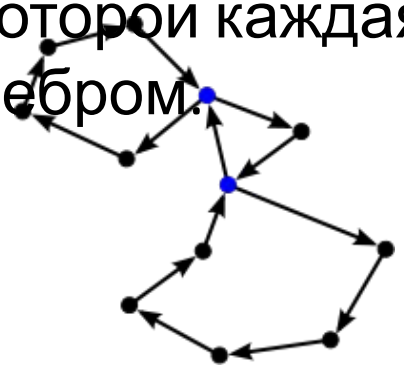
Кратчайшие пути из одной вершины в ориентированных ациклических графах. **Алгоритм Дейкстры**

Определения

- Ориентированный граф без циклов называется ***ориентированным ациклическим графом.***
- **Путь** в **графе** — последовательность вершин, в которой каждая вершина соединена со следующей ребром.



Ориентированный цикл. Без стрелок это просто цикл.

Постановка задачи

- В задаче поиска кратчайших путей полагаются известными множества вершин и ребер ориентированного или неориентированного графа G(V,E) (V – множество вершин, E – множество ребер), а также вес ребер, где значение веса выражается действительным числом.
- Существует ряд задач поиска кратчайших путей из одной вершины, отличающихся своей постановкой, где такие отличия состоят в следующем:

- является ли граф ориентированным или неориентированным;
- является ли граф ациклическим или содержит циклы;
- принимают ли веса ребер только положительные значения или возможны и их отрицательные значения;
- принимают ли веса ребер только целочисленные значения;
- выражаются ли веса ребер малыми неотрицательными значениями

Способы решения

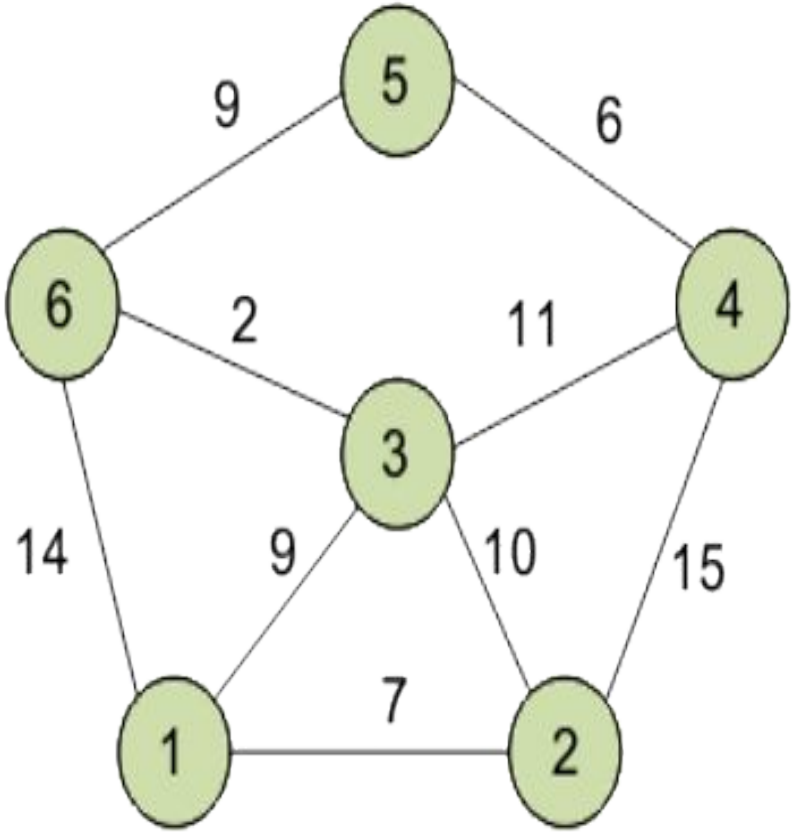
- **Алгоритм Беллмана – Форда** для общего случая, когда вес каждого из ребер может быть отрицательным, с трудоемкостью O(VxE), где дополнительно определяется наличие цикла с отрицательным весом;
- **Алгоритм поиска в ширину** для случая ориентированных ациклических графов с трудоемкостью O(V+E);
- **Алгоритм Дейкстры** для произвольных графов с неотрицательными весами ребер, где трудоемкость определяется в зависимости от способа выборки помеченной вершины с минимальным весом и при использовании пирамид Фибоначчи может достигать величины порядка O(VlgV+E).

Рассмотрим пример нахождение кратчайшего пути. Дана сеть автомобильных дорог, соединяющих области города. Некоторые дороги односторонние. **Найти кратчайшие пути от центра города до каждого города области**.

Для решения указанной задачи можно использовать *алгоритм Дейкстры* — алгоритм на графах, изобретённый нидерландским ученым Э. Дейкстрой в 1959 году. Находит кратчайшее расстояние от одной из вершин графа до всех остальных. Работает только для графов без рёбер отрицательного веса.

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

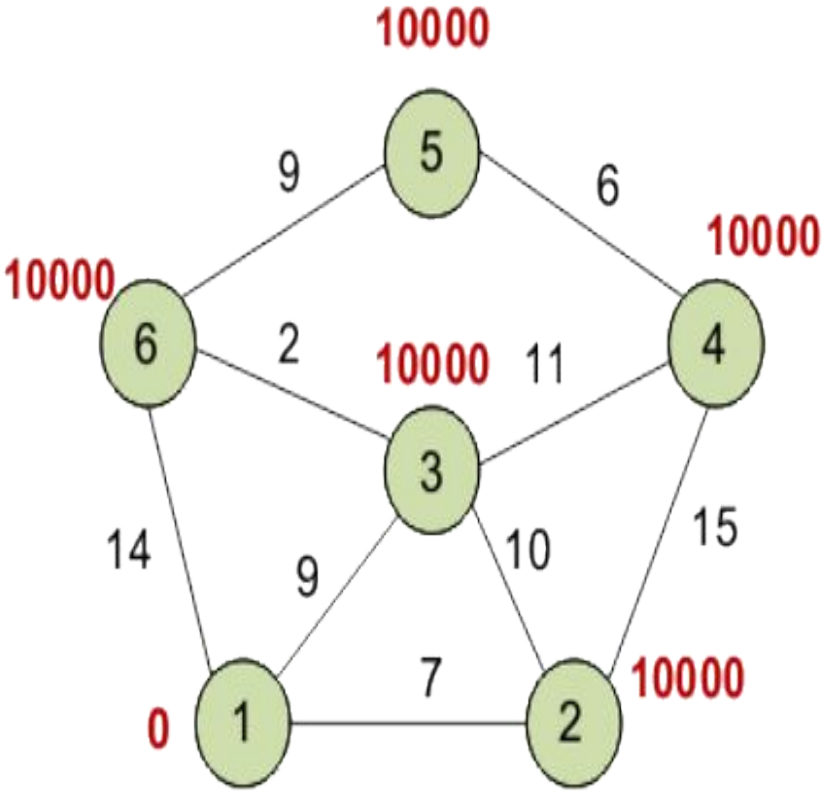
Кружками обозначены вершины, линиями – пути между ними (ребра графа). В кружках обозначены номера вершин, над ребрами обозначен их вес – длина пути. Рядом с каждой вершиной красным обозначена метка – длина кратчайшего пути в эту вершину из вершины 1.



Инициализация

Метка самой вершины 1 полагается равной 0, метки остальных вершин – недостижимо большое число (в идеале — бесконечность).

Это отражает то, что расстояния от вершины 1 до других вершин пока неизвестны. Все вершины графа помечаются как **непосещенные**.



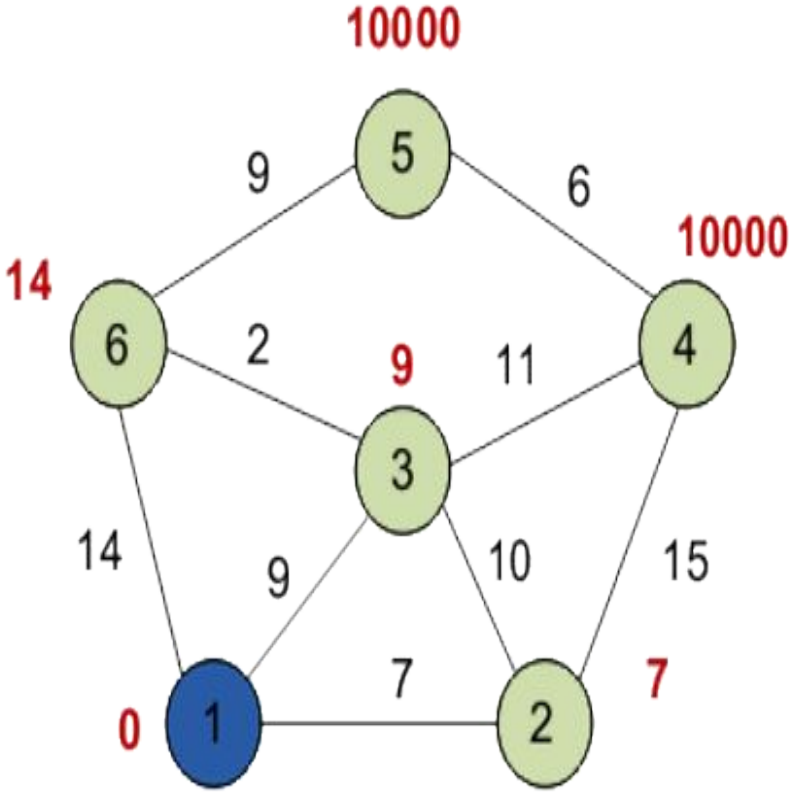
Первый шаг

Минимальную метку имеет вершина 1. Её соседями являются вершины 2, 3 и 6. Обходим соседей вершины по очереди.

Первый сосед вершины 1 – вершина 2, потому что длина пути до неё минимальна. Длина пути в неё через вершину 1 равна сумме кратчайшего расстояния до вершины 1, значению её метки, и длины ребра, идущего из 1-й в 2-ю, то есть 0 + 7 = 7. Это меньше текущей метки вершины 2 (10000), поэтому новая метка 2-й вершины равна 7.

Аналогично находим длины пути для всех других соседей (вершины 3 и 6).

Все соседи вершины 1 проверены. Текущее минимальное расстояние до вершины 1 считается окончательным и пересмотру не подлежит. Вершина 1 отмечается как посещенная.



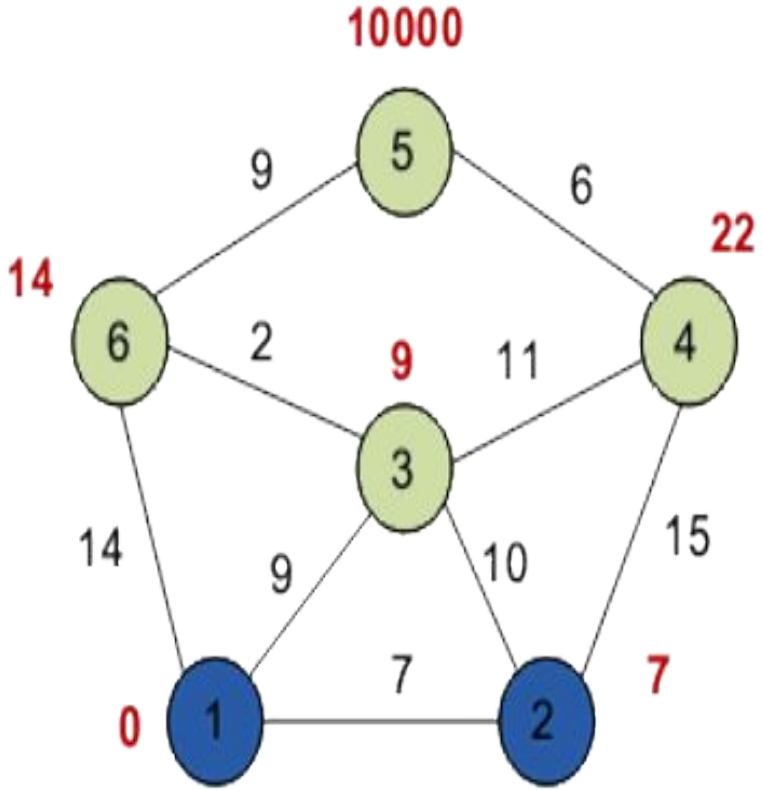
Второй шаг

- Шаг 1 алгоритма повторяется. Снова находим «ближайшую» из непосещенных вершин. Это вершина 2 с меткой 7.

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

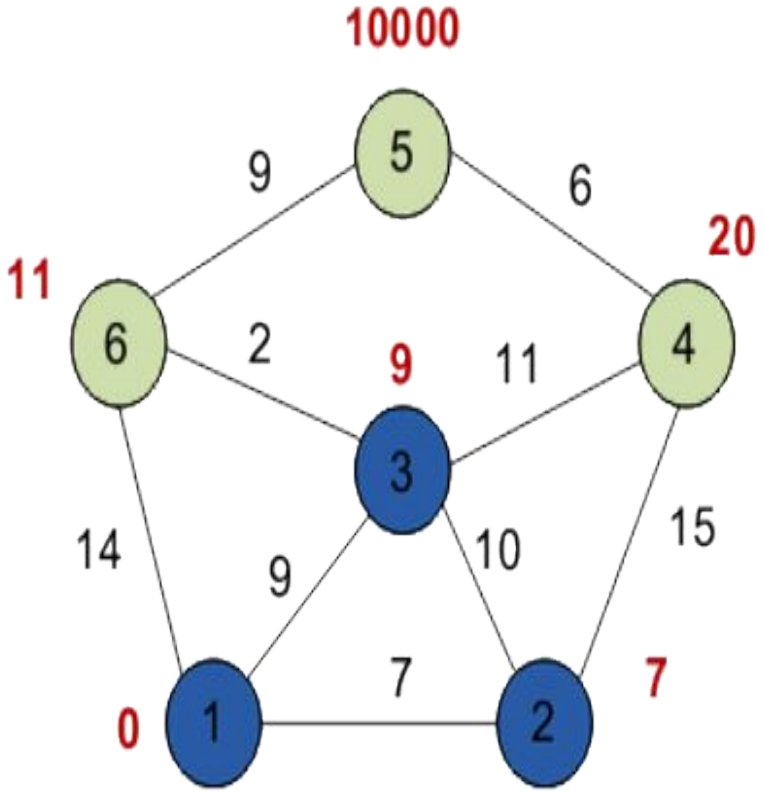
  Вершина 1 уже посещена. Следующий сосед вершины 2 — вершина 3, так как имеет минимальную метку из вершин, отмеченных как не посещённые. Если идти в неё через 2, то длина такого пути будет равна 17 (7 + 10 = 17). Но текущая метка третьей вершины равна 9, а 9 < 17, поэтому метка не меняется.
- Ещё один сосед вершины 2 — вершина 4. Если идти в неё через 2-ю, то длина такого пути будет равна 22 (7 + 15 = 22). Поскольку 22<10000, устанавливаем метку вершины 4 равной 22.

  Все соседи вершины 2 просмотрены, помечаем её как посещенную.

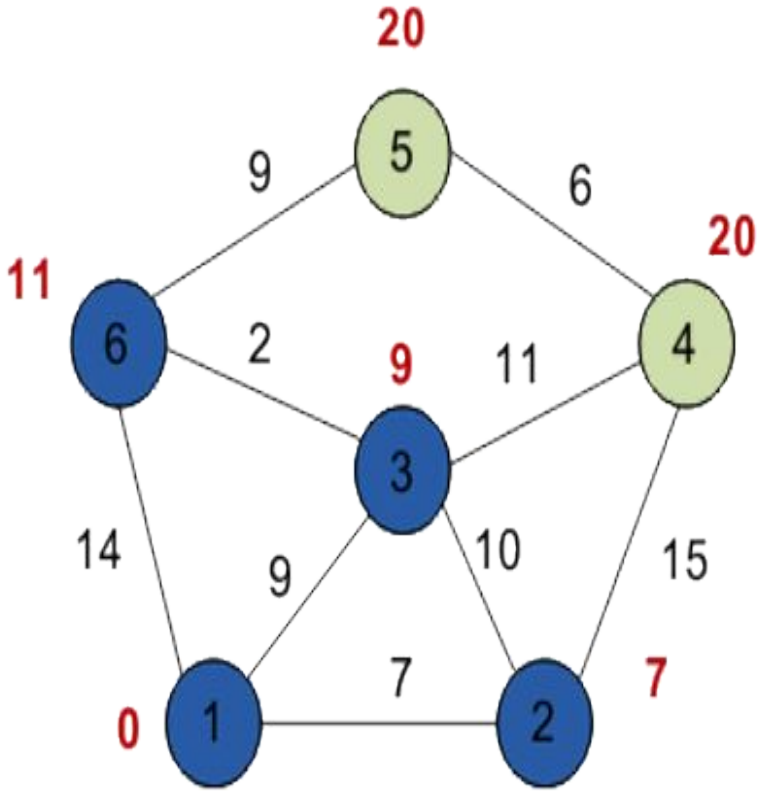


Третий шаг

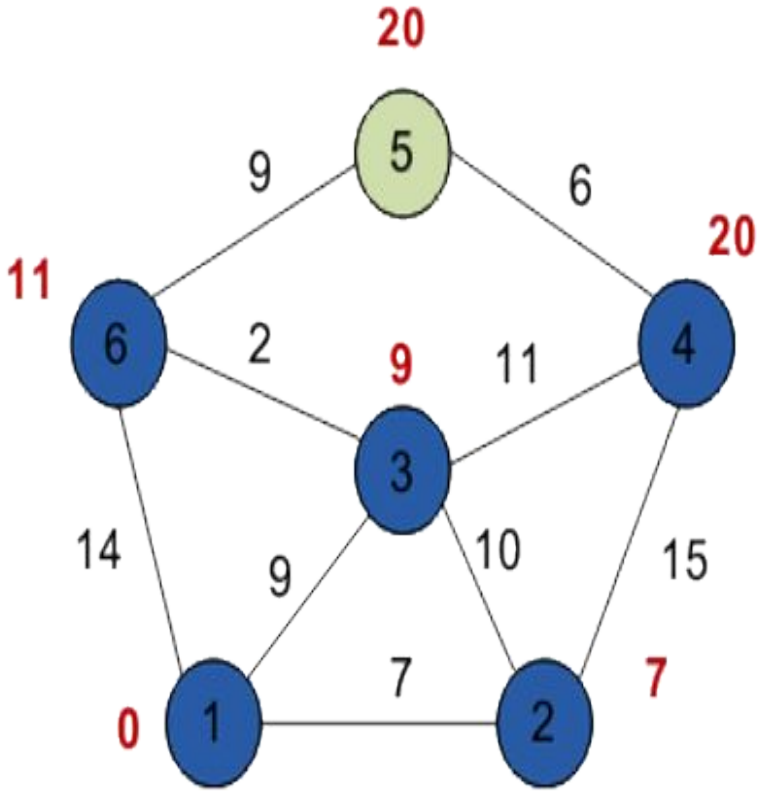
- Повторяем шаг алгоритма, выбрав вершину 3. После её «обработки» получим следующие результаты.



Четвертый шаг

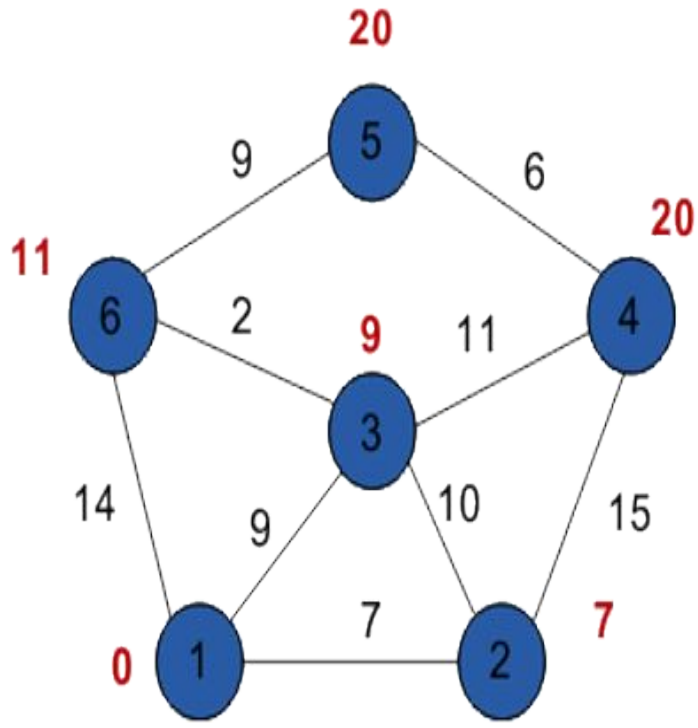


Пятый шаг



Шестой шаг

Таким образом, **кратчайшим путем из вершины 1 в вершину 5** будет путь через вершины **1 — 3 — 6 — 5**, поскольку таким путем мы набираем минимальный вес, равный 20.



Вывод кратчайшего пути

Займемся выводом кратчайшего пути. Мы знаем длину пути для каждой вершины, и теперь будем рассматривать вершины с конца. Рассматриваем конечную вершину (в данном случае — вершина **5**), и для всех вершин, с которой она связана, находим длину пути, вычитая вес соответствующего ребра из длины пути конечной вершины.

Так, вершина **5** имеет длину пути **20**. Она связана с вершинами **6** и **4**. Для вершины **6** получим вес **20 — 9 = 11 (совпал)**. Для вершины **4** получим вес **20 — 6 = 14 (не совпал)**.

Если в результате мы получим значение, которое совпадает с длиной пути рассматриваемой вершины (в данном случае — вершина **6**), то именно из нее был осуществлен переход в конечную вершину.

Отмечаем эту вершину на искомом пути. Далее определяем ребро, через которое мы попали в вершину **6**. И так пока не дойдем до начала.

Если в результате такого обхода у нас на каком-то шаге совпадут значения для нескольких вершин, то можно взять любую из них — несколько путей будут иметь одинаковую длину.

Реализация алгоритма Дейкстры

- Для хранения **весов графа** используется **квадратная матрица**. В заголовках строк и столбцов находятся вершины графа. А веса дуг графа размещаются во внутренних ячейках таблицы. Граф не содержит **петель**, поэтому на **главной диагонали** матрицы содержатся нулевые значения.

	1	2	3	4	5	6
1	0	7	9	0	0	14
2	7	0	10	15	0	0
3	9	10	0	11	0	2
4	0	15	11	0	6	0
5	0	0	0	6	0	9
6	14	0	2	0	9	0