Алгоритм Дейкстры

Сожержание

- Формулировка задачи
- Алгоритм
- Иллюстрация алгоритма
- Время выполнения алгоритма

Формулировка задачи

Дан взвешенный ориентированный граф G(V,E) без дуг отрицательного веса. Найти кратчайшие пути от некоторой вершины a графа G до всех остальных вершин этого графа.

V — множество вершин графа

E — множество рёбер графа

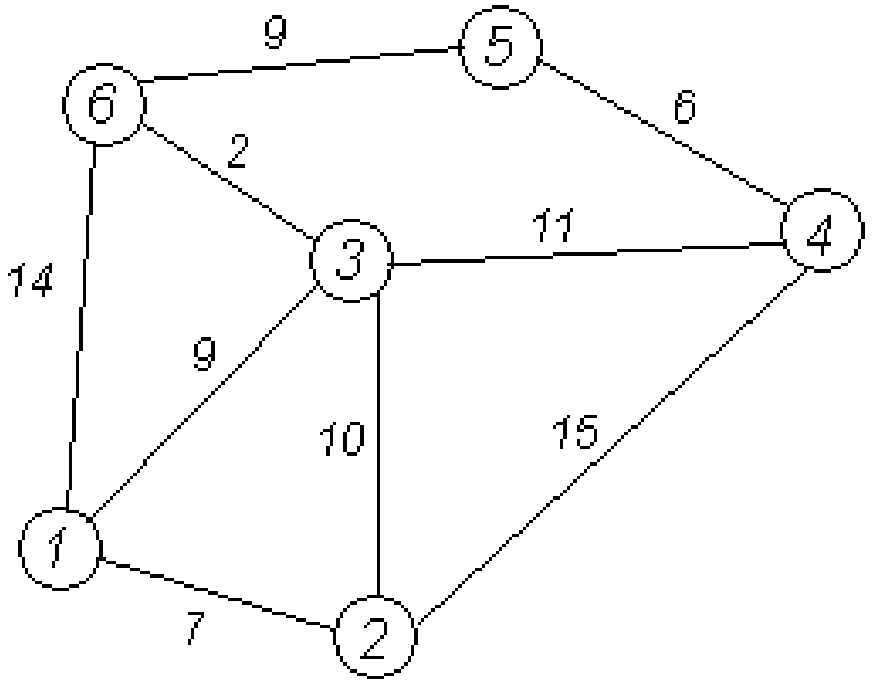
Алгоритм

Каждой вершине из V сопоставим метку — минимальное известное расстояние от этой вершины до a. Алгоритм работает пошагово — на каждом шаге он «посещает» одну вершину и пытается уменьшать метки. Работа алгоритма завершается, когда все вершины посещены.

Инициализация. Метка самой вершины a полагается равной 0, метки остальных вершин — бесконечности. Это отражает то, что расстояния от a до других вершин пока неизвестны. Все вершины графа помечаются как непосещённые.

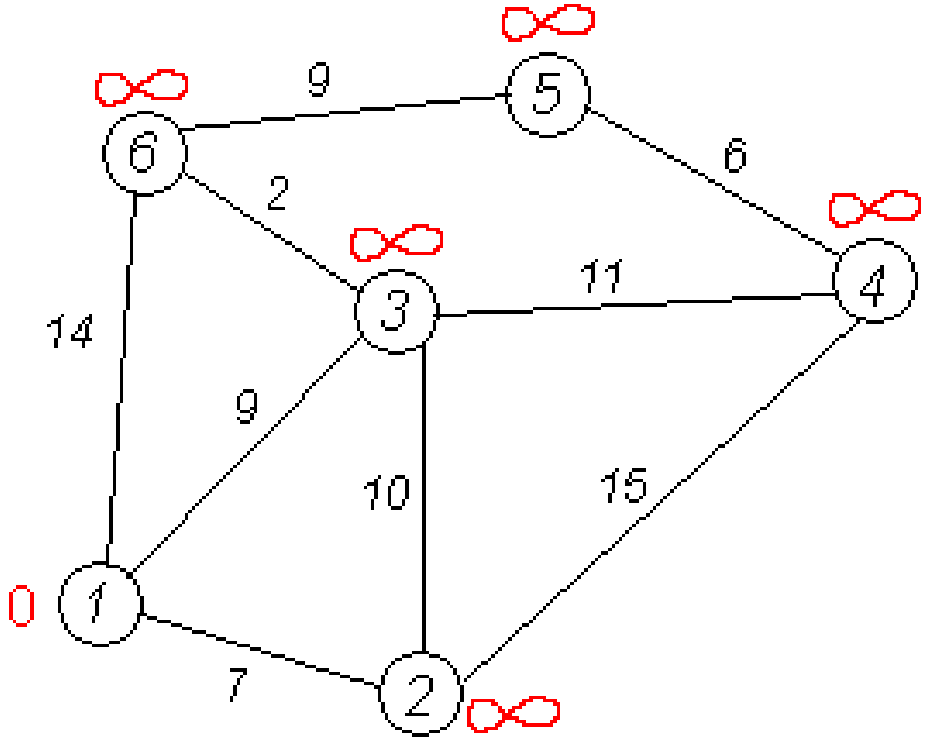
Шаг алгоритма. Если все вершины посещены, алгоритм завершается. В противном случае, из ещё не посещённых вершин выбирается вершина u, имеющая минимальную метку. Мы рассматриваем всевозможные маршруты, в которых u является предпоследним пунктом. Вершины, в которые ведут рёбра из u, назовём соседями этой вершины. Для каждого соседа вершины u, кроме отмеченных как посещённые, рассмотрим новую длину пути, равную сумме значений текущей метки u и длины ребра, соединяющего u с этим соседом. Если полученное значение длины меньше значения метки соседа, заменим значение метки полученным значением длины. Рассмотрев всех соседей, пометим вершину u как посещённую и повторим шаг алгоритма.

Иллюстрация агоритма



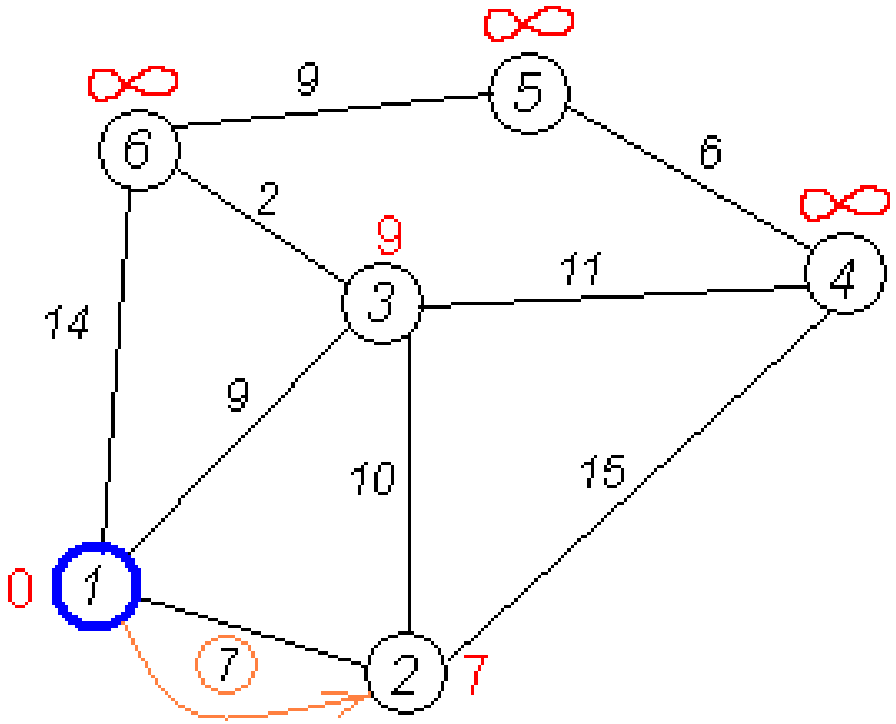
- Пусть требуется найти кратчайшие расстояния от 1-й вершины до всех остальных.

Иллюстрация агоритма



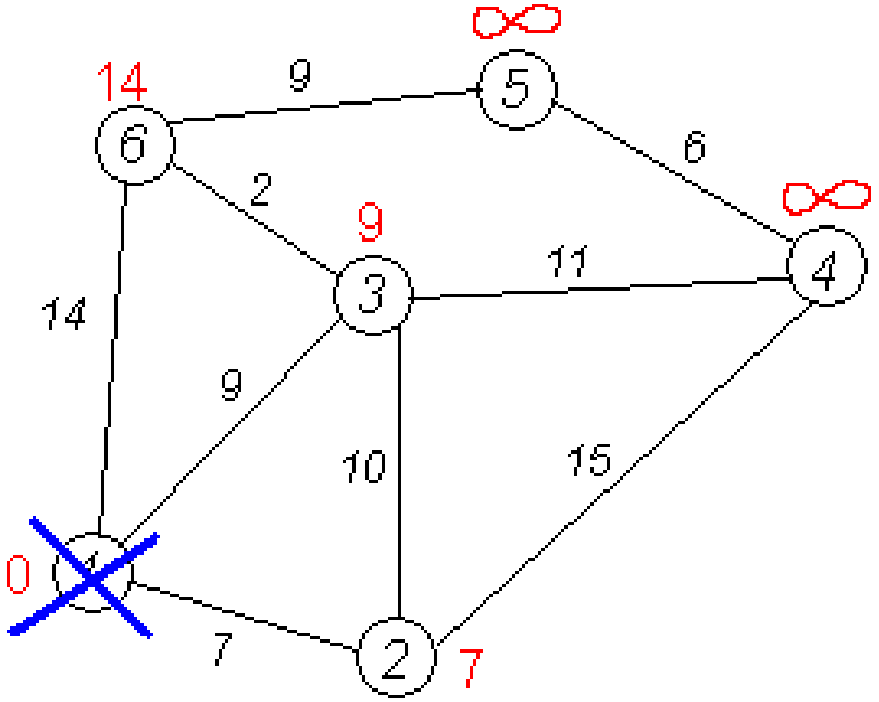
Кружками обозначены вершины, линиями — пути между ними (рёбра графа). В кружках обозначены номера вершин, над рёбрами обозначен их вес — длина пути. Рядом с каждой вершиной красным обозначена метка — длина кратчайшего пути в эту вершину из вершины 1.

Иллюстрация агоритма



Первый по очереди сосед вершины 1 — вершина 2, потому что длина пути до неё минимальна. Длина пути в неё через вершину 1 равна сумме значения метки вершины 1 и длины ребра, идущего из 1-й в 2-ю, то есть 0 + 7 = 7. Это меньше текущей метки вершины 2, бесконечности, поэтому новая метка 2-й вершины равна 7.

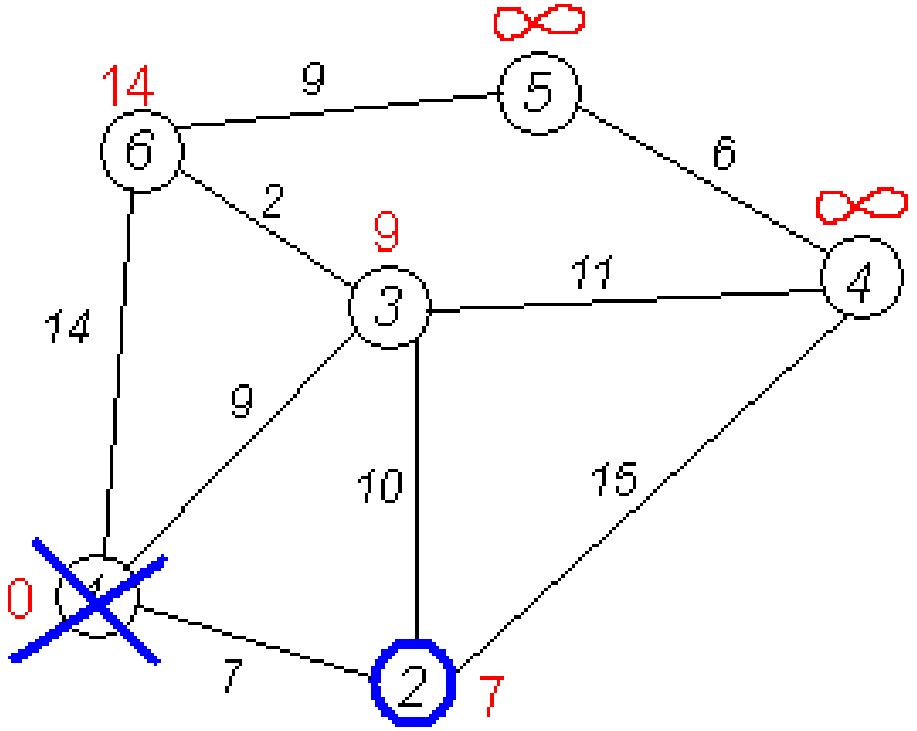
Иллюстрация агоритма



Аналогичную операцию проделываем с двумя другими соседями 1-й вершины — 3-й и 6-й

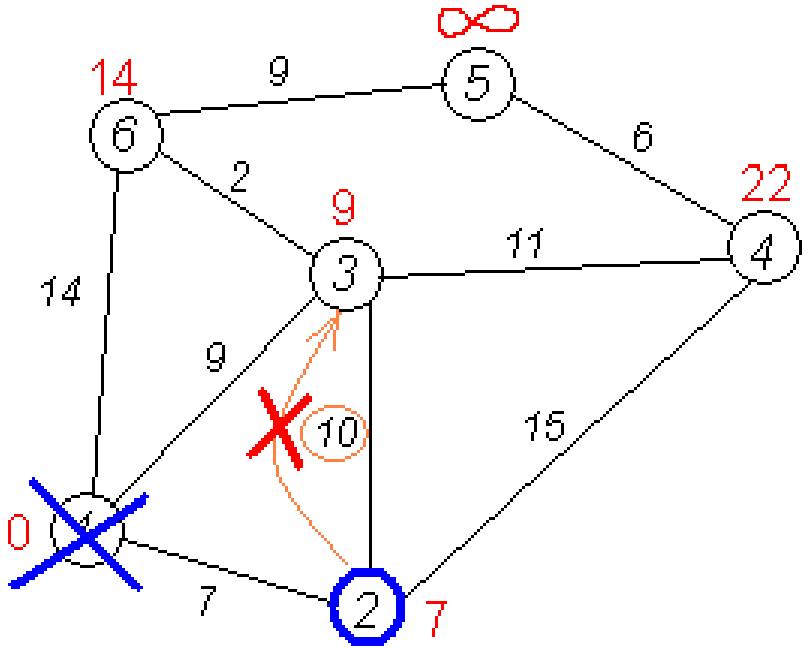
Все соседи вершины 1 проверены. Текущее минимальное расстояние до вершины 1 считается окончательным и пересмотру не подлежит (то, что это действительно так, впервые доказал Э. Дейкстра). Вычеркнем её из графа, чтобы отметить, что эта вершина посещена.

Иллюстрация агоритма



Шаг алгоритма повторяется. Снова находим «ближайшую» из непосещённых вершин. Это вершина 2 с меткой 7.

Иллюстрация агоритма

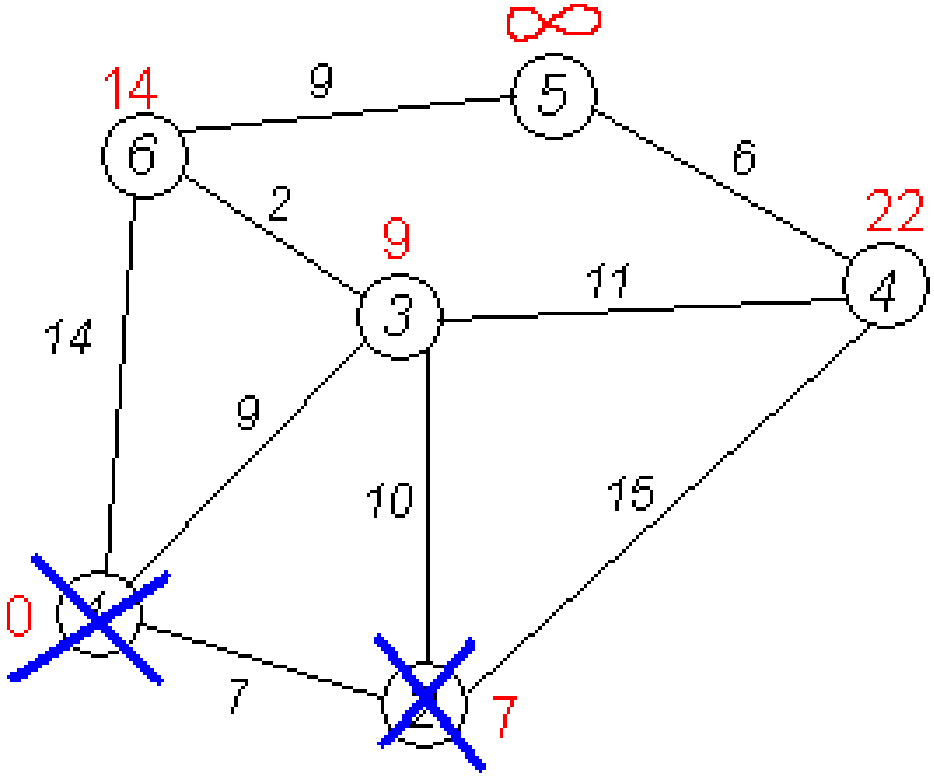


Снова пытаемся уменьшить метки соседей выбранной вершины, пытаясь пройти в них через 2 вершину. Соседями вершины 2 являются вершины 1, 3 и 4.

Первый (по порядку) сосед вершины 2 — вершина 1. Но она уже посещена, поэтому с 1-й вершиной ничего не делаем.

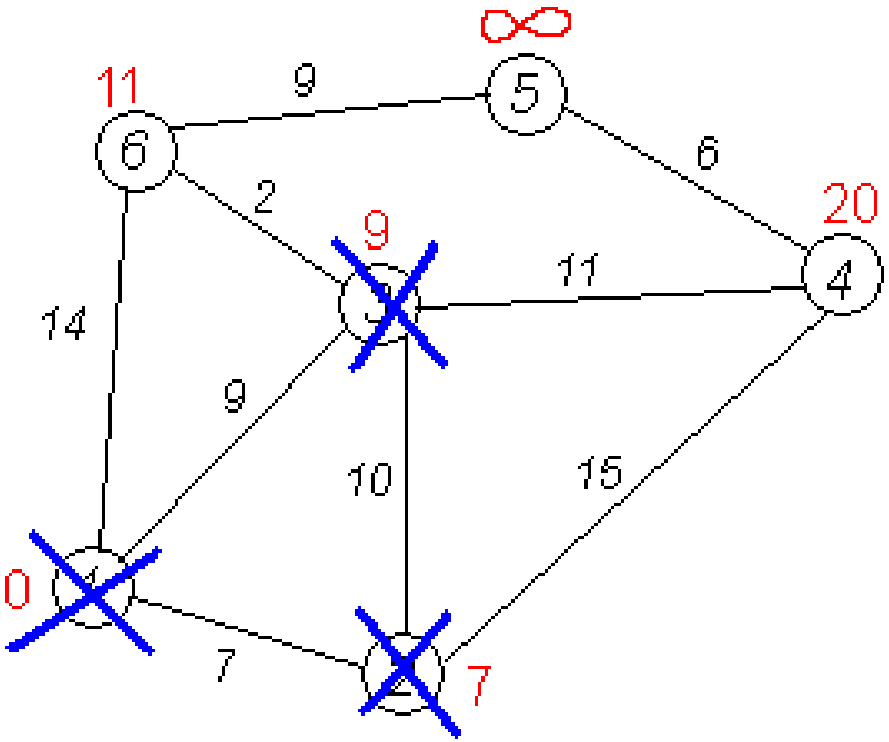
Следующий сосед вершины 2 — вершина 3, так как имеет минимальную метку из вершин, отмеченных как не посещённые. Если идти в неё через 2, то длина такого пути будет равна 17 (7 + 10 = 17). Но текущая метка третьей вершины равна 9, а это меньше 17, поэтому метка не меняется.

Иллюстрация агоритма



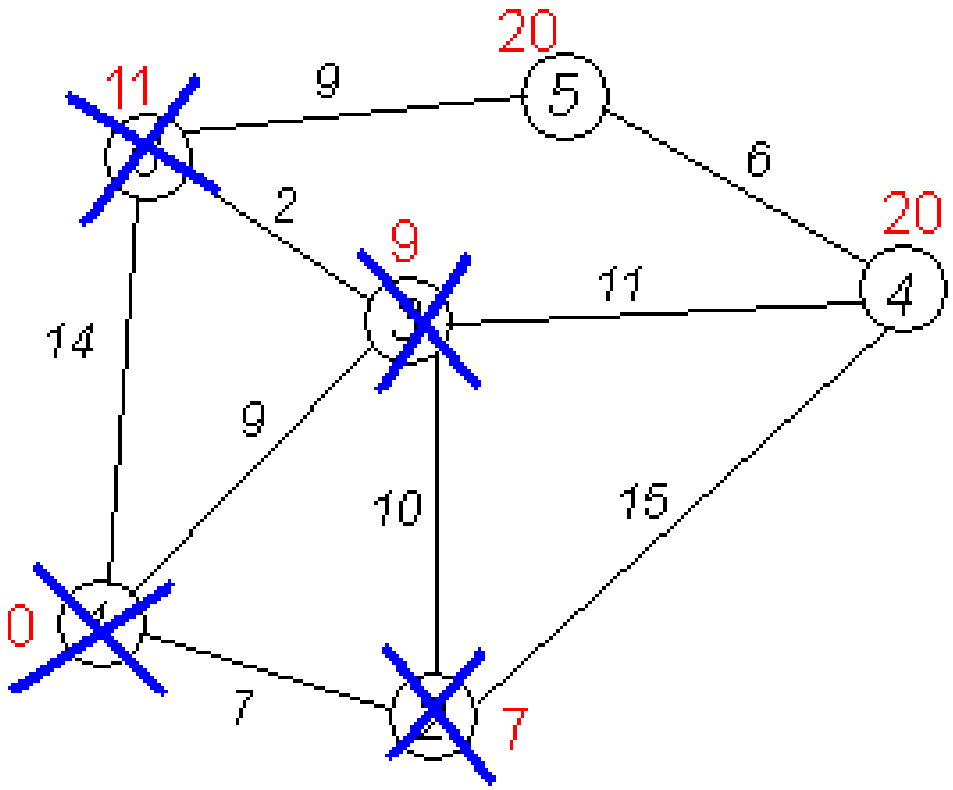
Ещё один сосед вершины 2 — вершина 4. Если идти в неё через 2-ю, то длина такого пути будет равна сумме кратчайшего расстояния до 2-й вершины и расстояния между вершинами 2 и 4, то есть 22 (7 + 15 = 22). Поскольку 22< {\displaystyle \infty } \infty , устанавливаем метку вершины 4 равной 22

Иллюстрация агоритма



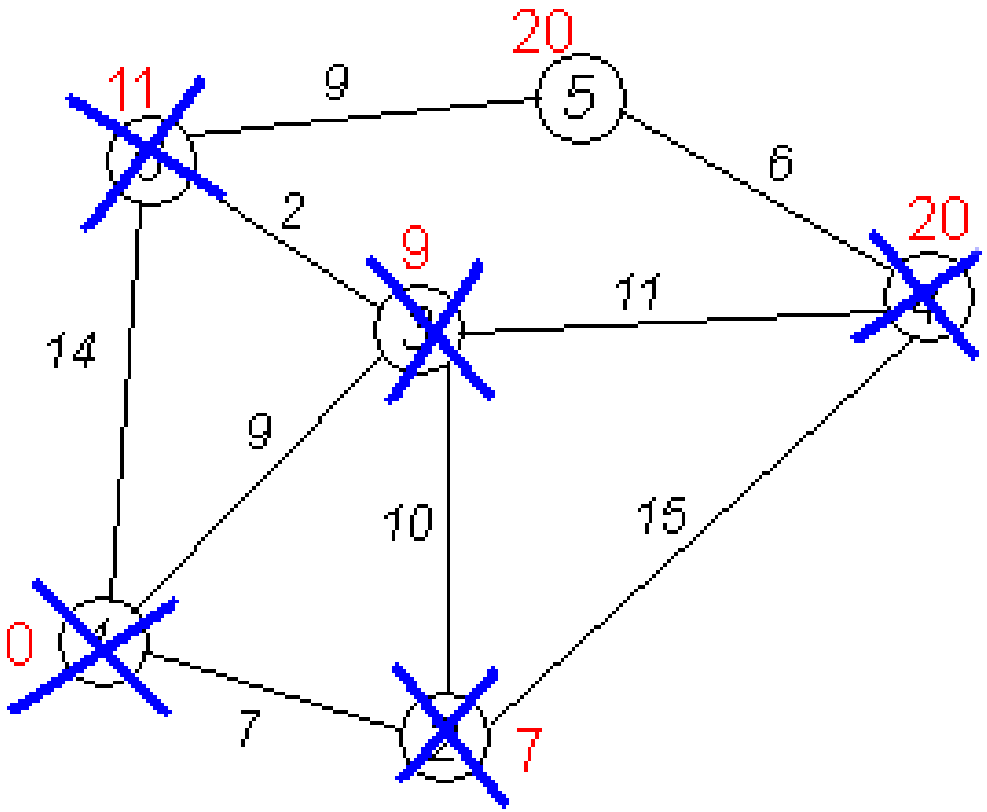
Повторяем шаг алгоритма, выбрав вершину 3.

Иллюстрация агоритма



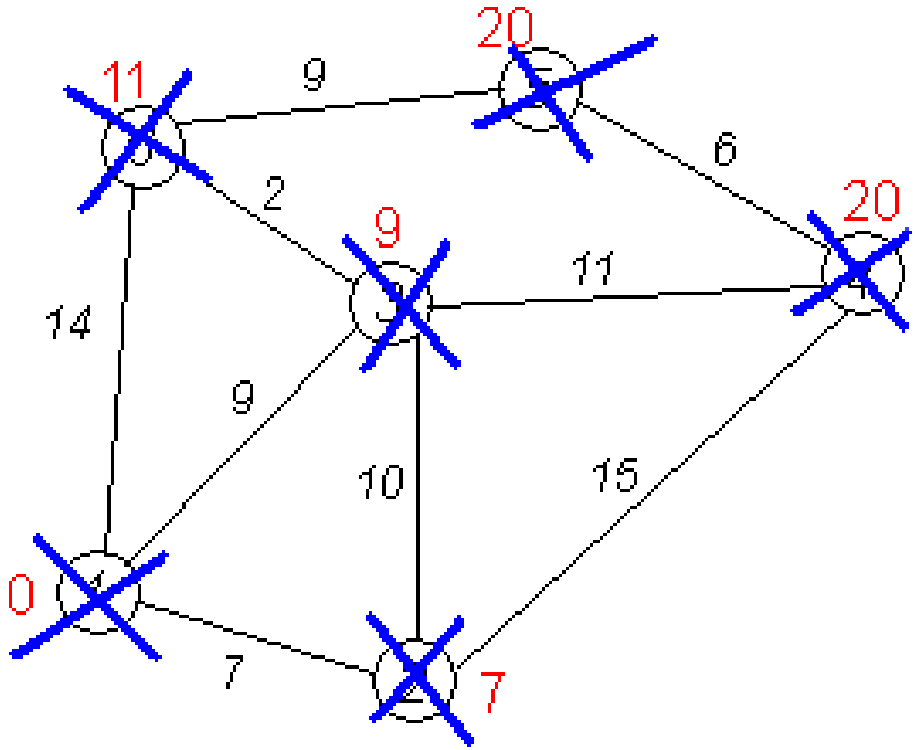
Повторяем шаг алгоритма, выбрав вершину 6.

Иллюстрация агоритма



Повторяем шаг алгоритма, выбрав вершину 4.

Илюстрация алгоритма



Завершение выполнения алгоритма. Алгоритм заканчивает работу, когда нельзя больше обработать ни одной вершины. В данном примере все вершины зачёркнуты, однако ошибочно полагать, что так будет в любом примере — некоторые вершины могут остаться незачёркнутыми, если до них нельзя добраться, то есть если граф несвязный. Результат работы алгоритма виден на последнем рисунке: кратчайший путь от вершины 1 до 2-й составляет 7, до 3-й — 9, до 4-й — 20, до 5-й — 20, до 6-й — 11.

Сложность алгоритма

- Сложность алгоритма Дейкстры зависит от способа нахождения вершины v, а также способа хранения множества непосещённых вершин и способа обновления меток. Обозначим через n количество вершин, а через m — количество рёбер в графе G.
- В простейшем случае, когда для поиска вершины с минимальным d[v] просматривается всё множество вершин, а для хранения величин d используется массив, время работы алгоритма есть $O(n^2)$. Основной цикл выполняется порядка n раз, в каждом из них на нахождение минимума тратится порядка n операций. На циклы по соседям каждой посещаемой вершины тратится количество операций, пропорциональное количеству рёбер m (поскольку каждое ребро встречается в этих циклах ровно дважды и требует константное число операций). Таким образом, общее время работы алгоритма $O(n^2+m)$, но, так как $m<n(n-1)$, оно составляет $O(n^2)$.

Сложность алгоритма

- Для разреженных графов (то есть таких, для которых m много меньше n^2) непосещённые вершины можно хранить в двоичной куче, а в качестве ключа использовать значения d[i], тогда время удаления вершины из U станет og n при том, что время модификации d[i] возрастёт до log (n). Так как цикл выполняется порядка n раз, а количество релаксаций (смен меток) не больше m, время работы такой реализации — O(n log (n)+m log (n)).