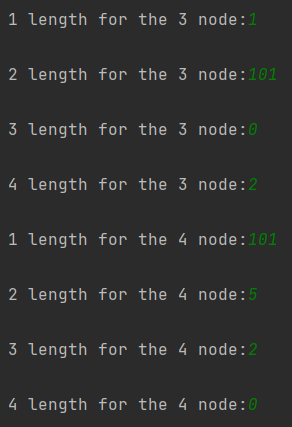


Рисунок 3. Продолжение ввода длин

Далее программа выводит старую матрицу дорог и новую с кратчайшими путями между узлами, рисунок 4.

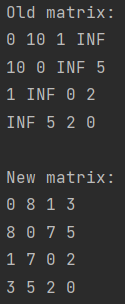


Рисунок 4. Результат работы программы

Теперь попробуем ввод матрицы дорог из файла, построенной на основе примерочного графа для индивидуального варианта, изображенного на рисунке 5. Содержание файла изображено на рисунке 6.

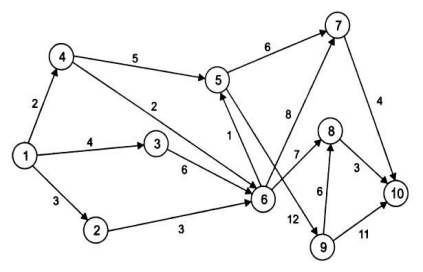


Рисунок 5. Граф индивидуального варианта задания

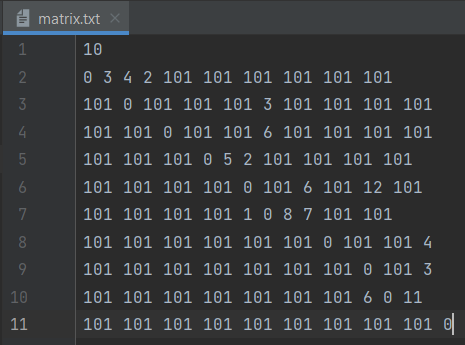


Рисунок 6. Содержание файла матрицы дорог индивидуального варианта

Протестируем программу, введя путь к файлу и проверим результат, рисунок 7.

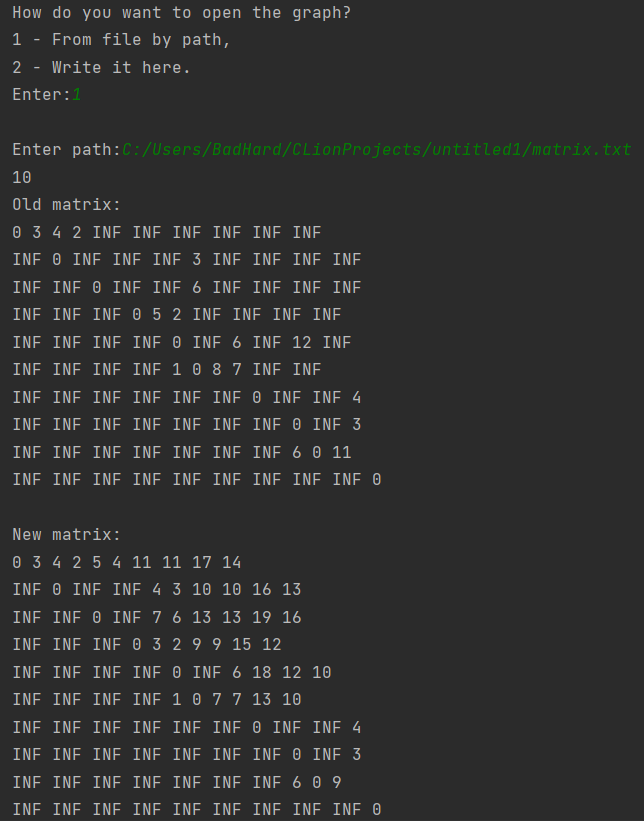


Рисунок 7. Вывод программы на основе данных, полученных из файла

На основе проведенного тестирования и выводов программы в зависимости от введенных данных, мы убедились в правильности работы программы и ее алгоритмов.

# Вывод

В результате выполнения данной работы мной были освоены навыки работы с графами. Также мной были освоены основные алгоритмы работы с графами, такие, как, например, поиск кратчайшего пути графа методом Флойда.