**Анализ Алгоритмов**

Рубежный контроль №1

«Реализация поиска кратчайшего маршрута в графе, описывающем карту метро»

Аминов Тимур

Группа: ИУ7-55

Преподаватели: Волкова Л.Л., Строганов Ю.В.

**Оглавление**

[**Введение** 3](#_Toc26498221)

[**Постановка задачи** 4](#_Toc26498222)

[**1. Аналитическая часть** 5](#_Toc26498223)

[**Математическое описание** 5](#_Toc26498224)

[**Вывод** 6](#_Toc26498225)

[**3. Технологическая часть** 7](#_Toc26498226)

[**Вывод** 7](#_Toc26498227)

[**4. Исследовательская часть** 8](#_Toc26498228)

[**Тестирование** 8](#_Toc26498229)

[**Вывод** 9](#_Toc26498230)

[**Литература** 10](#_Toc26498231)

# **Введение**

В данной работе будет рассмотрен алгоритм Дейкстры. Этот алгоритм на графах находит кратчайшие пути от одной из вершин графа до всех остальных. Недостатком этого алгоритма является то, что он не может работать в графе, ребра которого имеют отрицательные веса. Однако это не является проблемой для задачи поиска кратчайшего маршрута на карте метро, так как предполагается, что все ребра имеют положительные веса.

# **Постановка задачи**

Цель: реализовать поиск кратчайшего маршрута в графе, описывающем карту метро, используя алгоритм Дейкстры.

Задачи:

* реализовать алгоритм;
* протестировать алгоритм;
* привести листинг на одном из языков программирования.

# **Аналитическая часть**

В данной части будет приведен математический обзор и описание алгоритма Дейкстры.

## **Математическое описание**

Обозначения:

*V –* множество вершин графа

*E –* множество ребер графа

*w[i j] –* вес (длина) ребра *i j*

*a –* вершина, расстояния от которой ищется

*U –* множество посещенных вершин

*d[u] –* по окончании работы алгоритма равно длине кратчайшего пути из *а*

*p[u]* – по окончании работы алгоритма содержит кратчайший путь из *a* в *u*

*v* – вершина, расстояния до которой ищется

В простейшей реализации для хранения чисел *d[i]* можно использовать массив чисел, а для хранения принадлежности элемента множеству *U* — массив булевых переменных.

В начале алгоритма расстояние для начальной вершины полагается равным нулю, а все остальные расстояния заполняются большим положительным числом (большим максимального возможного пути в графе). Массив флагов заполняется нулями. Затем запускается основной цикл.

На каждом шаге цикла мы ищем вершину с минимальным расстоянием и флагом равным нулю. Затем мы устанавливаем в ней флаг в 1 и проверяем все соседние с ней вершины *u*. Если в них расстояние больше, чем сумма расстояния до текущей вершины и длины ребра, то уменьшаем его. Цикл завершается, когда флаги всех вершин становятся равны 1, либо когда у всех вершин c флагом 0 *d[i] = inf*. Последний случай возможен тогда и только тогда, когда граф G несвязный.

## 

## **Вывод**

Был проведен обзор математической составляющей алгоритма Дейкстры.

# **3. Технологическая часть**

В данном разделе будет приведен листинг алгоритма Дейкстры на языке Python 3.7 Был выбран данный язык, так как имеется большой опыт работы с ним.

Листинг 1. Алгоритм Дейкстры

|  |
| --- |
| 1. def dijkstra(matrix, start): 2. n = len(matrix) 3. inf\_num = 10000 4. u = [False for i in range(n)] 5. d = [inf\_num for i in range(n)] 6. d[start] = 0 7. paths = [-1 for i in range(n)] 8. for i in range(n): 9. cur\_d = inf\_num + 1 10. cur\_p = -1 11. for j in range(n): 12. if not u[j] and d[j] < cur\_d: 13. cur\_d = d[j] 14. cur\_p = j 15. u[cur\_p] = True 16. for j in range(n): 17. if matrix[cur\_p][j] != 0: 18. old\_d = d[j] 19. d[j] = min(d[j], d[cur\_p] + matrix[cur\_p][j]) 20. if d[j] != old\_d: 21. paths[j] = cur\_p 22. return d, paths |

## **Вывод**

В данном разделе был приведен код алгоритма Дейкстры на языке программирования Python.

# **4. Исследовательская часть**

## **Тестирование**

При каждом запуске программы, она составляет граф карты метро, где станции – это узлы, а переходы и соединения между станциями – ребра. Для тестов были выбраны маршруты с заранее известным временем и набором станций.

Маршрут на красной ветке от Проспекта Вернадского до Коммунарки. Как видно по схеме (Рисунок 1) время маршрута должно составить 14, а состоять он должен только из станций на этой ветке, находящихся между искомыми.

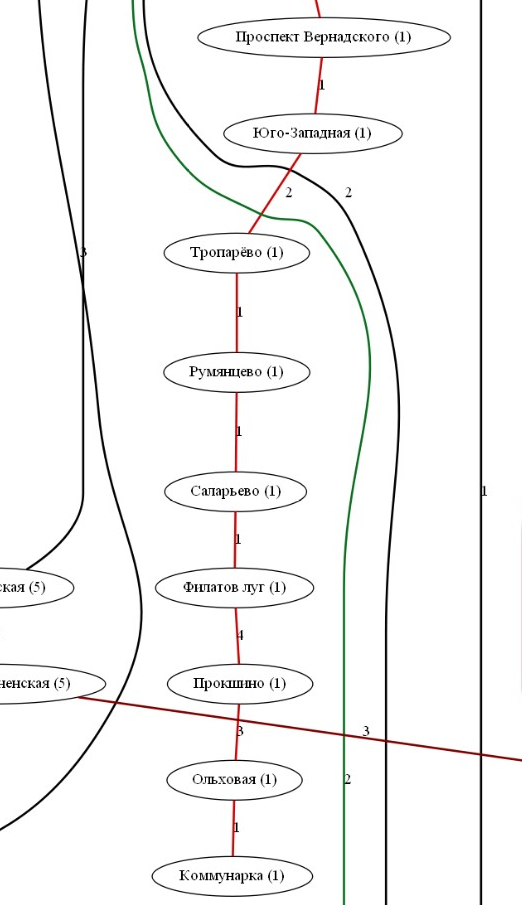


Рисунок 1.

*Листинг 2. Маршрут*

|  |
| --- |
| Станция отправления: Коммунарка  Станция прибытия: Проспект вернадского  Время маршрута: 14  Маршрут:  Коммунарка (1) --[1]--> Ольховая (1) --[3]--> Прокшино (1) --[4]--> Филатов луг (1) --[1]--> Саларьево (1) --[1]--> Румянцево (1) --[1]--> Тропарёво (1) --[2]--> Юго-Западная (1) --[1]--> Проспект Вернадского (1) |

Вывод программы (Листинг 2.) совпадает с приведенной схемой.

## **Вывод**

В данном разделе было проведено тестирование нахождения кратчайшего расстояния в графе, используя алгоритм Дейкстры.

# **Литература**

[1] *Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн.*  Introduction to Algorithms, 2006 .

[2] [*Левитин А. В.*](https://www.wikidata.org/wiki/Q21694518) Глава 9. Жадные методы: Алгоритм Дейкстры // [Алгоритмы. Введение в разработку и анализ](https://www.wikidata.org/wiki/Q21694522), 2006.

[3] Карта метро // [Электронный ресурс]. Режим доступа: https://yandex.ru/metro