Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Российский химико-технологический университет имени Д.И. Менделеева»

Кафедра информационных компьютерных технологий

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4

Выполнил студент группы КС-36 Алёшин Михаил Алексеевич

Ссылка на репозиторий: https://github.com/MUCTR-IKT-CPP/MAAleshin\_36\_algo

Приняли: Пысин Максим Дмитриевич

Краснов Дмитрий Олегович

Дата сдачи: 14.02.2022

Оглавление

[Описание задачи. 2](#_Toc63548272)

[Описание метода/модели. 2](#_Toc63548273)

[Выполнение задачи. 2](#_Toc63548274)

[Заключение. 2](#_Toc63548275)

# Описание задачи.

Необходимо создать взвешенный граф, состоящий из [100, 200, 500, 1000] вершин.

* Каждая вершина графа связана со случайным количеством вершин, минимум с [30, 40, 100, 200].
* Веса ребер задаются случайным значением от 1 до 20.
* Каждая вершина графа должна быть доступна, т.е. до каждой вершины графа должен обязательно существовать путь до каждой вершины, не обязательно прямой.

Выведите получившийся граф в виде матрицы смежности.

Для каждого графа требуется провести серию из 5 - 10 тестов, в зависимости от времени, затраченного на выполнение одного теста, необходимо найти кратчайшие пути между всеми вершинами графа и их длину с помощью алгоритма Дейкстры.

В рамках каждого теста, необходимо замерить потребовавшееся время на выполнение задания из пункта 3 для каждого набора вершин. По окончанию всех тестов необходимо построить график используя полученные замеры времени, где на ось абсцисс (Х) нанести N – количество вершин, а на ось ординат(Y) - значения затраченного времени.

# Описание метода/модели.

Алгоритм находит кратчайший путь от заданной вершины ко всем другим вершинам графа, включая требуемую конечную вершину t.

Псевдокод:

Кратчайший-путь-Дейкстры (Граф, начальная, конечная)

Из начальной точки до начальной точки путь занимает 0 единиц.

Итерируем от 1 до n, устанавливаем дистанции для каждой вершины в максимум

Начиная с начальной точки и до тех пор, пока есть новый узел:

Начиная с первого ребра от текущей точки и до тех пор, пока есть необследованные ребра

Берем вес текущего ребра и номер конечной вершины.

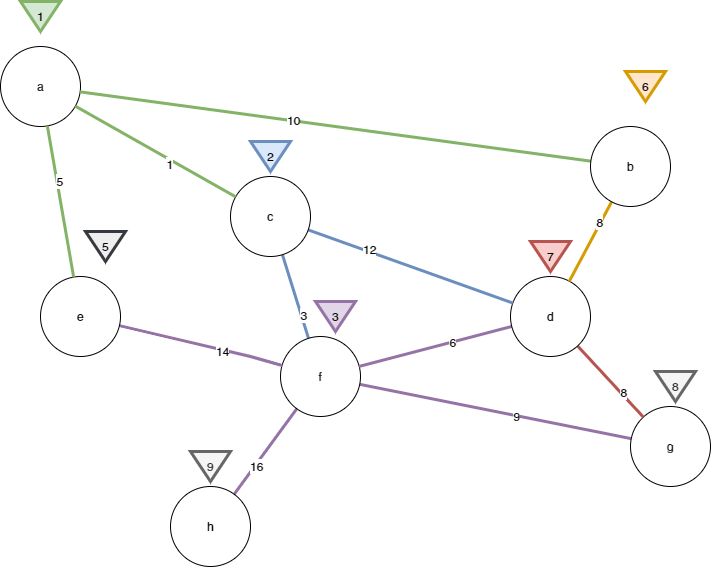
Сравниваем уже вычисленное ранее расстояние до конечной вершины ребра и дистанцию до текущего узла суммированную с весом ребра, выбираем минимальное и записываем как дистанцию до конечной вершины ребра, а также значение родительского для измененного пути.

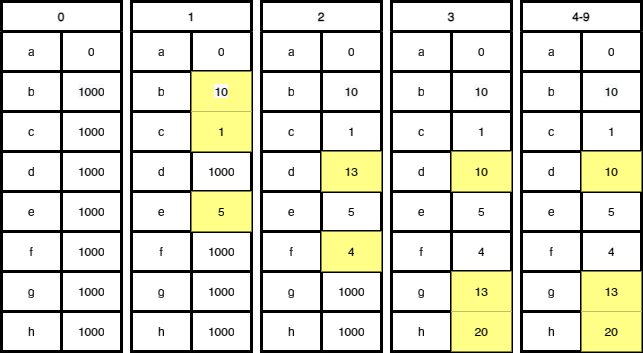
Выбираем следующей вершиной ту, которая имеет минимальное расстояние от изначальной.

Важно помнить, что алгоритм дейкстры не работает в графах с отрицательным значением. Сложность алгоритма O(n^2)

Путь из алгоритма получается обратным раскручиванием массива родителей.

Предполагается что алгоритм останавливается только тогда, когда посетит все вершины, даже если таблица маршрутов перестанет меняться.



a

# Выполнение задачи.

Данная лабораторная работа была реализована на языке python. Реализован класс Graph. При объявлении класса Generate при помощи метода generate\_graph() генерируется новый граф с заданными условиями. Далее при помощи метода get\_adjacency\_matrix() можно получить матрицу смежности с весами. На рис. 1 представлена данная матрица.

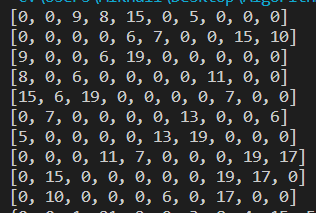


Рис. 1. Матрица весов.

Так же реализован сам алгоритм Дейкстры в аналогичной функции. Данный алгоритм находит кратчайшее расстояние до всех точек от заданной нами начальной точки. Вывод результатов алгоритмов представлен на рис. 2.



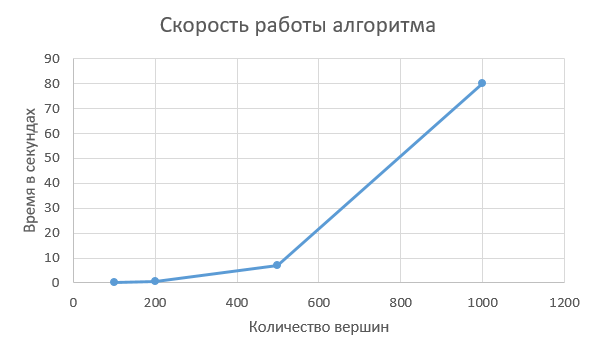
Рис. 2. Результат работы алгоритма Дейкстры.

Для того, чтобы найти пути до каждой точки, была реализована функция get\_path(), в которой с помощью найденных значений из алгоритма Дейкстры ищется кратчайшие пути до всех точек.



Рис. 3. Кратчайшие пути до точек.

Далее в функции analys() генерируются 5 графов, каждый из которых больше предыдущего. Для каждой вершины есть минимальное количество исходящих ребер, что обеспечивает существование пути до каждой точки с большой вероятностью. После этого происходит поиск кратчайших весов до точки и поиск самого пути до точки, просчитывается скорость работы алгоритма. Далее на рис. 1 можно увидеть результаты скорости работы алгоритма в зависимости от количества вершин:

  
Рис. 2. Скорость работы алгоритма.

# Заключение.

Исходя из полученных нами данных можно увидеть, что скорость работы алгоритма действительно близится к O(n^2). Я считаю данных алгоритм довольно хорошо оптимизирован, по сравнению, например, с DFS, в котором при большом количестве вершин происходит переполнение стека. Однако данных алгоритм имеет и свои минусы. Например, алгоритм Дейкстры не может работать с минусовыми весами, так как он будет зацикливаться при расчете лучшего пути до точки. Так же, важным критерием для работы алгоритма является наличие пути до всех точек (что можно будет проверить алгоритмами BFS или DFS). Скорость работы алгоритма возможно оптимизировать до O(n log n), если попытаться из очереди убрать повторяющиеся точки, что увеличит скорость работы алгоритма.