**Министерство образования и науки Российской Федерации**

**ФГБОУ ВО «Тульский государственный университет»**

**Технический колледж им. С.И. Мосина**

**Курсовая работа**

**по дисциплине «Математическое моделирование»**

**на тему: «Реализация задачи о нахождении кратчайшего пути в графе»**

**Выполнил,**

**студент гр. 3-090203**

**Проверил преподаватель:**

**Янтиков И.А.**

**Воронцова Н.В.**

**Тула 2018**

Содержание

[Введение 3](#_Toc512011460)

[1. Теоретическая часть 4](#_Toc512011461)

[1.1 Общая постановка задачи 4](#_Toc512011462)

[1.2 Алгоритмы нахождения кратчайшего пути 5](#_Toc512011463)

[2. Практическая часть 8](#_Toc512011464)

[2.1 Постановка задачи 8](#_Toc512011465)

[2.2 Аналитическое решение задачи 9](#_Toc512011466)

[2.3 Решение задачи с применением вычислительной техники. 11](#_Toc512011467)

[Заключение 14](#_Toc512011468)

[Список использованной литературы 15](#_Toc512011469)

# Введение

Поставлена задача нахождения кратчайших путей в графе. Требуется вычислить минимальное расстояние между вершиной графа и всеми остальными его вершинами (метод Дейкстры, Беллмана-Форда), кратчайшее расстояние между всеми вершинами графа (метод Флойда-Уоршелла).

# 1. Теоретическая часть

## 1.1 Общая постановка задачи

**Задача о кратчайшем пути** (англ. *shortest path problem*) — задача поиска самого короткого пути (цепи) между двумя точками (вершинами) на графе, в которой минимизируется сумма весов ребер, составляющих путь [1].

Задача о кратчайшем пути является одной из важнейших классических задач теории графов. Сегодня известно множество алгоритмов для ее решения.

Значимость данной задачи определяется ее различными практическими применениями. Например, в GPS-навигаторах, где осуществляется поиск кратчайшего пути между двумя перекрестками. В качестве вершин выступают перекрестки, а дороги являются ребрами, которые лежат между ними. Сумма расстояний всех дорог между перекрестками должна быть минимальной, тогда найден самый короткий путь.

Существуют различные постановки задачи о кратчайшем пути:

* Задача о кратчайшем пути в заданный пункт назначения. Требуется найти кратчайший путь в заданную вершину назначения t, который начинается в каждой из вершин графа (кроме t). Поменяв направление каждого принадлежащего графу ребра, эту задачу можно свести к задаче о единой исходной вершине (в которой осуществляется поиск кратчайшего пути из заданной вершины во все остальные).
* Задача о кратчайшем пути между заданной парой вершин. Требуется найти кратчайший путь из заданной вершины u в заданную вершину v.
* Задача о кратчайшем пути между всеми парами вершин. Требуется найти кратчайший путь из каждой вершины u в каждую вершину v. Эту задачу тоже можно решить с помощью алгоритма, предназначенного для решения задачи об одной исходной вершине, однако обычно она решается быстрее.
* В различных постановках задачи, роль длины ребра могут играть не только сами длины, но и время, стоимость, расходы, объем затрачиваемых ресурсов (материальных, финансовых, топливно-энергетических и т. п.) или другие характеристики, связанные с прохождением каждого ребра. Таким образом, задача находит практическое применение в большом количестве областей (информатика, экономика, география и др.).

Эти задачи известны как задачи «единственным источником» и «между всеми парами вершин» [2].

## 1.2 Алгоритмы нахождения кратчайшего пути

**Алгоритм Дейкстры**

Данный алгоритм является алгоритмом на графах, который изобретен нидерландским ученым Э. Дейкстрой в 1959 году. Алгоритм находит кратчайшее расстояние от одной из вершин графа до всех остальных и работает только для графов без ребер отрицательного веса [3].

Каждой вершине приписывается вес – это вес пути от начальной вершины до данной. Также каждая вершина может быть выделена. Если вершина выделена, то путь от нее до начальной вершины кратчайший, если нет – то временный. Обходя граф, алгоритм считает для каждой вершины маршрут, и, если он оказывается кратчайшим, выделяет вершину. Весом данной вершины становится вес пути. Для всех соседей данной вершины алгоритм также рассчитывает вес, при этом ни при каких условиях не выделяя их. Алгоритм заканчивает свою работу, дойдя до конечной вершины, и весом кратчайшего пути становится вес конечной вершины.

*Алгоритм Дейкстры*

Шаг 1. Всем вершинам, за исключением первой, присваивается *вес* равный бесконечности, а первой вершине – 0.

Шаг 2. Все вершины не выделены.

Шаг 3. Первая *вершина* объявляется текущей.

Шаг 4. *Вес* всех невыделенных вершин пересчитывается по формуле: *вес* невыделенной вершины есть минимальное число из старого веса данной вершины, суммы веса текущей вершины и веса *ребра*, соединяющего текущую вершину с невыделенной.

Шаг 5. Среди невыделенных вершин ищется *вершина* с минимальным весом. Если таковая не найдена, то есть *вес* всех вершин равен бесконечности, то *маршрут* не существует. Следовательно, *выход*. Иначе, текущей становится найденная *вершина*. Она же выделяется.

Шаг 6. Если текущей вершиной оказывается конечная, то *путь* найден, и его *вес* есть *вес* конечной вершины.

Шаг 7. Переход на шаг 4.

**Алгоритм Беллмана — Форда**

Также как и алгоритм Дейкстры, алгоритм Беллмана — Форда вычисляет во взвешенном графе кратчайшие пути от одной вершины до всех остальных. Он подходит для работы с графами, имеющими ребра с отрицательным весом. Но спектр применимости алгоритма затрагивает не все такие графы, ввиду того, что каждый очередной проход по пути, составленному из ребер, сумма весов которых отрицательна (т. е. по отрицательному циклу), лишь улучшает требуемое значение. Бесконечное число улучшений делает невозможным определение одного конкретного значения, являющегося оптимальным. В связи с этим алгоритм Беллмана — Форда не применим к графам, имеющим отрицательные циклы, но он позволяет определить наличие таковых, о чем будет сказано позже [4].

Основными вычисляемыми величинами этого алгоритма являются величины λj(k), где i = 1, 2, … , n (n – число вершин графа); k = 1, 2, … , n – 1. Для фиксированных i и k величина λj(k) равна длине минимального пути, ведущего из заданной начальной вершины х1 в вершину хi и содержащего не более k дуг.

Шаг 1. Установка начальных условий.

Ввести число вершин графа n и матрицу весов C = (cij).

Шаг 2. Положить k = 0. Положить λi(0) = ∞ для всех вершин, кроме х1; положить λi(0) = 0.

Шаг 3. В цикле по k, k = 1,..., n – 1, каждой вершине xi на k-ом шаге приписать индекс λi(k) по следующему правилу:

для всех вершин, кроме х1, положить λi(k)= 0.

В результате работы алгоритма формируется таблица индексов λi(k), i = 1, 2, … , n, k = 0, 1, 2, … , n – 1. При этом λi(k) определяет длину минимального пути из первой вершины в i-ую, содержащего не более k дуг.

Шаг 4. Восстановление минимального пути.

Для любой вершины xs предшествующая ей вершина xr определяется из соотношения:

где G -1(xs) - прообраз вершины xs.

Для найденной вершины xr предшествующая ей вершина xq определяется из соотношения:

где G -1(xr) - прообраз вершины xr, и т. д.

Последовательно применяя это соотношение, начиная от последней вершины xi , найдем минимальный путь.

**Алгоритм Флойда — Уоршелла**

Этот алгоритм предназначен для нахождения кратчайших расстояний между всеми вершинами взвешенного ориентированного графа [5].

Пусть вершины графа пронумерованы от 1 до n и введено обозначение для длины кратчайшего пути от i до j, который кроме самих вершин i, j проходит только через вершины 1…k.

Очевидно, что — длина (вес) ребра (i, j), если таковое существует (в противном случае его длина может быть обозначена как ∞)

Существует два варианта значения :

1. Кратчайший путь между i, j не проходит через вершину k, тогда

Существует более короткий путь между i,\;j, проходящий через k, тогда он сначала идёт от i до k, а потом от k до j. В этом случае, очевидно,

1. Таким образом, для нахождения значения функции достаточно выбрать минимум из двух обозначенных значений.

Тогда рекуррентная формула для имеет вид:

— длина ребра (i, j)

Алгоритм Флойда — Уоршелла последовательно вычисляет все значения \forall i,\; j для k от 1 до n. Полученные значения d_{i j}^{n} являются длинами кратчайших путей между вершинами i,\; j.

# 2. Практическая часть

## 2.1 Постановка задачи

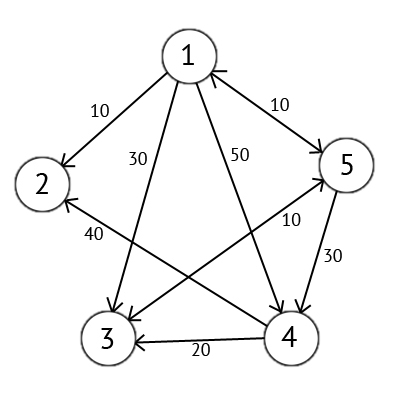
Дан граф G (рис. 1): 

Рисунок 1

Этот граф мы можем представить в виде матрицы С (табл. 1):

Таблица 1 Матрица смежности

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 |
| 1 |  | 10 | 30 | 50 | 10 |
| 2 |  |  |  |  |  |
| 3 |  |  |  |  | 10 |
| 4 |  | 40 |  |  |  |
| 5 | 10 |  | 10 | 30 |  |

Требуется найти минимальные расстояния от вершины 1 ко всем остальным вершинам графа.

## 2.2 Аналитическое решение задачи

Возьмем в качестве источника вершину 1. Это значит, что мы будем искать кратчайшие маршруты из вершины 1 в вершины 2, 3, 4 и 5.

Данный алгоритм пошагово перебирает все вершины графа и назначает им метки, которые являются известным минимальным расстоянием от вершины источника до конкретной вершины.

Присвоим 1-й вершине метку равную 0, потому как эта вершина — источник. Остальным вершинам присвоим метки равные бесконечности (рис. 2):

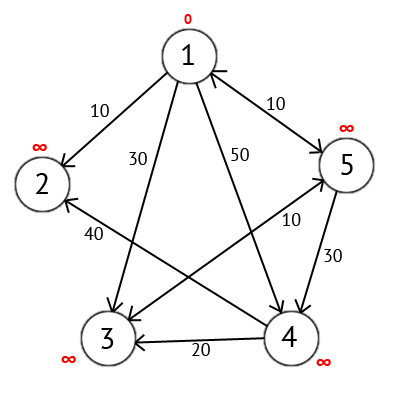


Рисунок 2

Далее выберем такую вершину W, которая имеет минимальную метку (сейчас это вершина 1) и рассмотрим все вершины, в которые из вершины W есть путь, не содержащий вершин посредников. Каждой из рассмотренных вершин назначим метку равную сумме метки W и длинны пути из W в рассматриваемую вершину, но только в том случае, если полученная сумма будет меньше предыдущего значения метки. Если же сумма не будет меньше, то оставляем предыдущую метку без изменений (рис. 3):

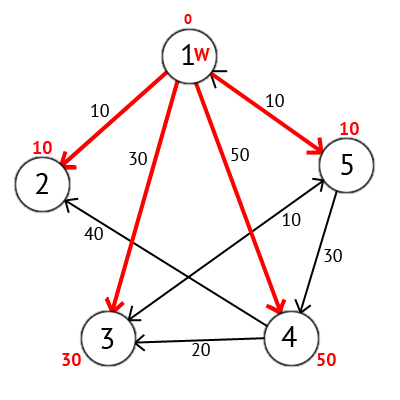


Рисунок 3

После того как мы рассмотрели все вершины, в которые есть прямой путь из W, вершину W мы отмечаем как посещённую, и выбираем из ещё не посещенных такую, которая имеет минимальное значение метки, она и будет следующей вершиной W. В данном случае это вершина 2 или 5.

Мы выберем вершину 2. Но из нее нет ни одного исходящего пути, поэтому мы сразу отмечаем эту вершину как посещенную и переходим к следующей вершине с минимальной меткой. На этот раз только вершина 5 имеет минимальную метку. Рассмотрим все вершины, в которые есть прямые пути из 5, но которые ещё не помечены как посещенные. Снова находим сумму метки вершины W и веса ребра из W в текущую вершину, и если эта сумма будет меньше предыдущей метки, то меняем значение метки на полученную сумму (рис. 4):

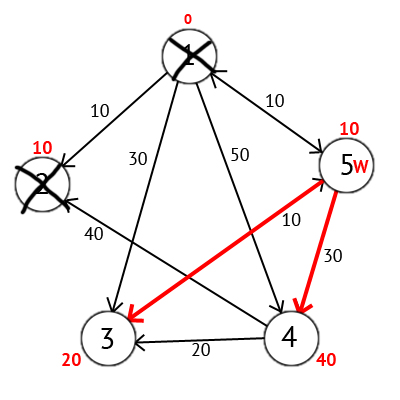
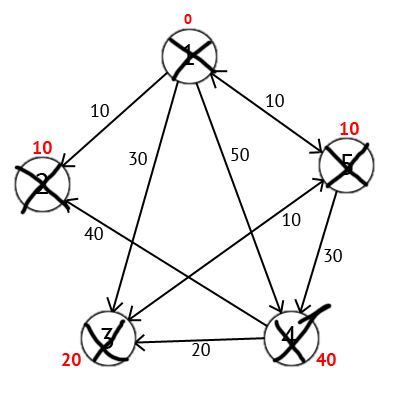


Рисунок 4

Мы можем увидеть, что метки 3-ей и 4-ой вершин стали меньше, то есть был найден более короткий маршрут в эти вершины из вершины источника. Далее отмечаем 5-ю вершину как посещенную и выбираем следующую вершину, которая имеет минимальную метку. Повторяем все перечисленные выше действия до тех пор, пока есть непосещенные вершины.

Выполнив все действия, получим такой результат (рис. 5):

Минимальные расстояния представлены в таблице 2:

Таблица 2 Минимальные расстояния

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 |
| 1  Рисунок 5 | 0 | 10 | 20 | 40 | 10 |

## 2.3 Решение задачи с применением вычислительной техники.

Для реализации данной была выбрана интегрированная среда разработки IntelliJ IDEA в сочетании с платформой разработки JavaFX, функционирующей на основе языка программирования Java.

Программа в исходном состоянии имеет вид (рис. 6):

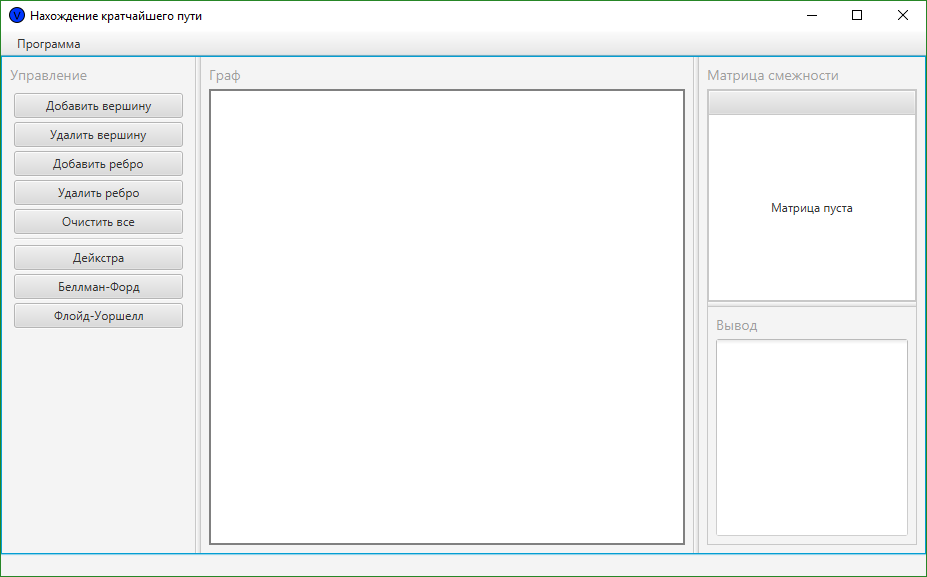


Рисунок 6

Визуализация графа и реализация поставленной задачи в приложении методом Дейкстры / Беллмана-Форда (рис. 7):

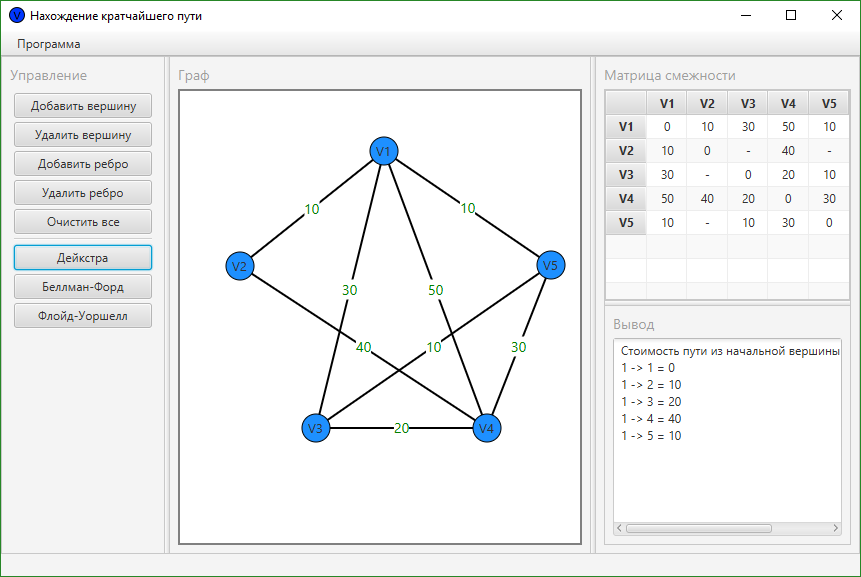


Рисунок 7

Реализация методом Флойда-Уоршелла (рис. 8):

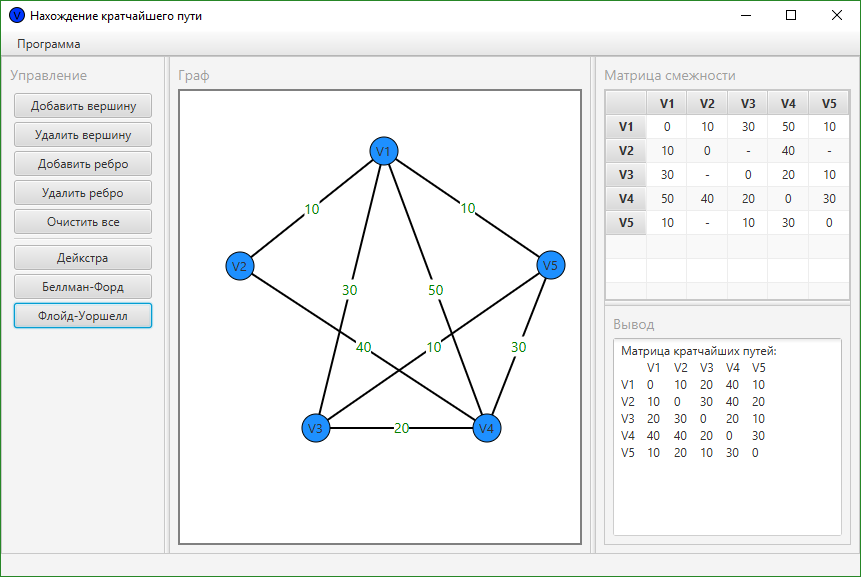
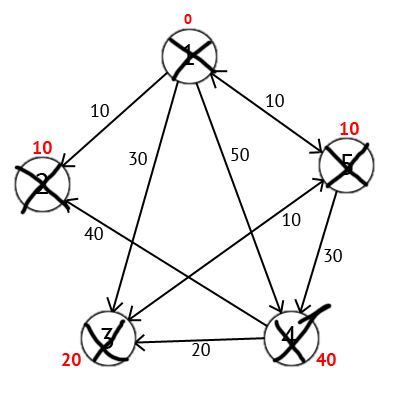


Рисунок 8

2.4 Сравнение полученных результатов

После решения задачи аналитически были получение следующие результаты

(рис. 9):



Минимальные расстояния:

Таблица 2 Минимальные расстояния

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 |
| 1 | 0 | 10 | 20 | 40 | 10 |

Рисунок 9

Как видно, данные вычислений при использовании программы полностью совпадают с результатами аналитического решения (рис. 10):

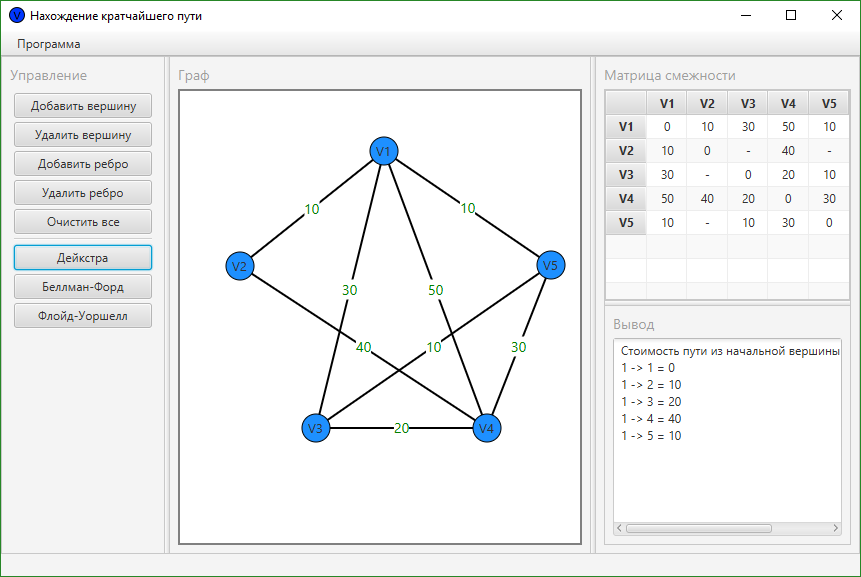


Рисунок 10

# Заключение

Задача нахождения кратчайших путей в графе решена. Требуемые минимальные расстояния в графе были вычислены представленными выше методами.

# Список использованной литературы

1. Нахождение кратчайшего пути в графе

<http://life-prog.ru/1_56629_nahozhdenie-kratchayshego-puti-v-grafe.html>

1. Касьянов В. Н., Евстигнеев В. А. «Графы в программировании: обработка визуализации и применение». — СПб., БВХ-Петербург, 2003. — 1104 c.
2. Алгоритмы на графах. Алгоритм Дейкстры.

<https://www.intuit.ru/studies/courses/648/504/lecture/11475%253Fpage%253D2>

1. Алгоритм Беллмана — Форда

<https://studopedia.ru/5_159666_algoritm-floyda--uorshella.html>

1. Алгоритм Флойда — Уоршелла

<https://dic.academic.ru/dic.nsf/ruwiki/358111>