МГТУ им. Н.Э. Баумана

**Дисциплина Типы и Структуры данных.**

**Лабораторный практикум №7**

**по теме:** «Графы»

Работу выполнил:

студент группы ИУ7-35Б

Прянишников Александр

**Цель работы**: Обработать графовую структуру в соответствии с заданным вариантом. Обосновать выбор необходимого алгоритма и выбор структуры для представления графов. Ввод данных осуществить на усмотрение программиста. Результат выдать в графической форме..

# Условие задачи. Вариант 3

# 

# Требования к задаче

**Входные данные**

Название файла, где хранится граф.

**Вывод данных**

Граф в текстовом виде, граф в графическом виде, пары вершин графа и длина кратчайшего пути между ними, время эффективности работы.

**Описание задачи, реализуемой программой**

Программа считывает граф из файла, а затем по выбранным алгоритмам подсчитывает для каждой пары длину кратчайшего пути.

**Способ обращения к программе.**

Программу можно запустить через терминал.

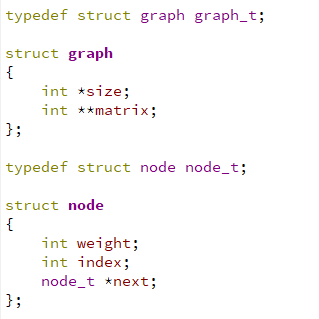
**Описание возможных аварийных ситуаций и ошибок пользователя**

* Выбор отсутствующего пункта меню (от 0 до 4)  
  *В этом случае программа сообщит пользователю о невозможности совершения операции и попросит ввести новое значение пункта меню.*
* Ввод букв при вводе числа  
  *В этом случае программа сообщит пользователю о невозможности совершения операции и попросит ввести новое значение пункта меню.*
* Пустой ввод  
  *В этом случае программа сообщит пользователю о невозможности совершения операции и попросит ввести новое значение пункта меню.*
* Ввод файла, которого не существует  
  *В этом случае программа сообщит пользователю, что файла не существует.*
* Граф не связный.  
  *В этом случае для вершин, которых достичь нельзя, будет выведено –1.*

# Описание внутренних структур данных.

Для решения этой задачи я решил использовать две структуры: матрица стоимостей и список стоимостей. При этом нужно учитывать, что по условию граф неориентированный, и веса дуг не могут быть отрицательными.

Вот так описаны структуры:



# Описание алгоритма

Сначала опишем пункты меню, которые можно вызвать:

**1 – Вывести текущий граф**Программа выводит на экран и в файл текущий граф.

**2 – Найти кратчайшие пути между вершинами методом Флойда**  
Программа подсчитывает для текущего графа кратчайшие пути между вершинами методом Флойда, а затем выводит результат на экран и в файл.

**3 – Найти кратчайшие пути между вершинами методом Дейкстры**Программа подсчитывает для текущего графа кратчайшие пути между вершинами методом Дейкстры, а затем выводит результат на экран и в файл.

**4 – Найти кратчайшие пути между вершинами методом Беллмана**Программа подсчитывает для текущего графа кратчайшие пути между вершинами методом Беллмана, а затем выводит результат на экран и в файл.

**5 – Сравнить эффективность**Программа строит таблицу затрат по времени и памяти для различных способов хранения данных и алгоритмов подсчёта кратчайшего пути.

**6 – Сменить граф**Программа считывает от пользователя название нового файла с графом и открывает его.

**0 – Выход.**

Теперь обговорим способы хранения матрицы.

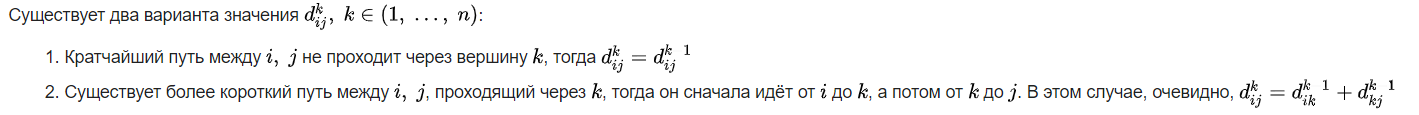
Первый вариант – матрица стоимостей. Пусть у нас есть N вершин, тогда создаётся матрица N \* N, и значению Aij соответствует вес дуги между вершинами.

Второй вариант – список стоимостей. Создаётся массив связных списков, и для каждой вершины в соответствующем связном списке хранятся те элементы, куда идёт дуга из заданной вершины, также хранится вес дуги.

Также реализовано три алгоритма подсчёта кратчайших путей.

**Флойд**

На каждом шаге алгоритм генерирует матрицу W, которая содержит длины кратчайших путей между всеми вершинами графа. Перед работой алгоритма матрица W заполняется длинами рёбер графа.



**Дейкстра**

Каждой вершине из ***V*** сопоставим метку — минимальное известное расстояние от этой вершины до ***a***. Алгоритм работает пошагово — на каждом шаге он «посещает» одну вершину и пытается уменьшать метки. Работа алгоритма завершается, когда все вершины посещены.

Метка самой вершины ***a*** полагается равной 0, метки остальных вершин — бесконечности. то отражает то, что расстояния от ***a*** до других вершин пока неизвестны. Все вершины графа помечаются как непосещённые.

Если все вершины посещены, алгоритм завершается. В противном случае, из ещё не посещённых вершин выбирается вершина ***u***, имеющая минимальную метку. Мы рассматриваем всевозможные маршруты, в которых ***u*** является предпоследним пунктом. Вершины, в которые ведут рёбра из ***u***, назовём *соседями* этой вершины. Для каждого соседа вершины ***u***, кроме отмеченных как посещённые, рассмотрим новую длину пути, равную сумме значений текущей метки ***u*** и длины ребра, соединяющего ***u*** с этим соседом.

Если полученное значение длины меньше значения метки соседа, заменим значение метки полученным значением длины. Рассмотрев всех соседей, пометим вершину ***u*** как посещённую и повторим шаг.

**Беллман**

Построим матрицу A, элементы которой будут обозначать следующее: aij— это длина кратчайшего пути из s в j, содержащего не более j рёбер.

Путь, содержащий 0 рёбер, существует только до вершины s. Таким образом, a0j равно 0 при i = s и бесконечности в противном случае.

Теперь рассмотрим все пути из s в i, содержащие ровно j рёбер. Каждый такой путь есть путь из j-1 ребра, к которому добавлено последнее ребро. Если про пути длины j-1 все данные уже подсчитаны, то определить j-й столбец матрицы не составляет труда.

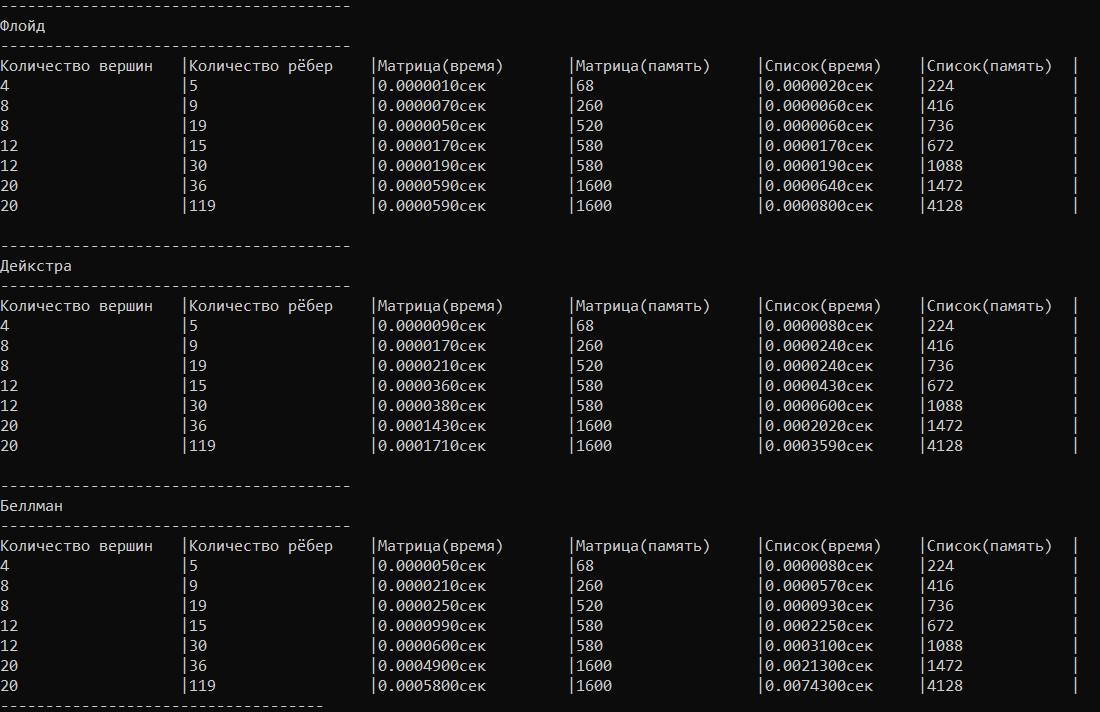
# Тесты

Тестирование происходило по такому принципу: каждый пункт тестировался отдельно, после чего контролировалась работа программы целиком. Поэтому здесь указаны только негативные тесты, которые в целом проверяли работоспособность модулей программы.

|  |  |
| --- | --- |
| **123345fd** | **Буквы при задании числа** |
|  | **Пустой ввод** |
| **–1** | **Выбор неправильного пункта меню** |
| **1 <–> 2 : не существует** | **Граф несвязный** |
| **Такого файла нет!** | **Некорректный ввод названия файла.** |

# Оценка эффективности

Сравнение эффективности по времени доступно в самой программе.



По этой таблице можно сделать несколько выводов.

Во–первых, реализация матрицей практически всегда быстрее, чем реализация на списке, но при этом список выгоднее по памяти только в случае разреженного графа. Поэтому экономить на памяти часто нерационально, лишь в случае очень разреженного графа список лучше.

Во–вторых, алгоритм Флойда показал наилучшие результаты: он быстрее остальных способов. Алгоритм Беллмана показал себя самым медленным среди всех, но при этом он работает и для отрицательных весов. Однако по условию задачи таких весов быть не может, поэтому для этой задачи он оказался невыгодным.

# Выводы по проделанной работе

Сегодня я познакомился с двумя способами хранения графа: в виде матрицы стоимостей и списка стоимостей, а также с тремя способами вычисления кратчайших путей. Алгоритм Флойда для этой задачи оказался оптимальным, как и матричное хранение.

# Ответы на контрольные вопросы

Что такое граф?

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

Как представляются графы в памяти?

С помощью матрицы смежности или списков смежности.

Какие операции возможны над графами?

Обход вершин, поиск различных путей, исключение и включение вершин.

Какие способы обхода графов существуют?

Обход в ширину и обход в глубину.

Где используются графовые структуры?

Графовые структуры могут использоваться в задачах, в которых между элементами могут быть установлены произвольные связи, необязательно иерархические.

Какие пути в графе Вы знаете?

Эйлеров путь, непростой путь, гамильтонов путь.

Что такое каркасы графа?

Каркас графа – дерево, в которое входят все вершины графа, и некоторые (необязательно все) его рёбра.