**Лабораторная работа №7**

**Условие**

Обработать графовую структуру в соответствии с указанным вариантом задания. Обосновать выбор необходимого алгоритма и выбор структуры для представления графов. Ввод данных – на усмотрение программиста. Результат выдать в графической форме. Задана система двусторонних дорог. Для каждой пары городов найти длину кратчайшего пути между ними.

**Сроки исполнения**

2 недели, начиная с 10.12.2018

**Основание для разработки**

Заказчик – Силантьева Александра Васильевна

Исполнитель – Ильясов Идрис Магомет-Салиевич, студент группы ИУ7-33Б

Основание для разработки – лабораторная работа №7

**Наименование**

Программа для хранения и поиска и поиска кратчайших путей между городами.

**Область применения**

Поиск кратчайшего пути между двумя точками (например, между городами, станциями метро).

**Назначение разработки**

Общая концепция системы – консольное приложение.

Описание функционала системы:

Ввод и вывод производится пользователем вручную через консоль с помощью меню.

В меню пользователь может загрузить информацию для графа из заранее подготовленного файла, задать граф вручную, вывести полученный граф в файл файл .png и получить картинку, вывести длину кратчайшего пути между каждой парой городов, длину кратчайшего пути из одного города до других, вывести на экран матрицы кратчайших расстояний, проверить связность графа, вывести минимальное остовное дерево в файл .png.

**Требования к программе**

Программа написана на языке СИ.

При неправильном вводе пользователем какого-либо значения программа просит пользователя повторить попытку ввода, выдавая сообщение об ошибке.

**Входные данные**

Граф задается матрицей стоимостей. Матрица стоимости – квадратная матрица размерности n, в которой значение элемента равно весу ребра, если вершины смежные, и равно знаку «бесконечности», если вершины несмежные.

Данная матрица получается программой из файла: первое число в файле – размерность матрицы. После этого идет сама матрица такая, что элемент [i][j] равен весу ребра, если из i в j есть дорога, [i][j] равен 0, если дороги нет.

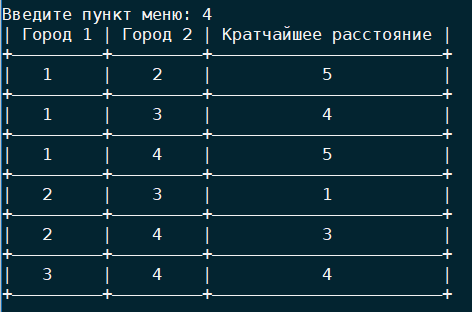
Граф также можно задать вручную: пользователь вводит размерность графа, вершину i, вершину j, вес ребра из i в j через пробел.

Например, пользователь вводит 1 2 4. Значит, дорога между городами №1 и №2 равная 4.

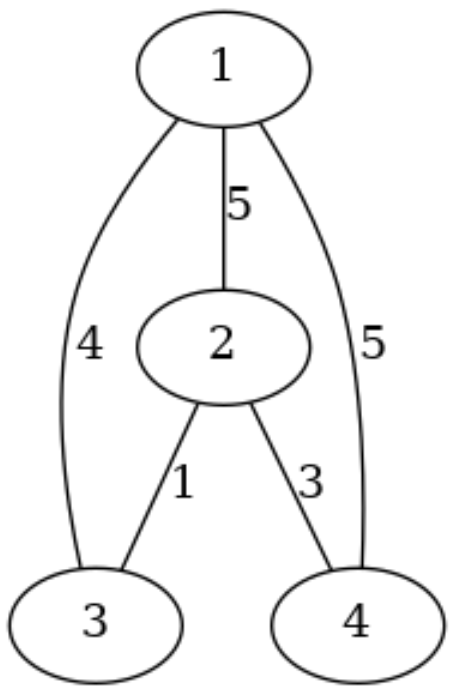
**Выходные данные**

Выходные данные в данной программе:

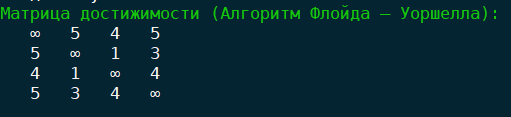
1. Таблица кратчайших расстояний между каждым из городов. Таблица представлена в формате:



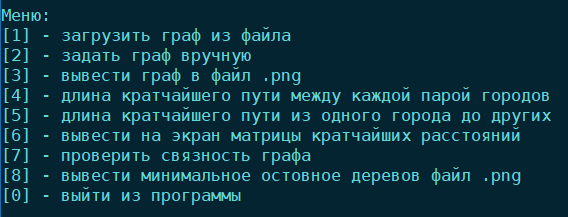
1. Изображение графа в формате .png, заданного матрицей стоимостей с указанием весов ребер:



1. Матрица кратчайших расстояний, полученная вследствие работы алгоритма Флойда-Уоршелла.



**Взаимодействие с программой**



[1] – загрузить граф из файла (уже заранее подготовленный).

[2] – задать граф вручную. Пользователь сам задаст граф, используя программу.

[3] – вывести граф в файл .png. Выводится граф в виде картинки.

[4] – длина кратчайшего пути между каждой парой городов. Выводится таблица с расстояниями до каждый городов.

[5] – длина кратчайшего пути из одного города до других. Выводятся расстояния между городом, который указал пользователь, и другими городами.

[6] – вывести на экран матрицы кратчайших расстояний. Выводится матрица, полученная по алгоритму Флойда-Урошелла.

[7] – проверить связность графа. Проверяется связность/несвязность графа.

[8] – вывести минимальное остовное дерево в файл .png. Выводится остовное дерево в виде картинки.

[0] – выйти из программы. Производится выход из программы

**Используемые алгоритмы в программе:**

Для поиска кратчайших расстояний между любыми двумя городами используется алгоритм Флойда – Уоршелла. Этот алгоритм необходим для нахождения кратчайших расстояний между всеми вершинами взвешенного ориентированного графа.

Алгоритм:

W - матрица стоимостей.

После работы алгоритма W - матрица кратчайших расстояний.

**for** k = 1 **to** n  
 **for** i = 1 **to** n  
 **for** j = 1 **to** n  
 W[i][j] = min(W[i][j], W[i][k] + W[k][j])

Алгоритм Флойда – Уоршелла является эффективным для расчета всех кратчайших путей в плотных графах, когда имеет место большое количество пар ребер между парами вершин. Алгоритм имеет кубическую сложность.

Для поиска кратчайших расстояний из одного города до всех остальных используется алгоритм Дейкстры. Так как расстояние между городами не может быть представлено отрицательным числами, значит, эффективнее использовать алгоритм Дейкстры, нет нужды в алгоритме Белламана – Форда, который уступает по времени.

Алгоритм:

already\_used - массив посещенных вершин

min\_rasst - массив минимальных расстояний

do

{

min\_ind = INFINITY;

min = INFINITY;

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

{

if (already\_used[i] == 1 && min\_len[i] < min)

{

min = min\_len[i];

min\_ind = i;

}

}

if (min\_ind != INFINITY)

{

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

{

if (matrix[min\_ind][i] > 0)

{

tmp = min + matrix[min\_ind][i];

if (tmp < min\_len[i])

{

min\_len[i] = tmp;

}

}

}

already\_used[min\_ind] = 0;

}

} while (min\_ind < INFINITY);

Для построения минимального оства используется алгоритм Прима – алгоритм построения минимального остовного дерева взвешенного связного неориентированного графа.

Алгоритм:

На вход алгоритма подаётся связный неориентированный граф. Для каждого ребра задаётся его стоимость.

Сначала берется произвольная вершина и находится ребро, инцидентное данной вершине и обладающее наименьшей стоимостью. Найденное ребро и соединяемые им две вершины образуют дерево. Затем, рассматриваются рёбра графа, один конец которых — уже принадлежащая дереву вершина, а другой — нет; из этих ребер выбирается ребро наименьшей стоимости. Выбираемое на каждом шаге ребро присоединяется к дереву. Таким образом, при выполнении каждого шага алгоритма, высота формируемого дерева увеличивается на 1. Рост дерева происходит до тех пор, пока не будут исчерпаны все вершины исходного графа.

**Функции, используемые в программе для работы с графами**

**int \*\*allocate\_matrix(int n, int m);**

**int \*\*fill\_matrix(FILE \*file, int \*\*matrix, int cnt);**

**void print\_matrix(int \*\*matrix, int cnt, int flag);**

**void get\_graph(int \*\*matrix, int cnt);**

**void get\_graph\_prim(int cnt, int \*\*matrix, int \*vector);**

**int \*dijkstra(int cnt, int \*\*matrix, int v0, int \*min\_len);**

**int minimum(int a, int b);**

**int \*\*floyd\_warshall(int n, int \*\*matrix, int \*\*short\_matrix);**

**int connected\_graph(int \*\*short\_matrix, int \*\*matrix, int cnt);**

**int first\_zero(int \*arr, int n);**

**int \*prim(int \*\*matrix, int \*result, int cnt, int v0);**

**Модули и тесты**

|  |  |  |
| --- | --- | --- |
| **Входные данные** | **Вывод** | **Класс эквивалентности** |
| Меню: y | Сообщение об ошибке, завершение программы | Некорректный ввод в меню |
| Меню: 3 | Успешный ввод | Корректный ввод в меню |
| Ввод номер узла (города): 2 | Успешный ввод | Корректный ввод номера узла |
| Ввод номер узла (города): f | Сообщение об ошибке, завершение программы | Некорректный ввод номера узла |
| Ввод данных: 1 2 5 | Успешный ввод | Корректный ввод при заполнении вручную |
| Ввод данных: -1 -2 -5 | Успешный ввод | Выход из режима заполнения вручную |
| Ввод данных: f a b | Сообщение об ошибке, завершение программы | Некорректный ввод при заполнении вручную |

**Вывод**

В ходе данной работы я научился работать с графами, матрицами смежности, достижимости. Научился определять кратчайшее расстояние между двумя узлами графа, используя разные алгоритмы. Причину выбора именно алгоритмов Дейкстры, Флойда-Уоршелла описана в «используемые алгоритмы в программе».

**Контрольные вопросы:**

1. *Что такое граф?*

Граф – конечное множество вершин и соединяющих их рёбер; G = <V, E>. Если пары Е (ребра) имеют направление, то граф называется направленным; если ребро имеет вес, то граф называется взвешенным.

1. *Как представляются графы в памяти?*

Существуют различные методы представления графов в программе. Матрица смежности B(n\*n) – элемент b[i,j]=1, если существует ребро, связывающее вершины iи j, и =0, если ребра не существует.

Список смежностей – содержит для каждой вершины из множества вершин V список тех вершин, которые непосредственно связаны с ней. Входы в списки смежностей могут храниться в отдельной таблице, либо же каждая вершина может хранить свой список смежностей.

1. *Какие операции возможны над графами?*

Основные операции над графами: обход вершин и поиск различных путей: кратчайшего пути от вершины к вершине; кратчайшего пути от вершины ко всем остальным; кратчайших путей от каждой вершины к каждой; поиск эйлерова пути и гамильтонова пути, если таковые есть в графе.

1. *Какие способы обхода графов существуют?*

Один из основных методов проектирования графовых алгоритмов – поиск в глубину. Начиная с некоторой вершины v0, ищется ближайшая смежная ей вершина v, для которой в свою очередь осуществляется поиск в глубину до тех пор, пока не встретится ранее просмотренная вершина, или не закончится список смежности вершины v (то есть вершина полностью обработана). Если нет новых вершин, смежных с v, то вершина v считается использованной, идет возврат в вершину, из которой попали в вершину v, и процесс продолжается до тех пор, пока не получим v = v0. При просмотре используется стек.

Поиск в ширину – обработка вершины V осуществляется путем просмотра сразу всех «новых» соседей этой вершины, которые последовательно заносятся в очередь просмотра. Для поиска кратчайших путей используются алгоритмы Дейкстры, Беллмана-Форда, Флойда- Уоршалла.

1. *Где используются графовые структуры?*

Графовые структуры могут использоваться в задачах, в которых между элементами могут быть установлены произвольные связи, необязательно иерархические. Наиболее распространенным является использование графов при решении различных задач о путях, будь то построение коммуникационных линий между городами или прокладка маршрута на игровом поле.

1. *Какие пути в графе Вы знаете?*

Путь в графе, проходящий через каждое *ребро* ровно один раз, называется *эйлеровым* путём; путь может проходить по некоторым вершинам несколько раз – в этом случае он является непростым. Путь, проходящий через каждую вершину ровно один раз, называется гамильтоновым путем. Как эйлеров, так и гамильтонов путь могут не существовать в некоторых графах.

1. *Что такое каркасы графа?*

Каркас графа – дерево, в которое входят все вершины графа, и некоторые (не обязательно все) его рёбра. Для построения каркасов графа используются алгоритмы Крускала и Прима.