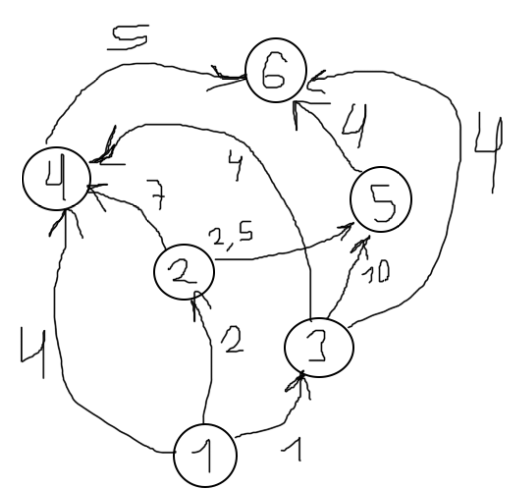
Дан ориентированный взвешенный граф следующего вида. Нужно найти с помощью алгоритма Дейкстры (о создателе алгоритма можно почитать [тут](https://habr.com/ru/articles/303712/) и [тут](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%B9%D0%BA%D1%81%D1%82%D1%80%D0%B0,_%D0%AD%D0%B4%D1%81%D0%B3%D0%B5%D1%80_%D0%92%D0%B8%D0%B1%D0%B5), [тут2](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%94%D0%B5%D0%B9%D0%BA%D1%81%D1%82%D1%80%D1%8B), также алгоритм Дейкстры схож с алгоритмом Примы для остовных деревьев) кратчайший путь от 1 до 6 вершины.

Основной материал взят отсюда <https://www.youtube.com/watch?v=tyQSgTytc4s>



Для этого будем составлять таблицу итераций алгоритма, но сначала обозначим несколько основных правил алгоритма (хаотично расположены):

1. Введем метку постоянности “[]”, когда она есть, то значит, что мы сейчас находимся в этой вершине. Для 0 итерации, мы задаем метку постоянности для 1 вершины, а вес назначаем равным нулю (запись для вершины [1] и его веса [0]). Для 1ой и последующих итераций метку постоянности выбираем таким образом: берем значение наименьшего веса в строке. При этом мы сохраняем значения посещенных вершин. На 0 итерации, посещенная вершина s = {1}, на момент, когда мы нашли наименьший вес и определили следующую вершину, мы также добавляем это значение в s. Например, это будет 3 вершина, тогда s = {1, 3}. И теперь мы перешли в 3 вершину.

2. Находясь в вершине, мы знаем в какие пути мы можем пойти. Например, от 1 вершины можно пойти в 2, 3 и 4.

3. После итерации мы должны сделать перерасчет для всех вершин по следующим правилам:  
-Если мы можем от нашей вершины перейти к вершине указанной в столбце таблицы, то прибавляем вес по такому принципу: к значению веса вершины, которая на данный момент выбрана меткой [], мы прибавляем значение веса между вершинами. И делаем это только в том случае, если значение веса не больше того значения, которое получится в процессе суммирования (исключение для 1 итерации), иначе оставляем. Если мы до этой вершины в столбце не можем дойти, то просто эти значения переносим с предыдущей строки.

Таблица:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Вершины  Итерация | [1] | [2] | [3] | [4] | [5] | [6] | Соседние  вершины | Посещенные  вершины  S = {} |
| 0 | [0] |  |  |  |  |  | 2,3,4 | 1 |
| 1 | - | 2 | [1] | 4 |  |  | 4,5,6 | 1,3 |
| 2 | - | [2] | - | 4 | 11 | 5 | 4,5 | 1,3,2 |
| 3 | - | - | - | [4] | 4,5 | 5 | 6 | 1,3,2,4 |
| 4 | - | - | - | - | [4,5] | 5 | 6 | 1,3,2,4,5 |
| 5 | - | - | - | - | - | [5] | - | 1,3,2,4,5,6 |
| ИТОГ | 0 | 2 | 1 | 4 | 4,5 | 5 | 5 |  |

Примечания:

1. “-” стоит там, где откуда мы идем, и что мы уже посетили.

2. Последнее значение есть ничто иное, как минимальное расстояние на графе, которое нужно преодолеть от вершины 1 до вершины 6. Это путь по вершинам 1,3,6. Как его определить этот путь численно? Тут в экселе говорят, что надо наоборот смотреть в какую вершину из какой мы попали. <https://www.youtube.com/watch?v=4TscCFj3REk> 23.00

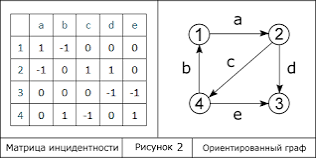
3. Алгоритм работает только для графов без рёбер отрицательного веса!

4. Значение веса в строке, обозначенной [], показывает минимальны путь от вершины 1 до это вершины.

Можно представить граф на рисунке следующим образом:

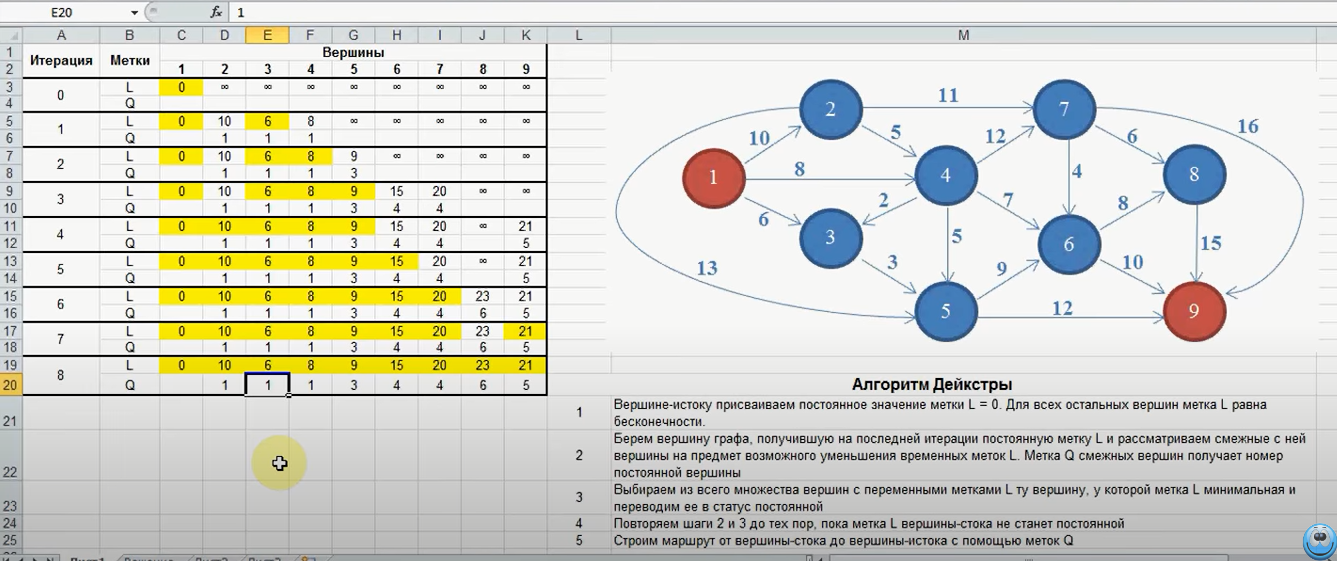


(это другое представление)

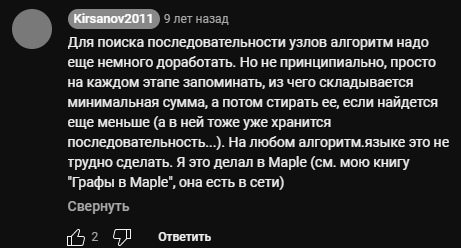


Я собираюсь сгенерировать матрицу смежности около 5 штук разных размерностей и дать каждому задание протестировать на этих данных свой алгоритм дейкстры, построить графики.

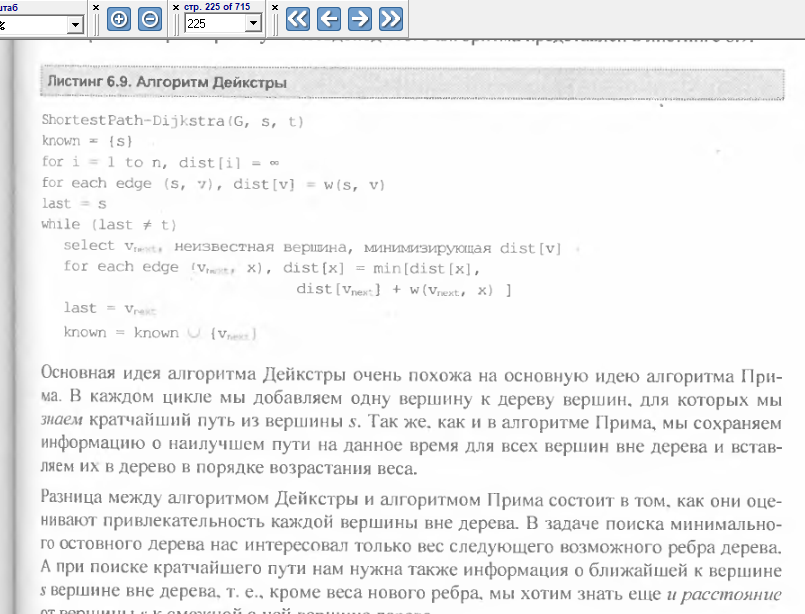
!Реализация очереди с приоритетом есть в книге алгоритмы по Python.

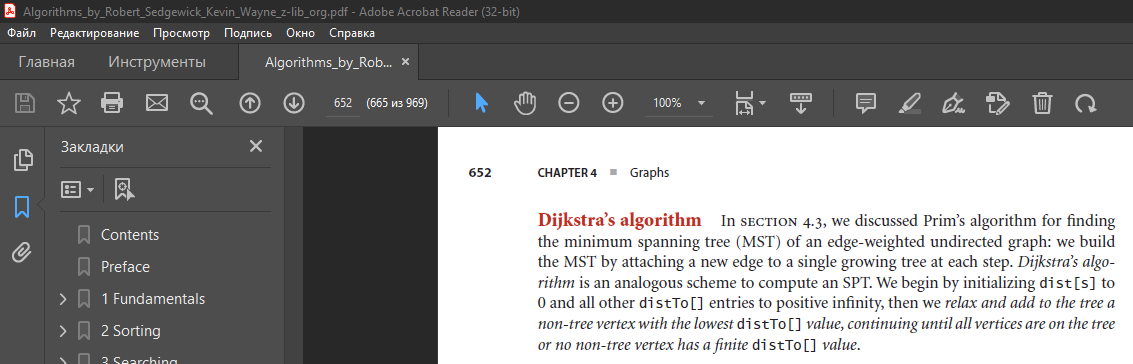


|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Вершины  Итерация | [1] | [2] | [3] | [4] | [5] | [6] | Соседние  вершины | Посещенные  вершины  S = {} |
| 0 | [0] |  |  |  |  |  | 2,3,4 | 1 |
| M |  |  |  |  |  |  |  |  |
| 1 | - | 2 | [1] | 4 |  |  | 4,5,6 | 1,3 |
| M |  | 1 | 1 | 1 |  |  |  |  |
| 2 | - | [2] | - | 4 | 11 | 5 | 4,5 | 1,3,2 |
| M |  | 1 | 1 | 1 | 2 | 2 |  |  |
| 3 | - | - | - | [4] | 4,5 | 5 | 6 | 1,3,2,4 |
| M |  | 1 | 1 | 1 | 4 | 4 |  |  |
| 4 | - | - | - | - | [4,5] | 5 | 6 | 1,3,2,4,5 |
| M |  |  |  |  | 4 | 4 |  |  |
| 5 | - | - | - | - | - | [5] | - | 1,3,2,4,5,6 |
| M |  |  |  |  |  | 5 |  |  |
| ИТОГ | 0 | 2 | 1 | 4 | 4,5 | 5 | 5 |  |



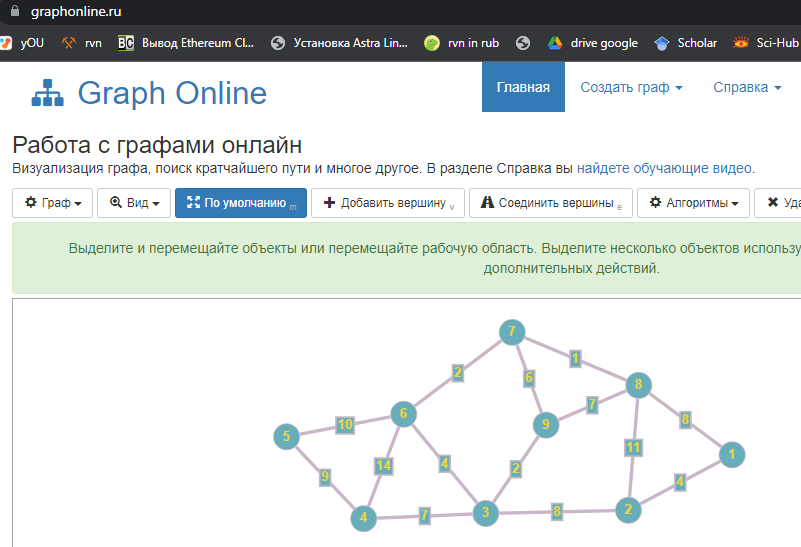
58 страниа его книги





<https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/>

<https://graphonline.ru/>



0, 4, 0, 0, 0, 0, 0, 8, 0,

4, 0, 8, 0, 0, 0, 0, 11, 0,

0, 8, 0, 7, 0, 4, 0, 0, 2,

0, 0, 7, 0, 9, 14, 0, 0, 0,

0, 0, 0, 9, 0, 10, 0, 0, 0,

0, 0, 4, 14, 10, 0, 2, 0, 0,

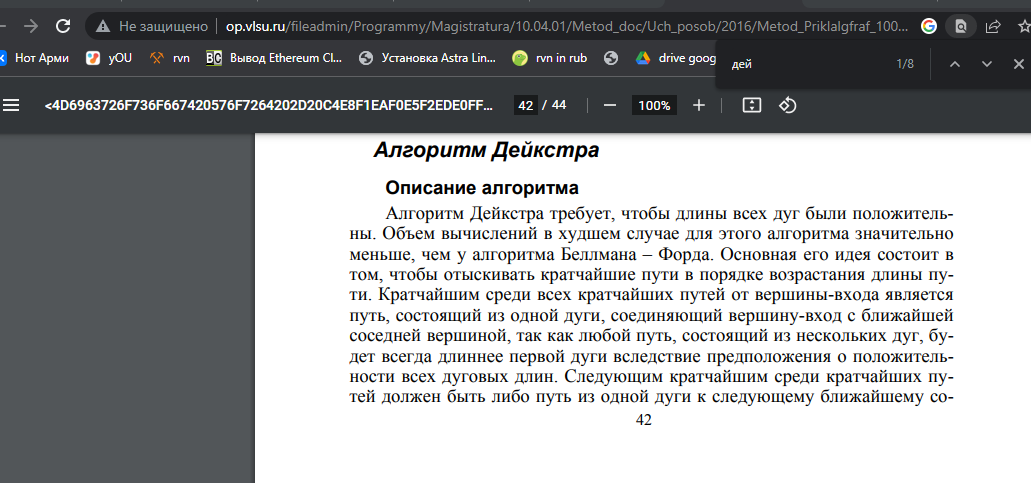
0, 0, 0, 0, 0, 2, 0, 1, 6,

8, 11, 0, 0, 0, 0, 1, 0, 7,

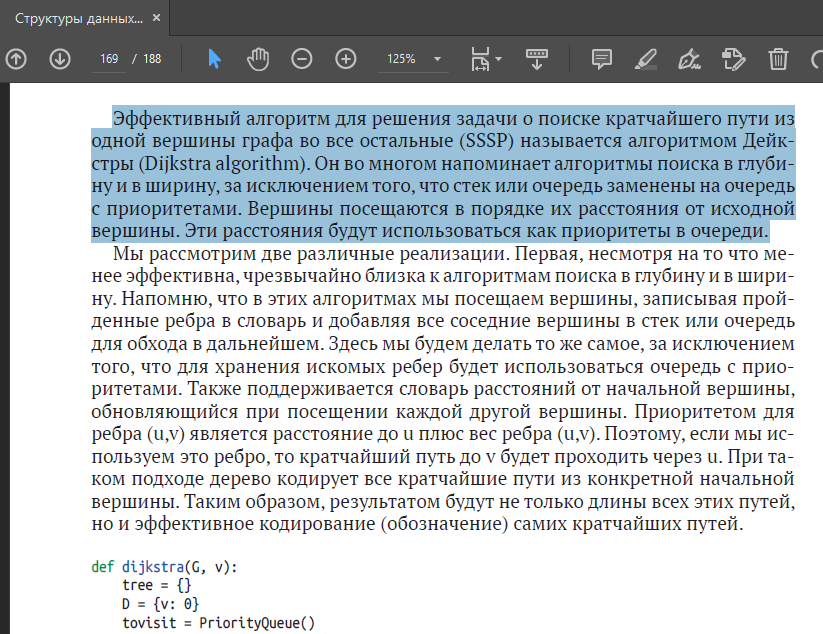
0, 0, 2, 0, 0, 0, 6, 7, 0,

Еще тут <https://python-scripts.com/dijkstras-algorithm>

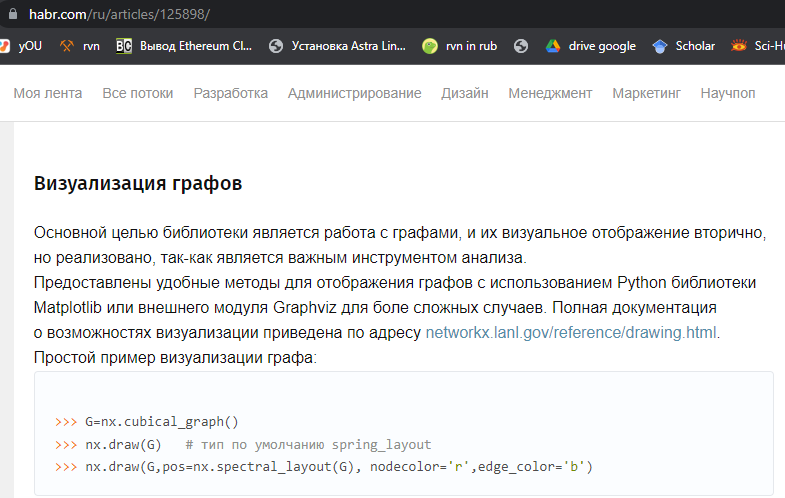
(тут вроде показано как помечать маршрут)

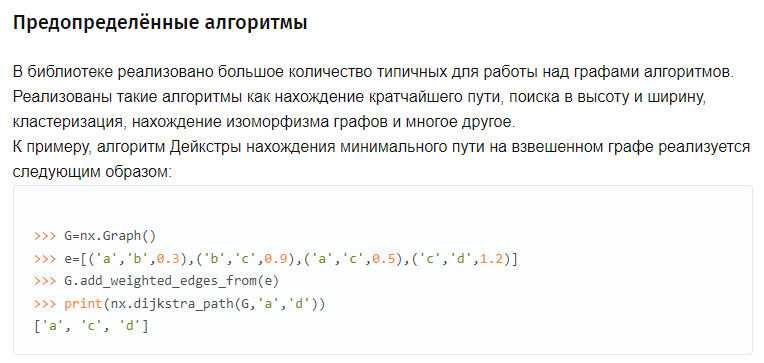


<https://www.youtube.com/watch?v=bSpRWSBJsaY>



<https://habr.com/ru/articles/125898/> или <http://asocialnetworks.blogspot.com/2016/02/python-networkx.html>





Основная документация <https://networkx.org/documentation/stable/reference/classes/graph.html>