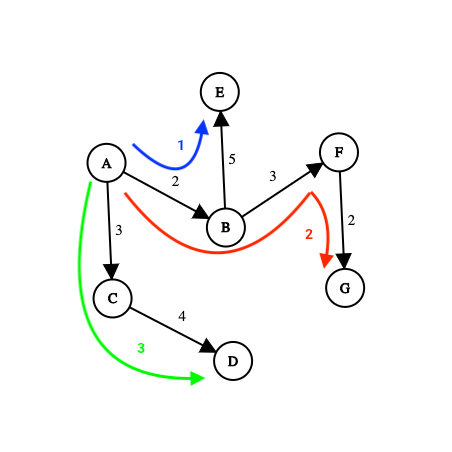
# **Алгоритмы на графах**

## **Поиск в глубину (англ. Depth-first search, DFS)**

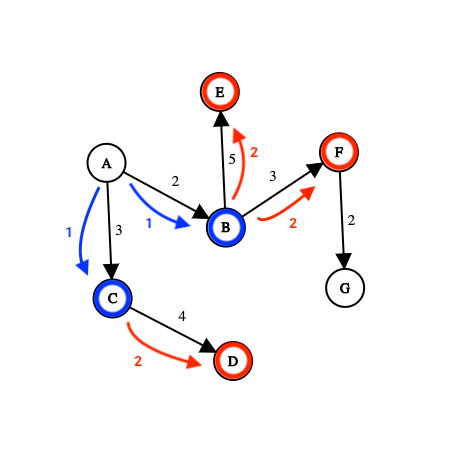
Рассмотрим идею данного алгоритма на примере ориентированного графа. Нам необходимо найти путь из точки A в точку D. Применяя алгоритм DFS, мы до конца исследуем один из возможных вариантов маршрута. Если искомая вершина не обнаружена, возвращаемся и исследуем другой путь, до тех пор пока не исследуем все варианты.

Необходимость многократного повторения процедуры указывает на необходимость использования рекурсии для реализации алгоритма.

Для того чтобы реализовать данный алгоритм, нам потребуется познакомиться с такой структурой данных, как стек. Немного подробнее про стек написано в дополнительных файлах: *text4teacher2* или *text4pup.*

## **Поиск в ширину (англ. breadth-first search, BFS)**

Вместо того чтобы двигаться по определенному пути до конца рассматриваемого варианта, алгоритм BFS подразумевает посещение ближайших соседей за одно действие (шаг), затем посещение соседей уже посещенных соседей (соседей второго уровня) и так далее – до тех пор, пока не будет обнаружен искомый узел.



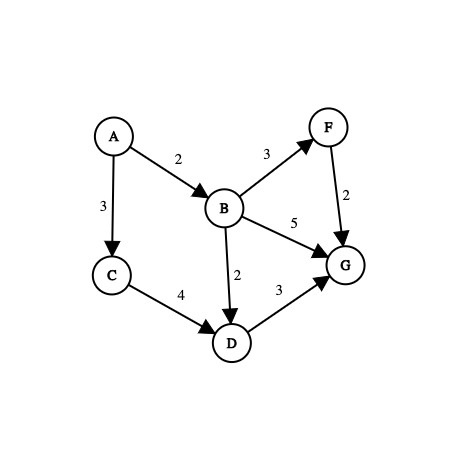
Для того чтобы реализовать данный алгоритм, нам потребуется познакомиться с такой структурой данных, как очередь. Немного подробнее про очередь написано в дополнительных файлах: *text4teacher2* или *text4pup.*

## **Алгоритм Дейкстры**

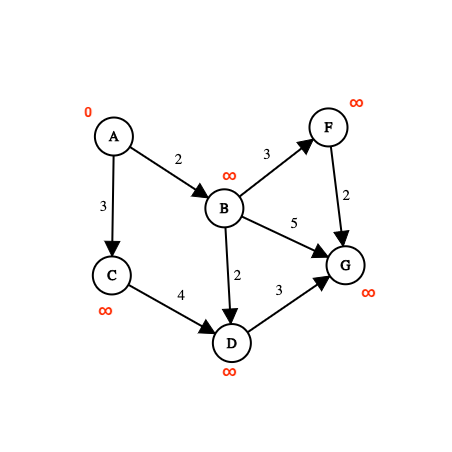
В дополнительном документе *case2* представлены примеры «жадных» алгоритмов. Они просты, но не всегда позволяют найти лучшее решение.

В 1959 году нидерландский учёный Эдсгер Дейкстра разработал алгоритм (англ. Dijkstra’s algorithm), который находит все кратчайшие пути от одной вершины графа до всех остальных. Алгоритм широко применяется в программировании и современных информационных технологиях.

Например, рассмотрим взвешенный ориентированный граф. Необходимо найти кратчайшие расстояния от начальной вершины A до всех остальных.



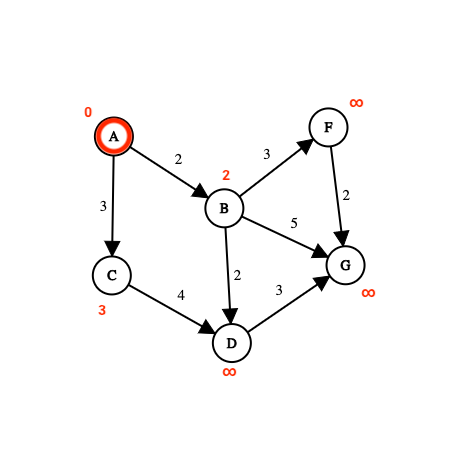
Вначале сопоставим каждой вершине метку – минимальное известное расстояние от этой вершины до вершины A. Метка самой вершины A полагается равной 0 (так как кратчайший путь из вершины A в саму эту вершину равен 0), метки остальных вершин – бесконечности. Это отражает то, что расстояния от A до других вершин пока неизвестны.



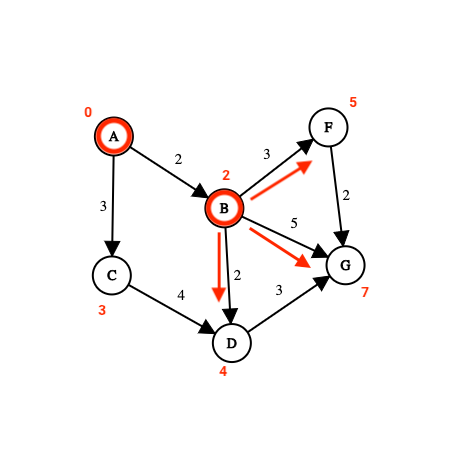
Рассмотрим основную идею на примере графа, изображенного на рисунке выше. Если сумма весов BD + DG меньше веса BG, то лучше воспользоваться маршрутом через вершину D.

Данный алгоритм работает пошагово – на каждом шаге он «посещает» одну вершину и пытается уменьшать метки. Работа алгоритма завершается, когда все вершины посещены.

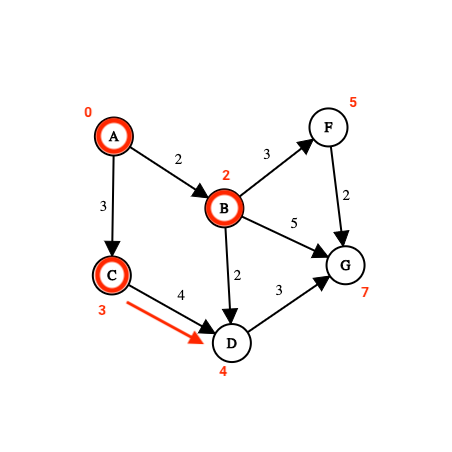
Из вершины A можно напрямую попасть в вершины B и C, длины маршрутов равны 2 и 3 соответственно. Записываем эти значения рядом с вершинами (здесь и далее – только в том случае, если новое значение метки меньше предыдущего значения). Вершину A отмечаем как посещенную.

Выбираем следующую вершину – такую, которая не является посещенной и имеет минимальное значение метки. Такой вершиной является вершина B.

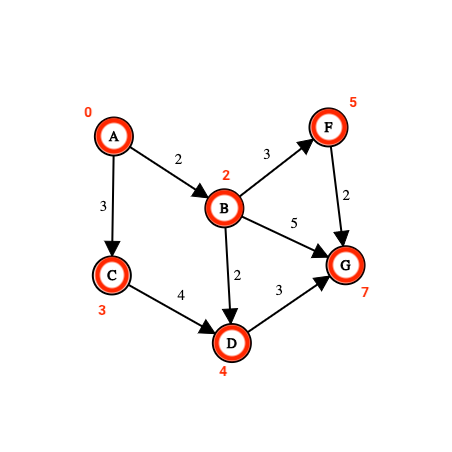
Из вершины B есть пути в вершины D, F и G. Новые значения меток около вершин рассчитываются как сумма метки около вершины B и значений весовых коэффициентов на соответствующих маршрутах – BD, BF и BG. Вершину B отмечаем как посещенную.



На следующем шаге снова выбираем вершину с наименьшим значением метки. Это вершина C. Из этой вершины есть только один путь – в вершину D. Сумма значения метки вершины C и весового коэффициента маршрута CD меньше значения метки вершины D, поэтому значение метки вершины D остается неизменным.



Далее поступаем аналогичным способом с оставшимися вершинами, пока все они не будут отмечены как посещенные. Значение метки вершины F не изменилось, так как суммы меток и весовых коэффициентов маршрутов DG и FG оказались равны актуальному значению метки.



Мы рассмотрели все возможные маршруты, алгоритм завершил работу.

Кратчайшие пути:

* из A в B = 2;
* из A в C = 3;
* из A в D = 4;
* из A в F = 5;
* из A в G = 7.

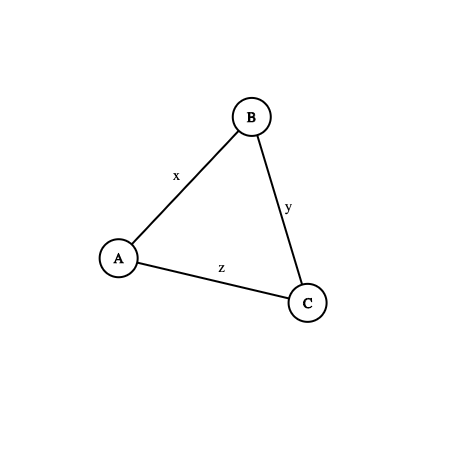
## **Алгоритм Флойда – Уоршелла**

Алгоритм Дейкстры решает задачу поиска кратчайшего пути только из одной заданной вершины. Чтобы решить эту задачу для всех вершин графа, можно применить алгоритм Дейкстры *n* раз, но существует более красивый способ – алгоритм Флойда – Уоршелла, основанный на той же идее сокращения маршрута.

Известно, что весовая матрица графа содержит информацию только о прямых связях между вершинами. Однако это вовсе не означает, что других путей между вершинами графов не существует. Прямой путь из одной вершины в другую не всегда является единственным и кратчайшим.

Рассмотрим пример. Возьмем первую вершину (например, B) и проверим, не будет ли маршрут из вершины A в вершину C короче, если идти через вершину B. Если сумма длин кратчайших маршрутов из вершины A в вершину B и из вершины B в вершину C окажется меньше, чем длина ребра AC, то использование «обхода» через вершину B позволит сократить путь.

Другими словами: если (x + y) < z, то нужно воспользоваться путем (x + y).



Если таким же образом перебрать последовательно все маршруты, то мы найдем все кратчайшие пути и возможные варианты сокращения пути от вершины к вершине. Полученные значения каждый раз обязательно запоминаются в весовой матрице только в том случае, если новое значение меньше предыдущего.

Решите задания 3.2 – 3.5 рабочего листа с помощью некоторых представленных в данном файле алгоритмов. Сравните полученные результаты.

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |