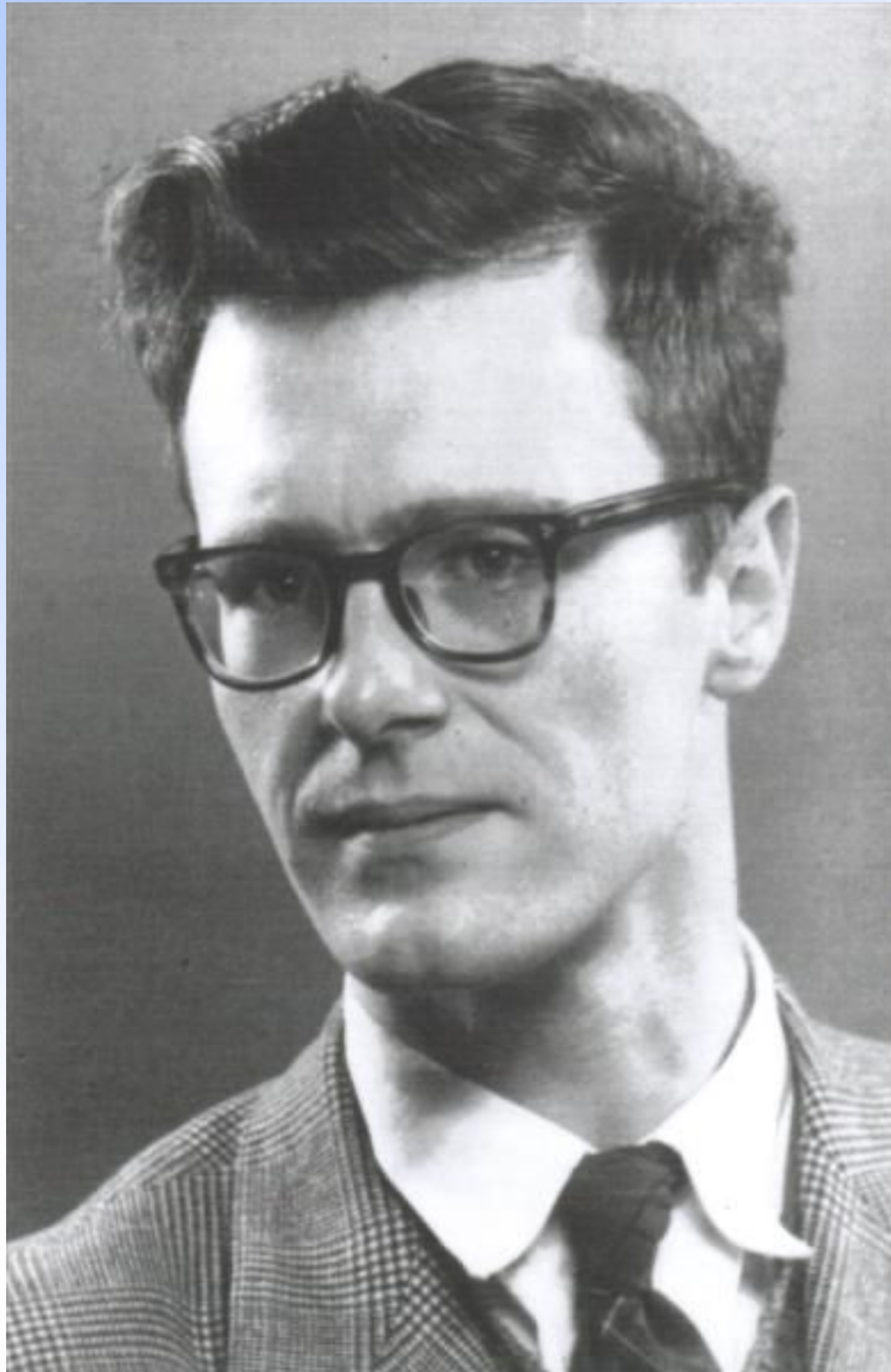


Алгоритм Дейкстры

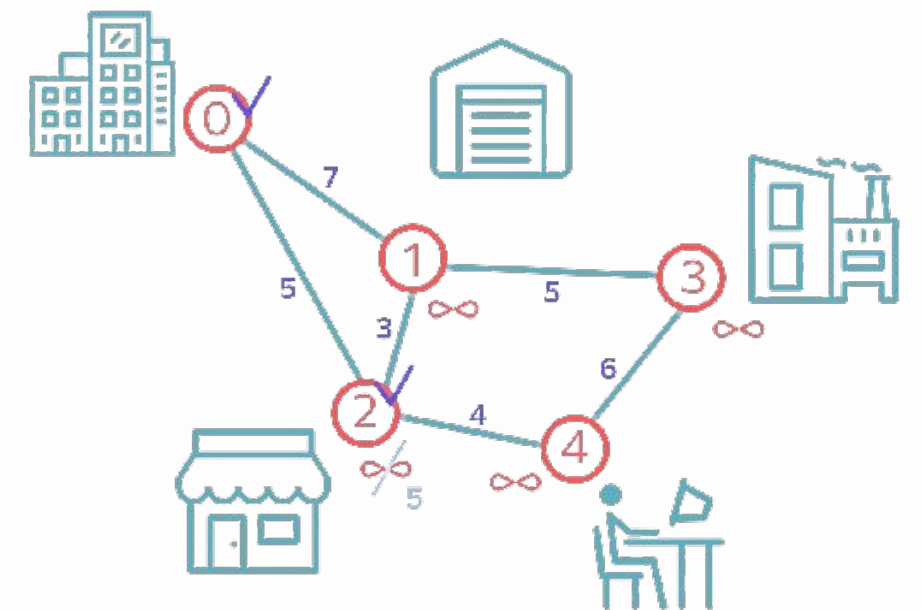
Гр.321702:

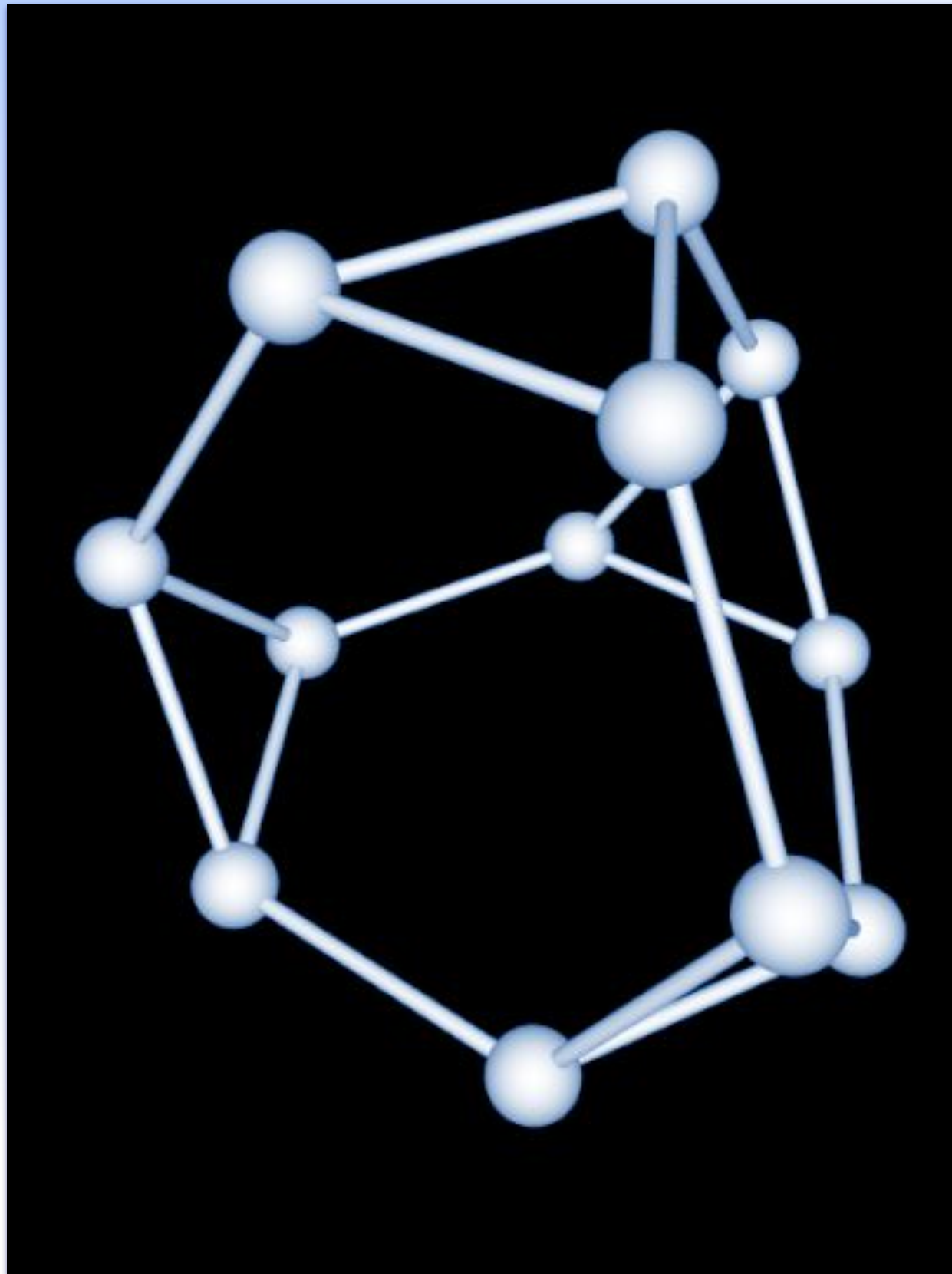
Рублевская Катя, Банкевич Яна, Сергиевич Дарья



Алгоритм поиска кратчайшего пути разработал голландский учёный Эдсгер Дейкстра в 1956 году. В то время он искал способ продемонстрировать возможности нового компьютера **ARMAC** и искал задачу, которую мог бы решить **ARMAC** и при этом понятную незнакомым с компьютерами людям.

Дейкстр взял задачу поиска кратчайшего пути и разработал алгоритм её решения. На базе алгоритма он разработал программу построения маршрутов между городами по транспортной карте Нидерландов.





Алгоритм Дейкстры – это метод, который находит кратчайший путь от одной вершины графа к другой.

Граф — это математическая структура, которая состоит из вершин (узлов) и рёбер (связей) между ними. Его можно представить как схему дорог или как компьютерную сеть.

Рёбра – это связи, по которым можно двигаться от одной вершине к другой. Они могут иметь направление, а также веса — числа, которые обозначают силу связей с вершинами.

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

Цель использования:

- при планировании автомобильных и авиамаршрутов,
- при разводке электронных плат,
- в протоколах маршрутизации,
- в планировании сервисов,
- при создании крупных карт.



Цель использования:

- поиск системой бронирования наиболее быстрых или дешевых билетов, в т.ч. с возможными пересадками;
- разработка поведения неигровых персонажей, создание игрового ИИ в геймдеве;
- моделирование движения робота, который перемещается по местности;



Кто использует алгоритм Дейкстры:

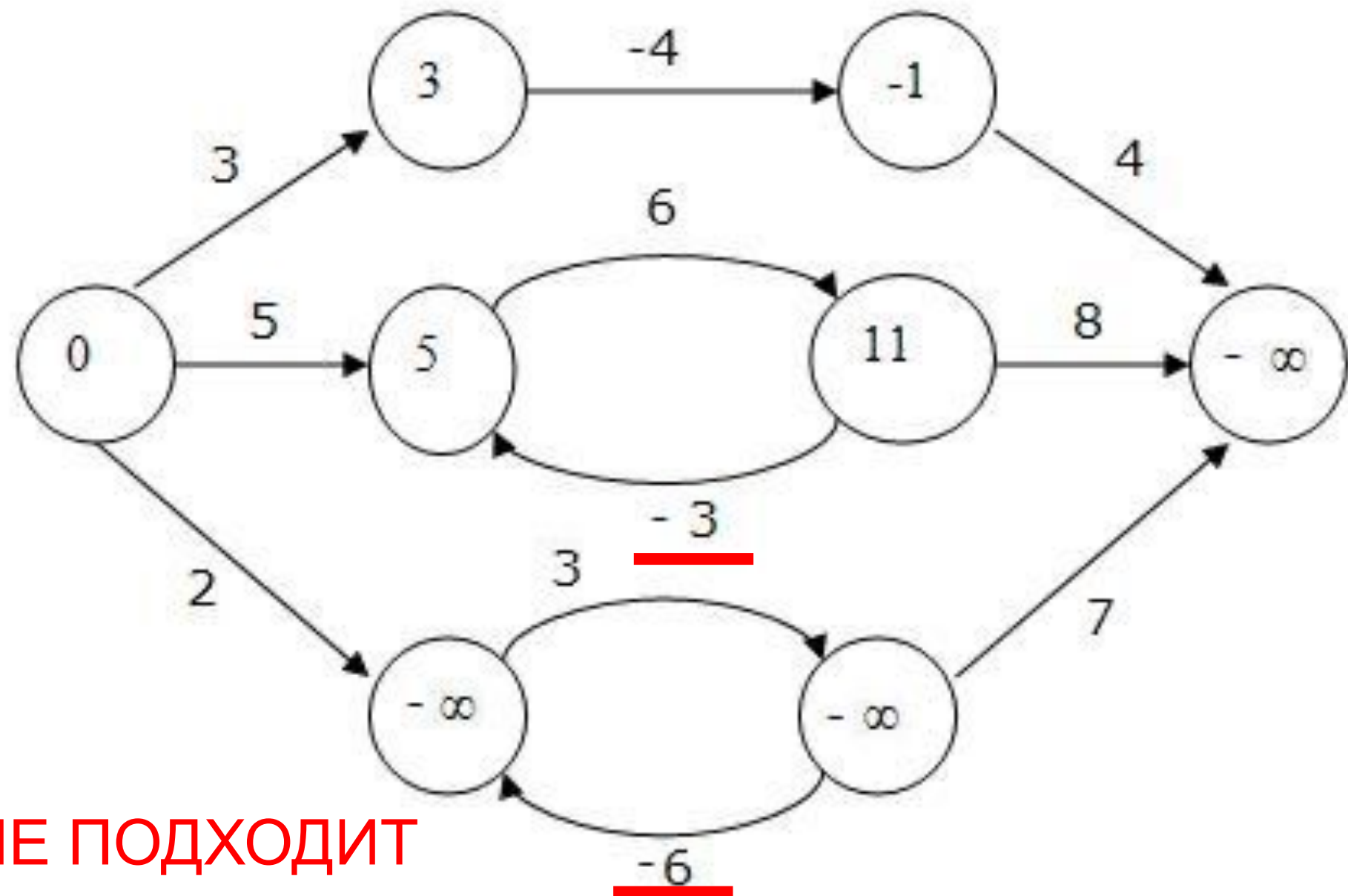
- **Математики и другие ученые**, которые пользуются графами как абстрактными единицами. Задача поиска маршрута в науке может быть и чисто фундаментальной, и прикладной.
- **Дата-сайентисты**. В этой области много математики, в том числе активно используется теория графов.
- **Сетевые инженеры**, так как алгоритм Дейкстры лежит в основе работы нескольких протоколов маршрутизации. Сама по себе компьютерная сеть представляет собой граф, поэтому специалисты по сетям должны знать, что это такое.

Важно:

Нужно понимать:

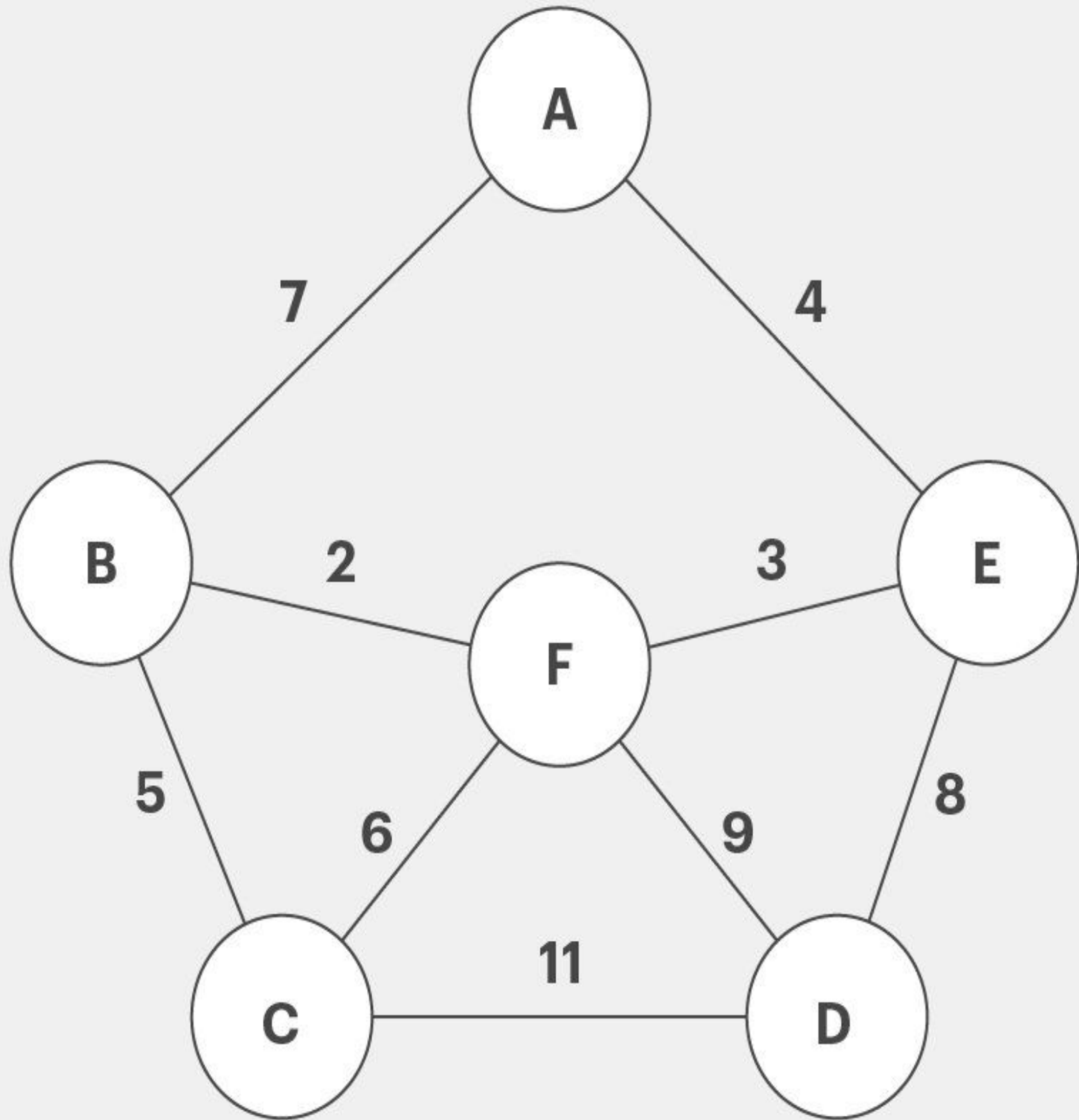
- он работает только со взвешенными графами.
- эти расстояния должны быть неотрицательными.
- Если алгоритм нашел два равных кратчайших пути:
 - Или на выходе будут оба (модифицированный алгоритм)
 - Или найдется один из (классический алгоритм)

Взвешенный граф — граф, каждому ребру которого поставлено в соответствие некое значение (вес ребра).



НЕ ПОДХОДИТ

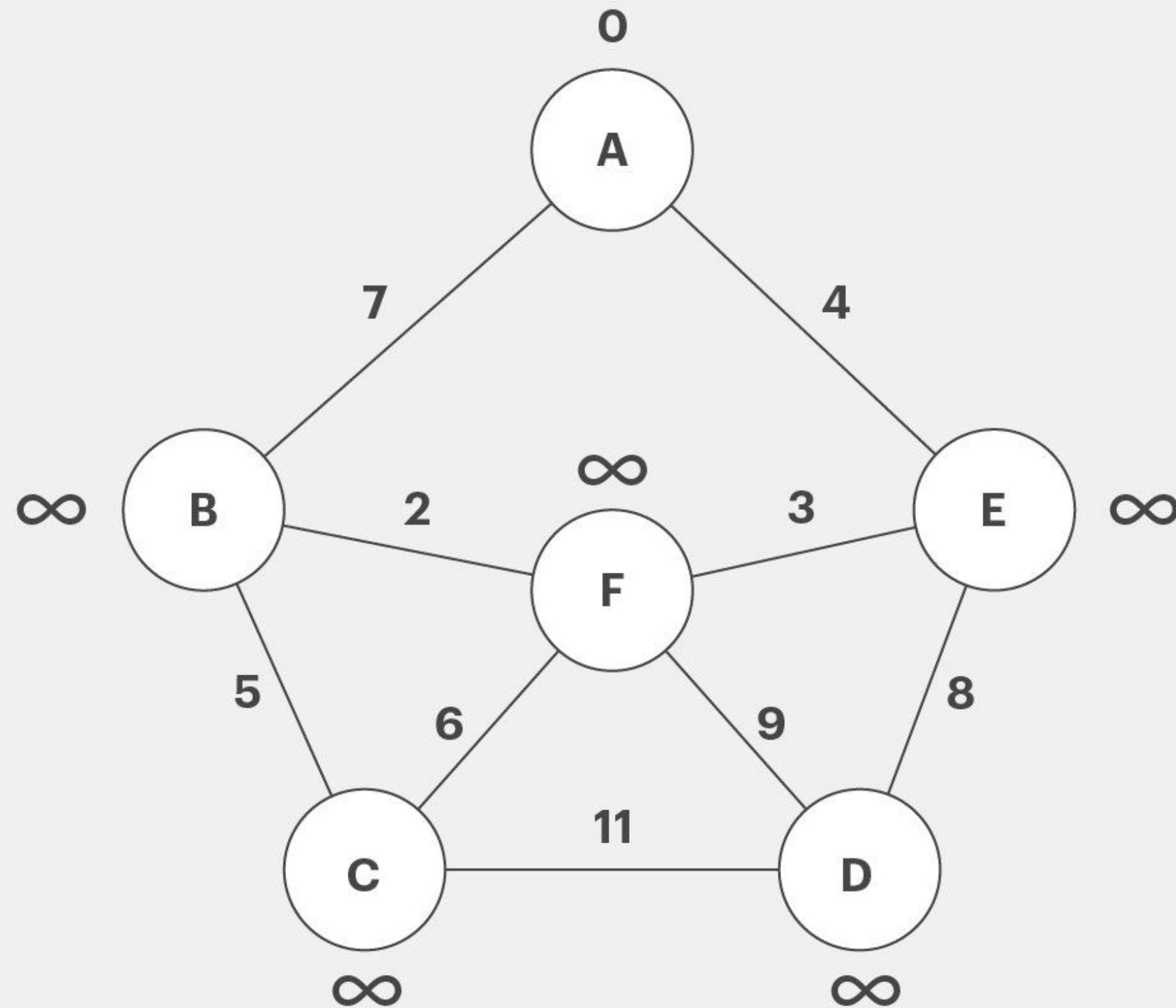
Как работает алгоритм



Идея алгоритма Дейкстры в том, что мы можем найти наименьшие расстояния от начальной вершины графа ко всем остальным. Зная эти расстояния, можно построить кратчайший маршрут между начальной и другими точками.

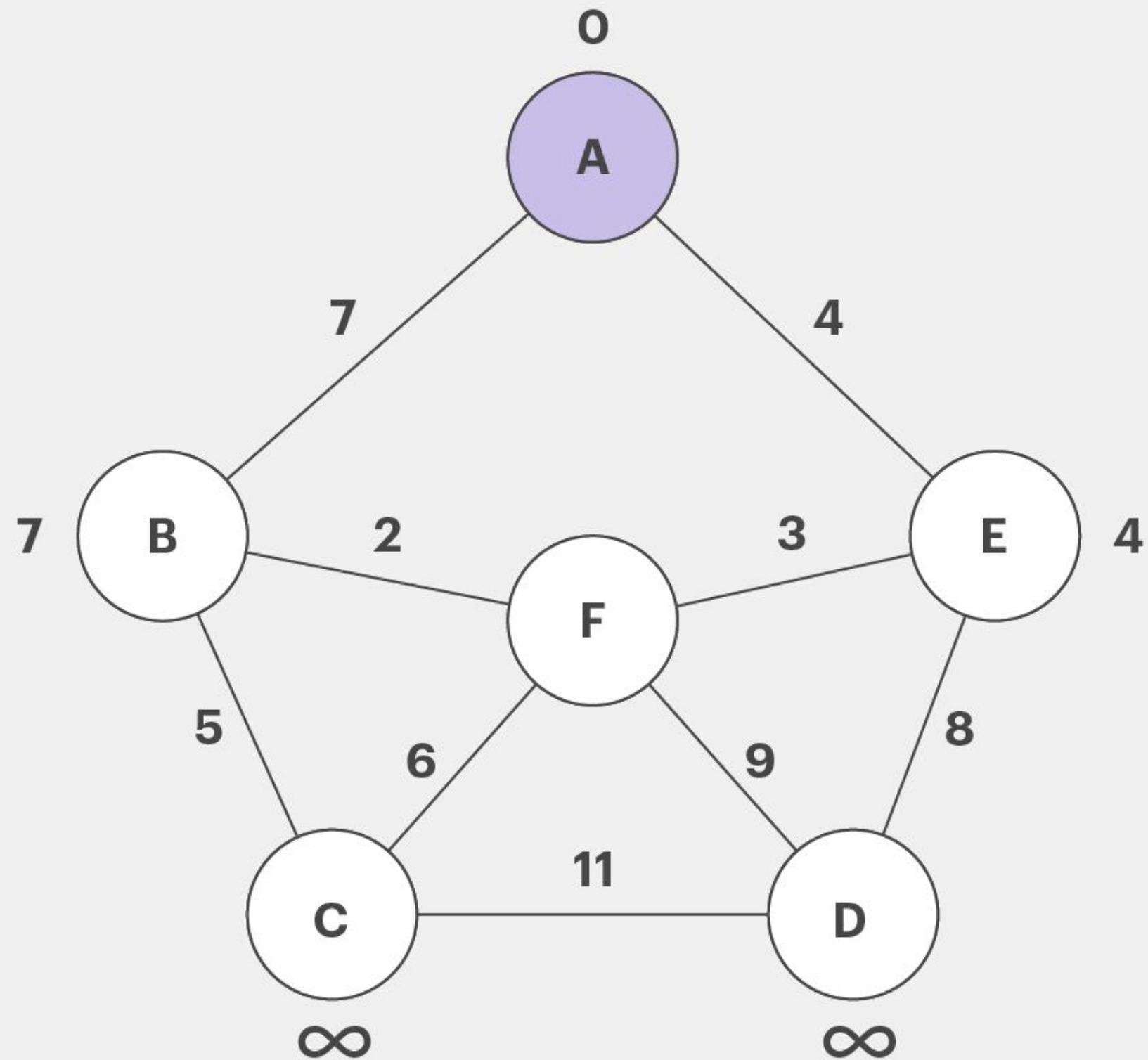
Допустим, у нас есть несколько городов, соединённых дорогами. **Назовём их А, В, С, D, E, F.** Числа возле рёбер — расстояния между городами в милях.

Шаг 1:



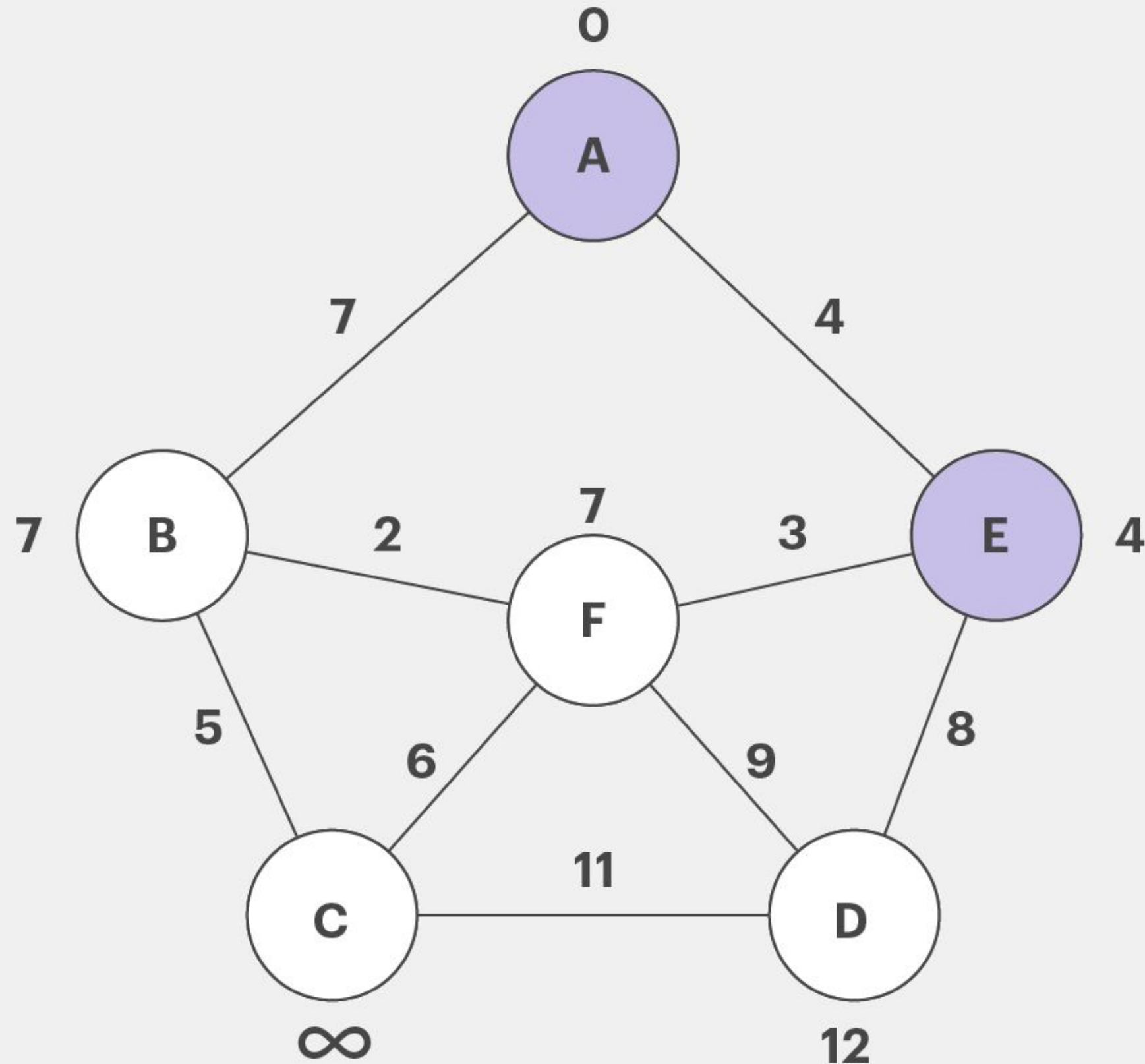
Установим для каждой вершины первоначальную оценку пути до A. **Для самой A оценка равна 0, остальным присвоим бесконечность**, так как мы пока не знаем их значения.

Шаг 2:



Рассмотрим соседние с A вершины — то есть те, которые связаны с A рёбрами напрямую. Это **B** и **E** — их расстояния до A равны 7 и 4 соответственно. Так как эти значения, очевидно, меньше бесконечности, обновим их на схеме. Вершину **A** будем считать **посещённой** — закрасим её и больше не рассматриваем.

Шаг 3:

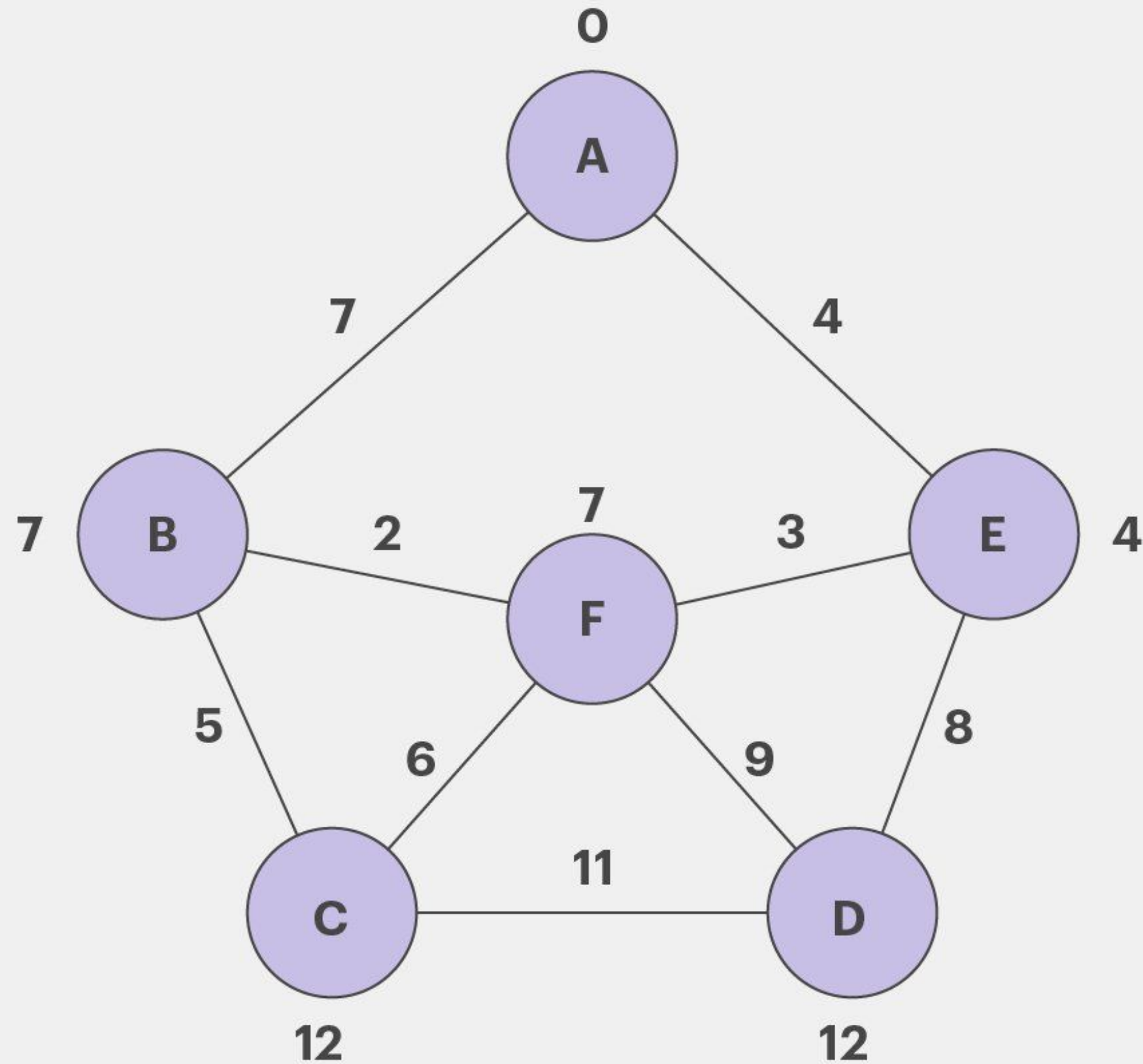


Теперь перейдём к непосещённой вершине с наименьшим расстоянием до **A**. Это вершина **E**. Соседние с ней непосещённые вершины — **F** и **D**. Их расстояния до A будут равны оценке E (то есть расстоянию от E до A) плюс веса рёбер от E до этих вершин. Получается так:

- Для F: $4 + 3 = 7$
- Для D: $4 + 8 = 12$

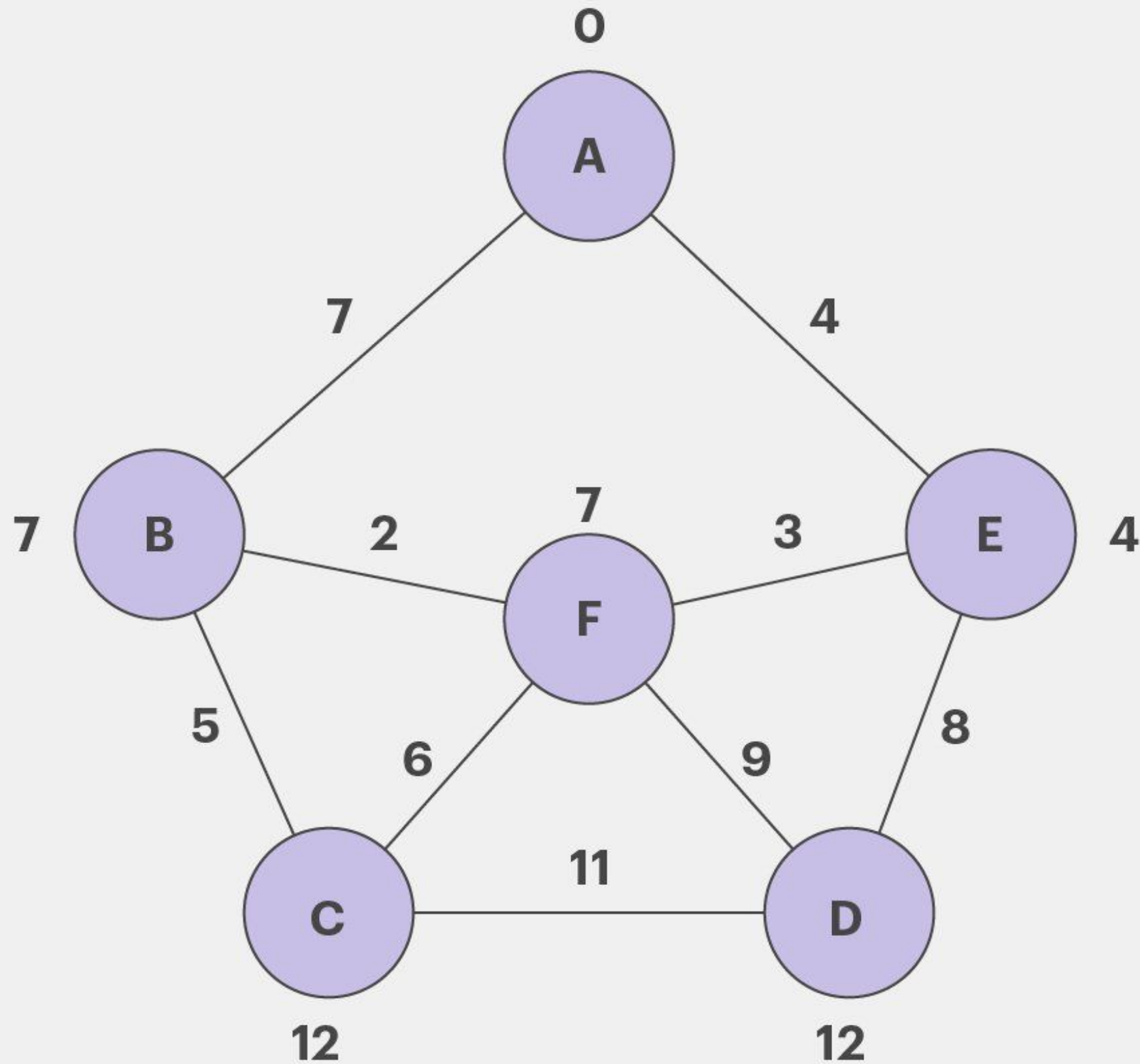
Полученные расстояния меньше предыдущих оценок, поэтому запишем их возле вершин F и D. Вершину **E** будем считать посещённой. Закрасим её.

Остальные шаги :



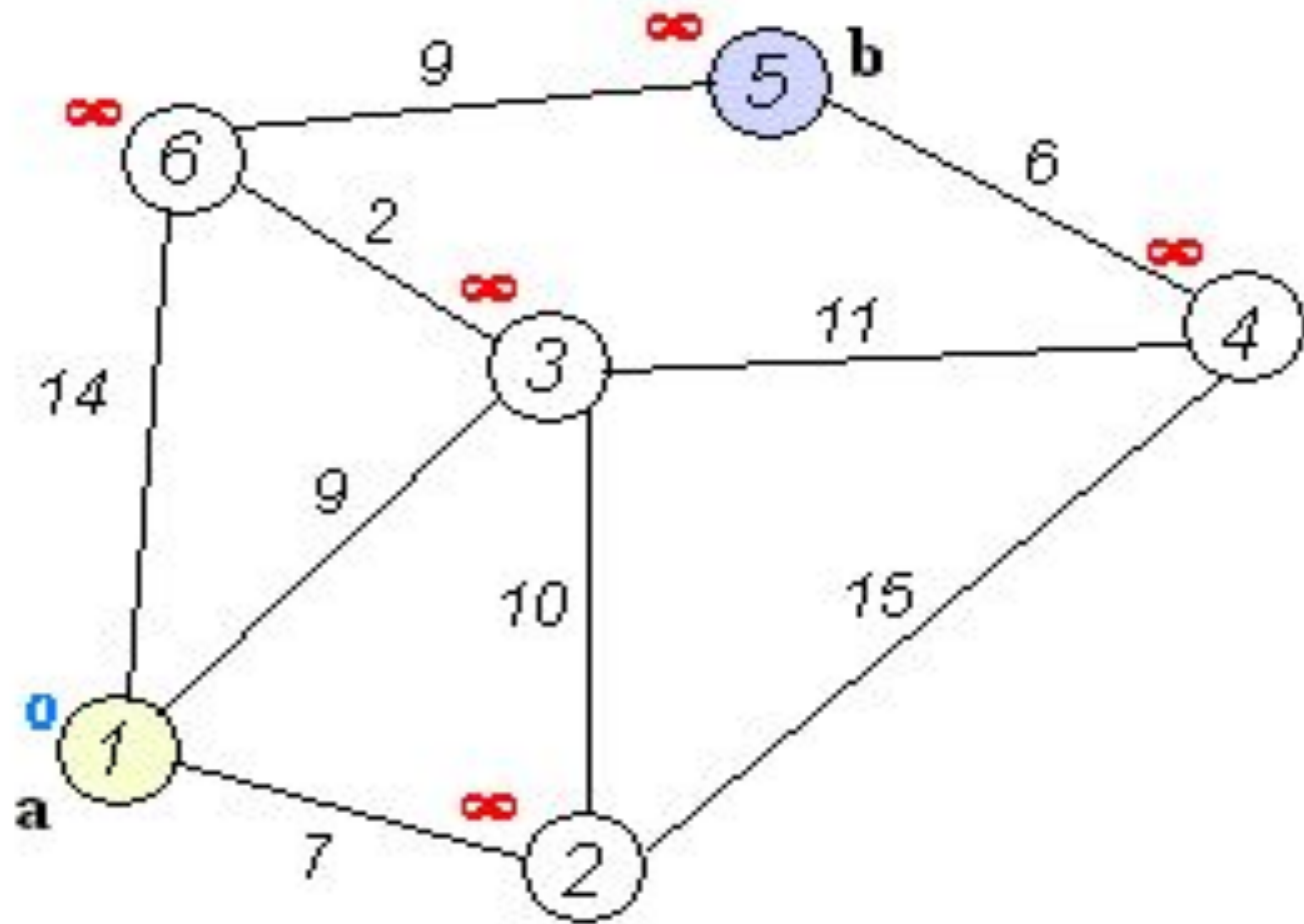
Дальше по аналогии: алгоритм выбирает непосещённые вершины с наименьшей оценкой и считает расстояния от соседних с ней вершин до A. И продолжается это до тех пор, пока алгоритм не вычислит кратчайшие расстояния до A для всех вершин.

Финал:



Готово! Теперь мы можем построить кратчайший маршрут от A до любой другой вершины. Например:

- От A до F: A — E — F
- От A до D: A — E — D
- От A до C: A — B — C



Алгоритм:

1. Подготовка:

- Отметьте все вершины как "не посещенные".
- Установите расстояние до начальной вершины как 0, а до всех остальных как бесконечность.
- Создайте список "не посещенных" вершин.

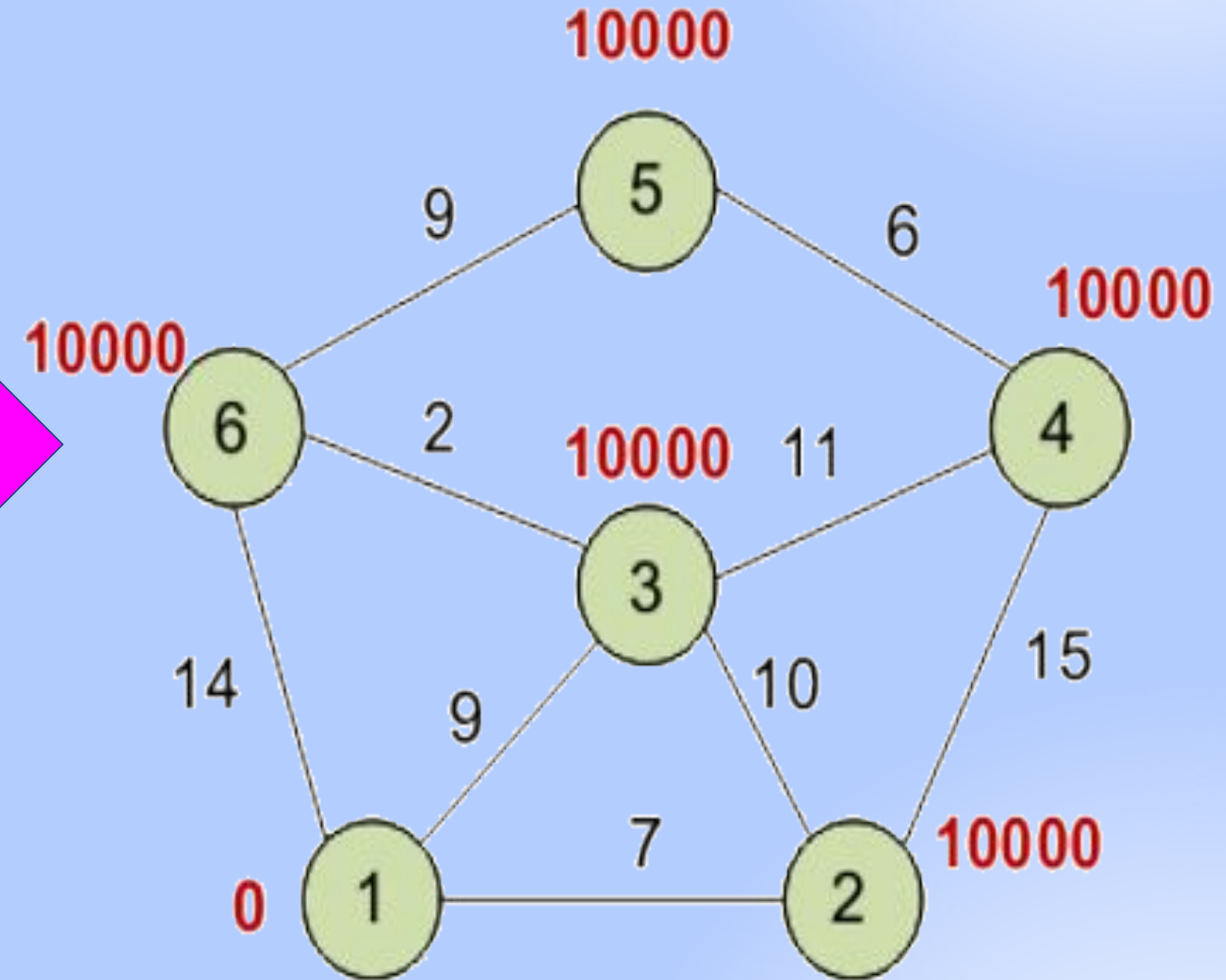
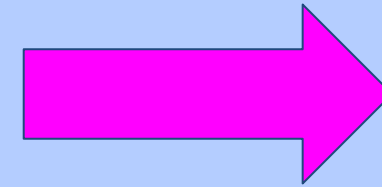
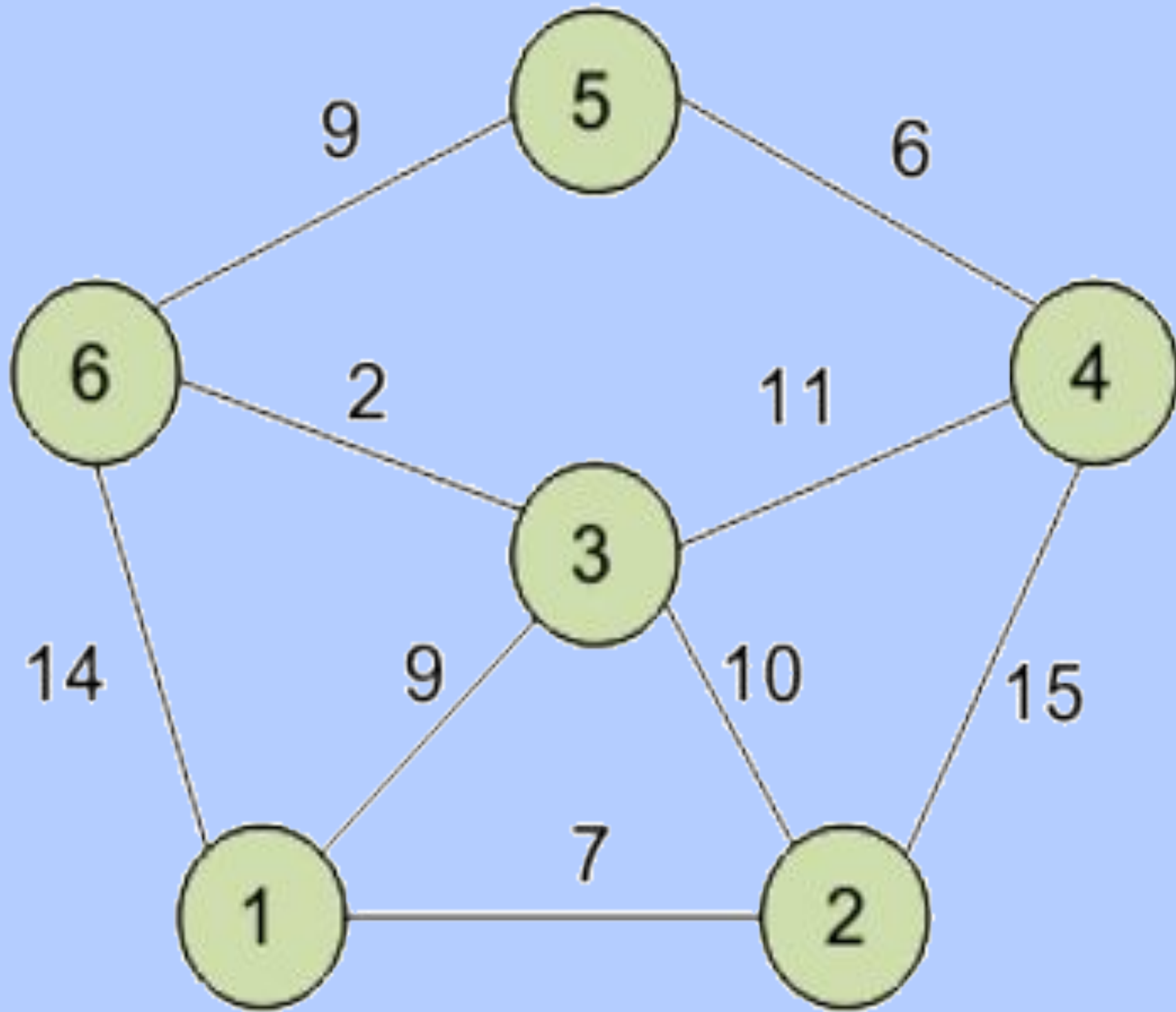
2. Основной шаг (повторяйте, пока список не пуст):

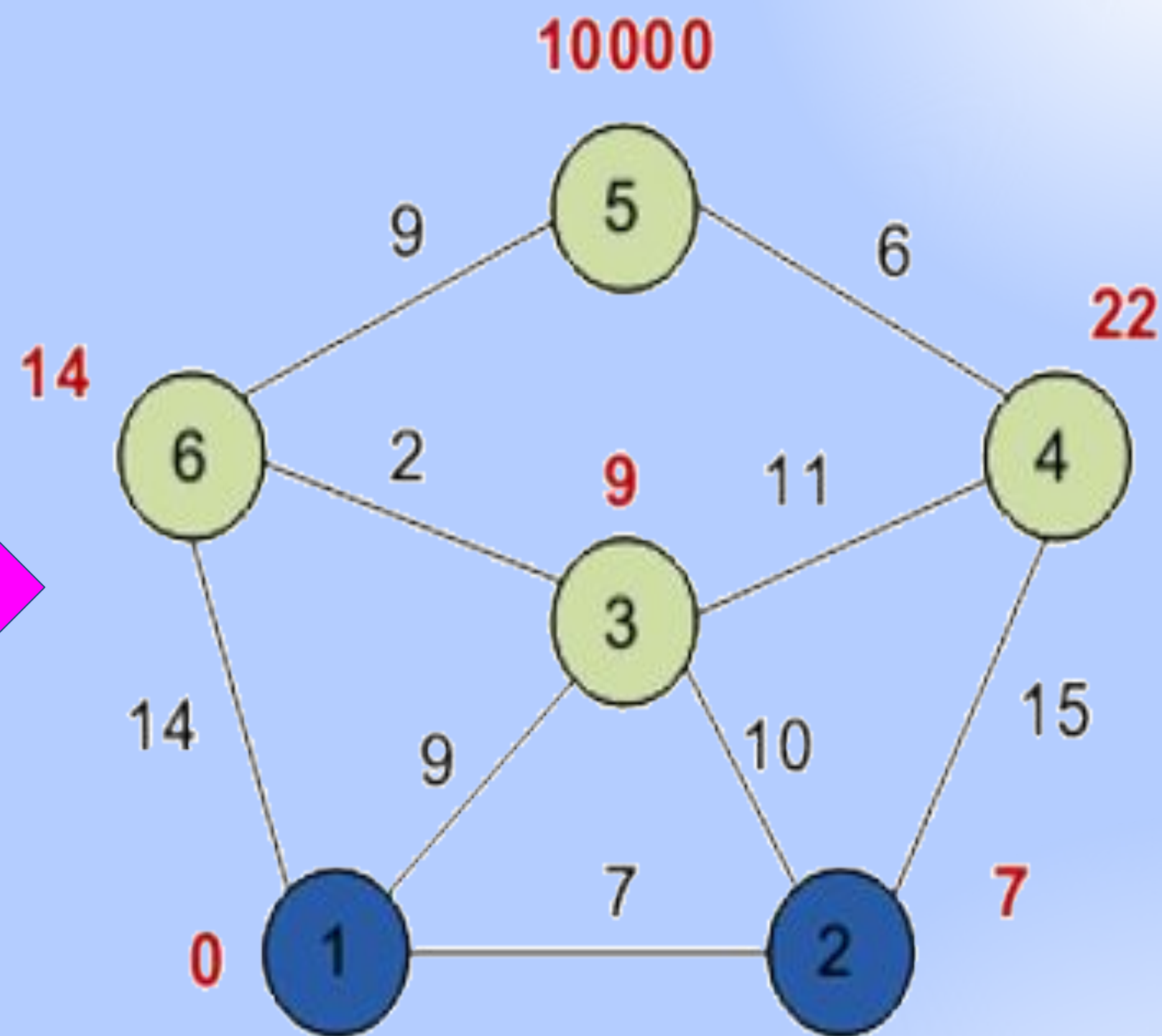
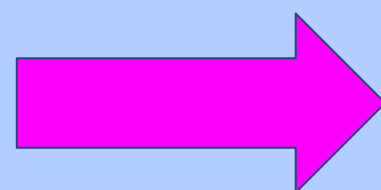
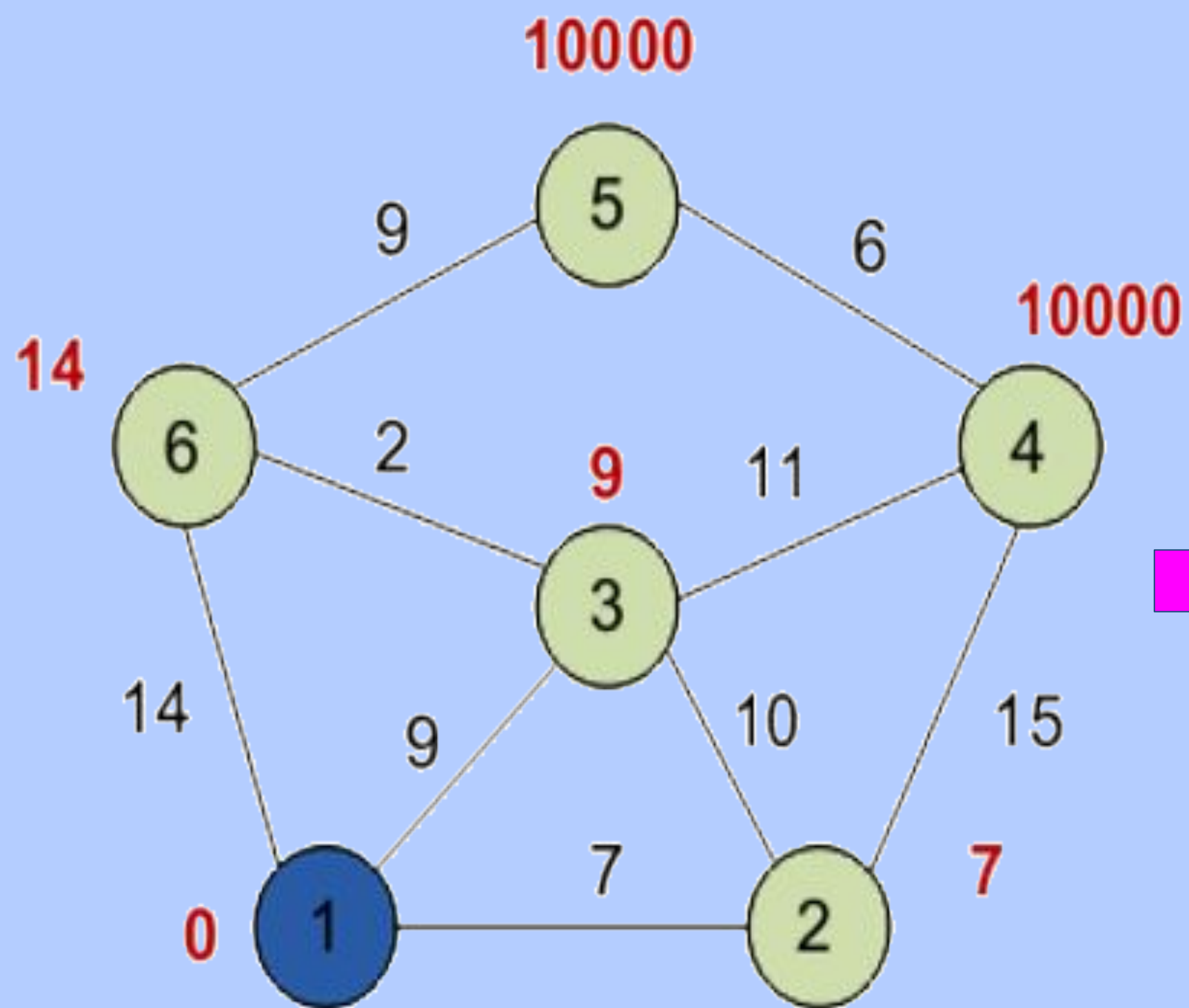
- Выберите из списка "не посещенных" вершин с наименьшим расстоянием.
- Отметьте его как "посещенный".
- Для каждого соседа выбранной вершины :
 - i. Рассчитайте новое расстояние до соседа через выбранную вершину (текущее расстояние до выбранной + вес ребра между ними).
 - ii. Если новое расстояние меньше текущего расстояния до соседа, обновите его.

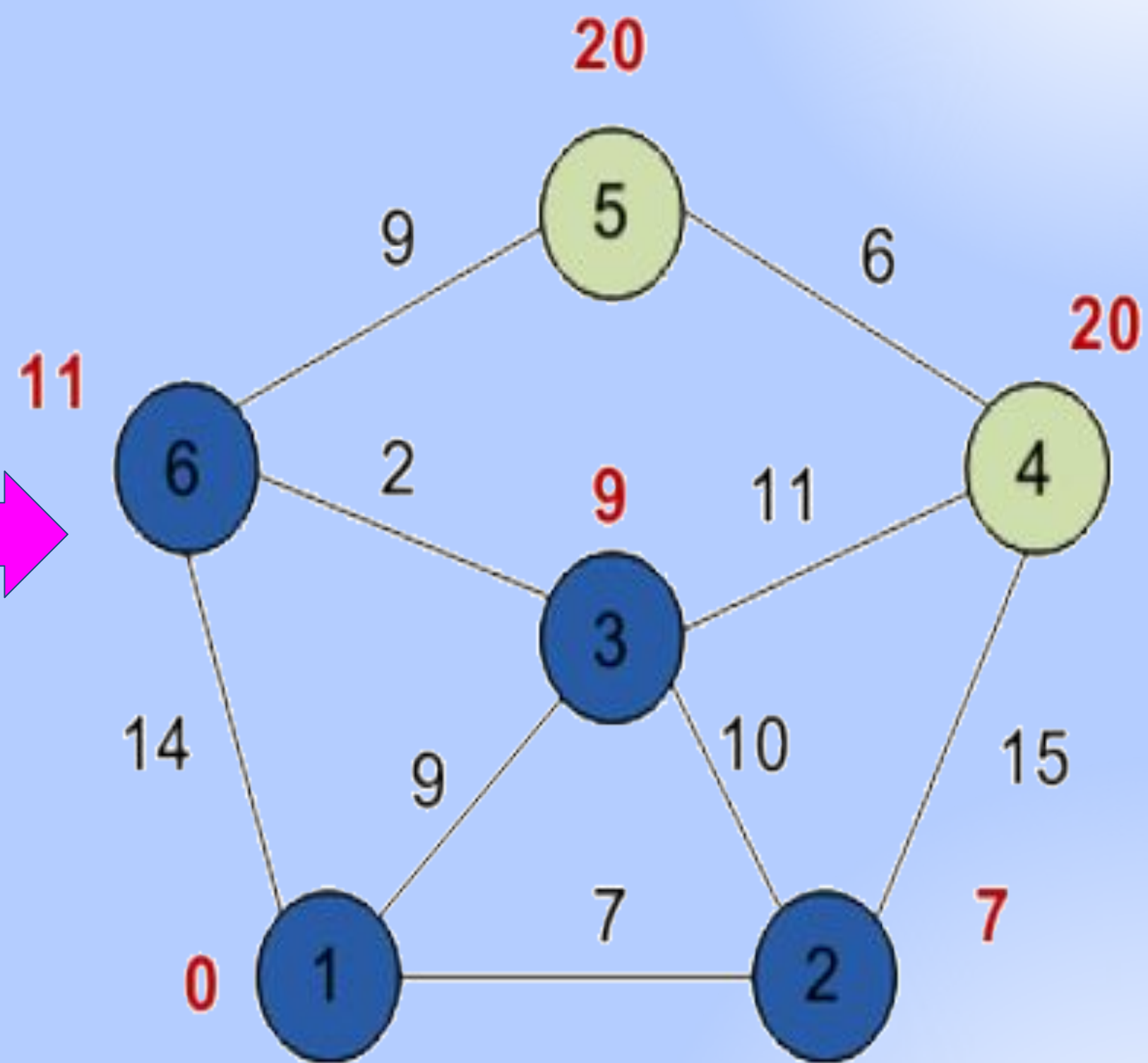
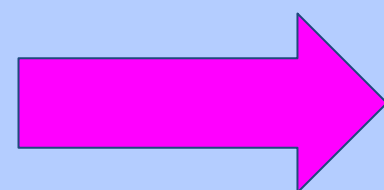
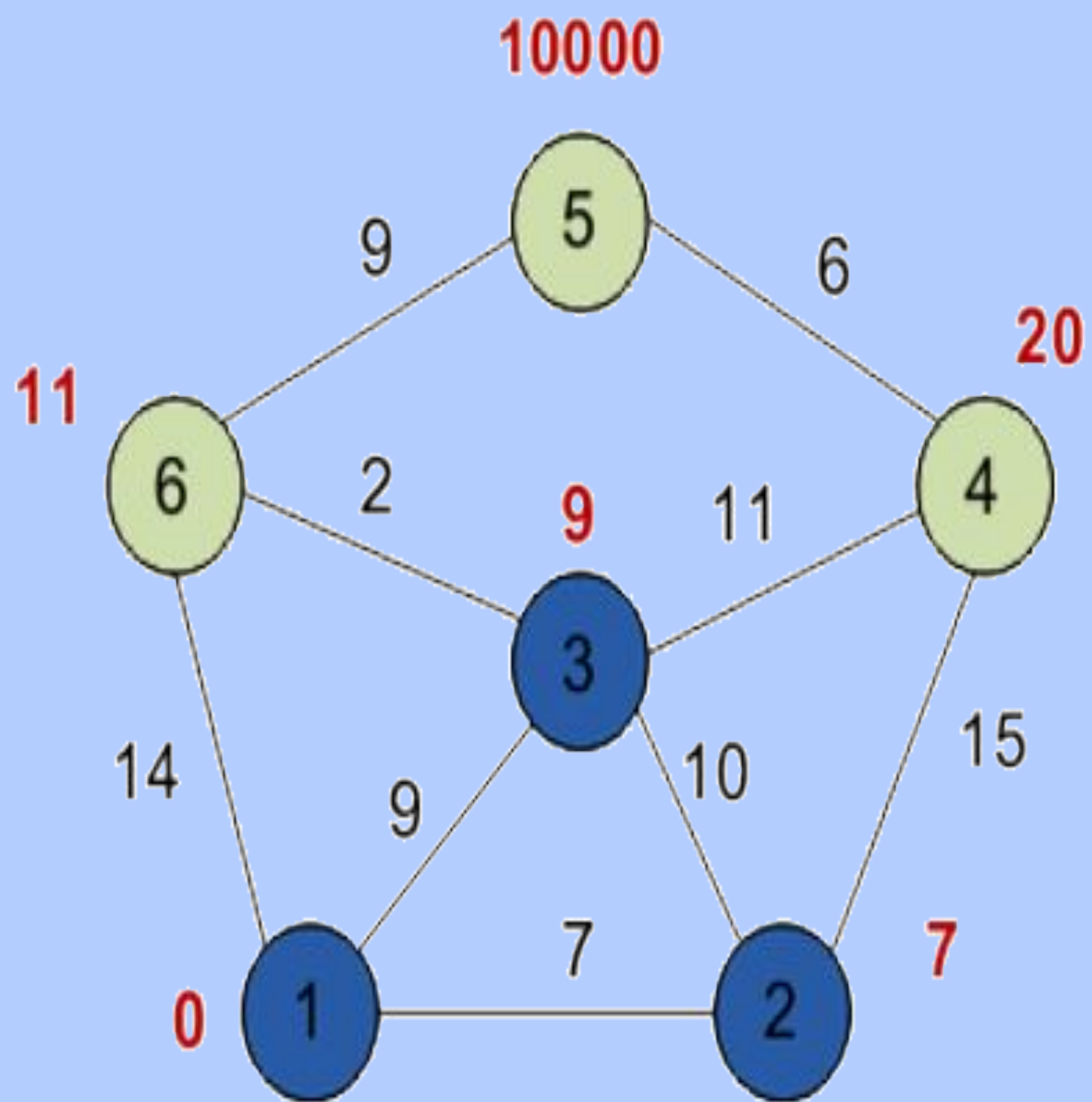
3. Результат: у вас есть кратчайшие расстояния от начальной вершины до всех остальных.

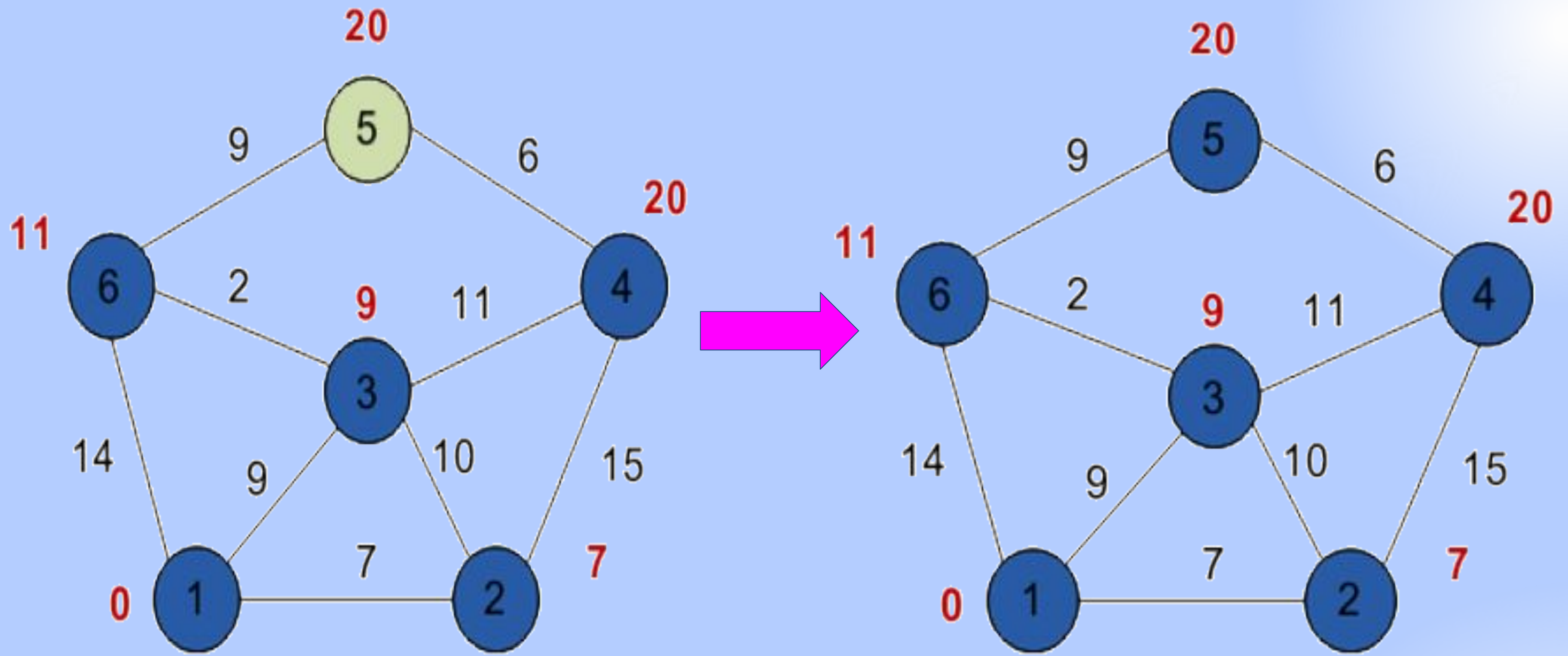
Пример №1:

Дан граф. Нужно найти минимальное расстояние от вершины 1 до вершины 5.







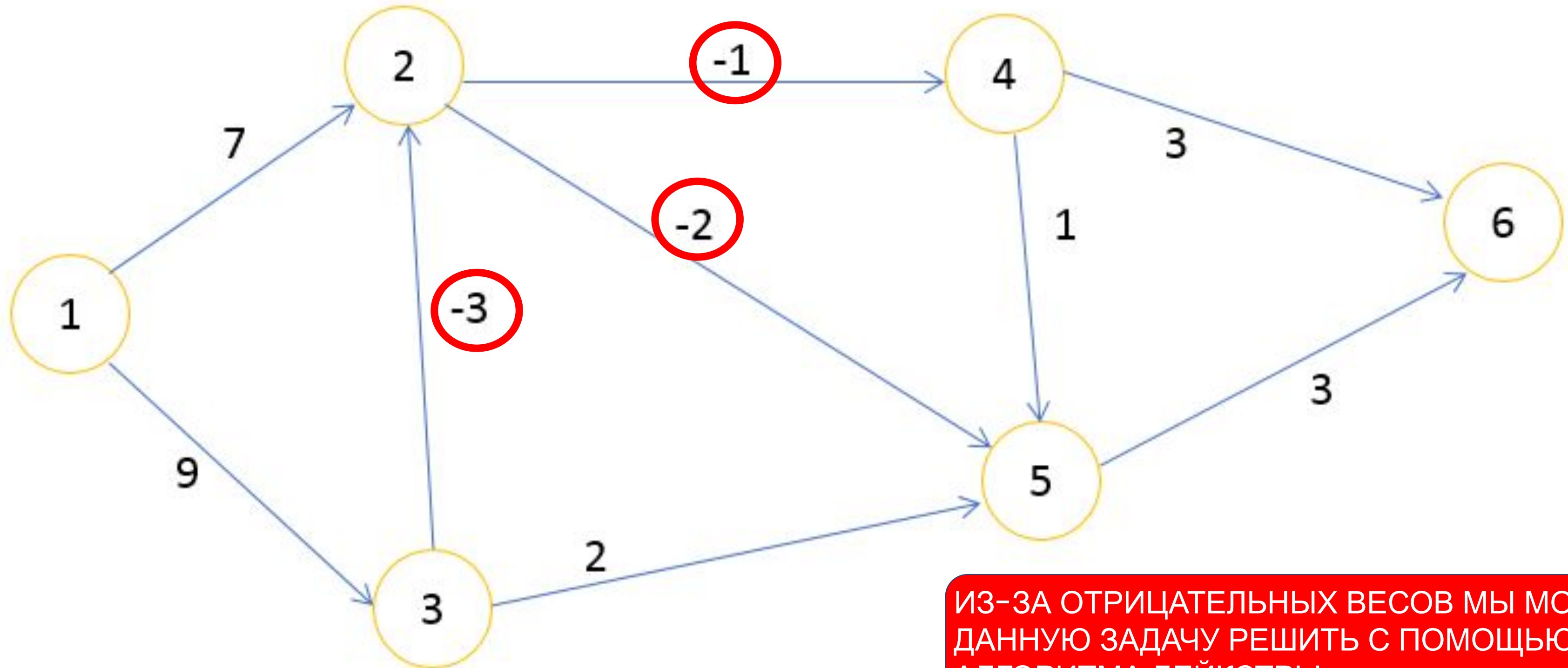


Таким образом, кратчайшим путем из вершины **1** в вершину **5** будет путь через вершины **1** – **3** – **6** – **5**, поскольку таким путем мы набираем минимальный вес, равный 20.

	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

Пример №2:

Дан граф. Нужно найти минимальное расстояние от вершины 1 до вершины 4 с помощью алгоритма Дейкстры.



ИЗ-ЗА ОТРИЦАТЕЛЬНЫХ ВЕСОВ МЫ МОЖЕМ ДАННУЮ ЗАДАЧУ РЕШИТЬ С ПОМОЩЬЮ АЛГОРИТМА ДЕЙКСТРЫ

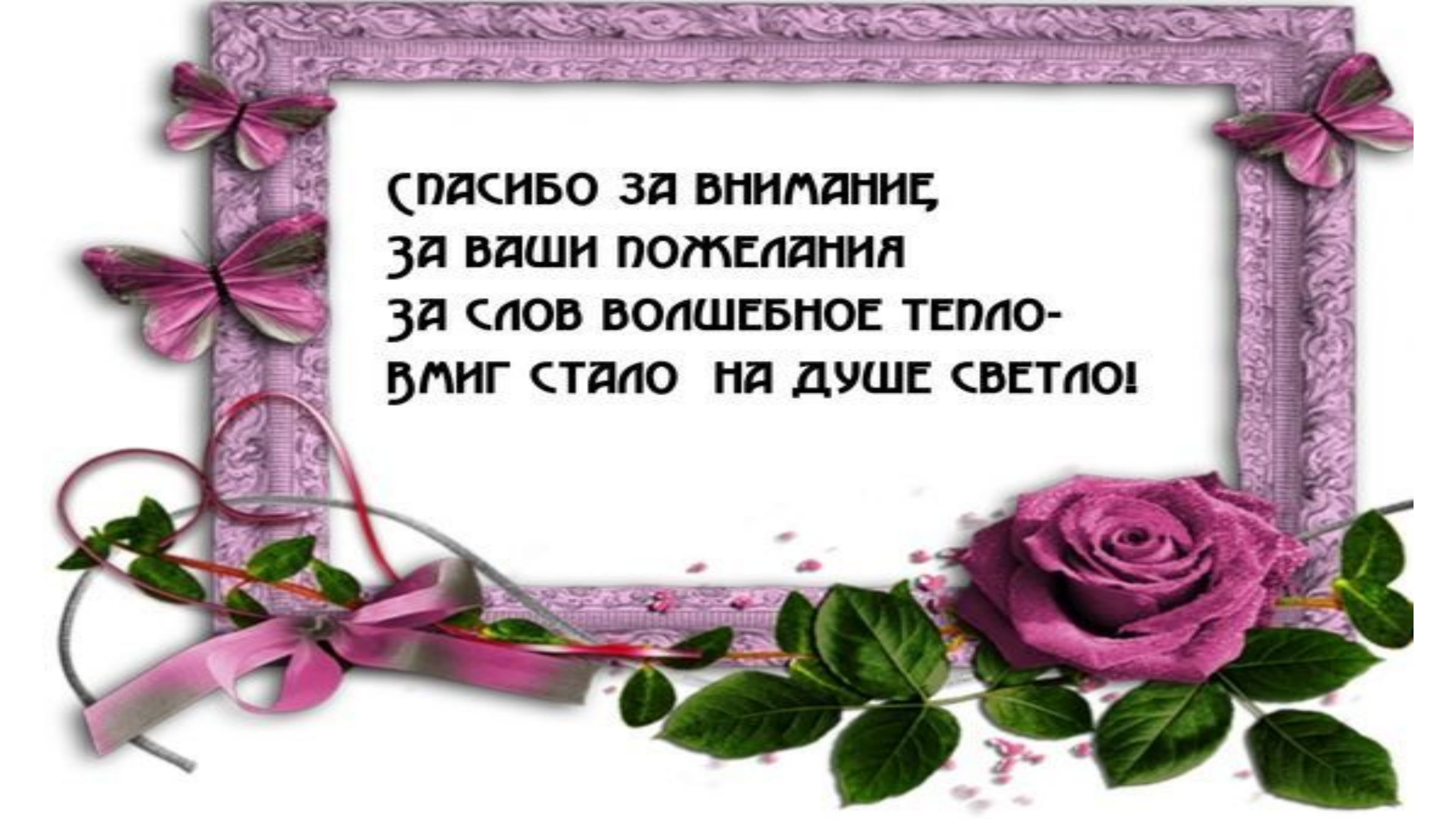
Выводы:

Алгоритм Дейкстры используется для нахождения кратчайшего пути от начальной вершины графа ко всем остальным.

Алгоритм Дейкстры рассматривает все вершины равнозначно и всегда выбирает вершину с наименьшим расстоянием до неё.

Алгоритм Дейкстры может быть медленным для больших графов, так как исследует все вершины.

Нужно всегда учитывать, что алгоритм работает только на взвешенных графах с положительными весами

A decorative purple frame with intricate patterns. Three pink butterflies are positioned along the left side of the frame. In the bottom right corner, there is a large, detailed pink rose with green leaves and a pink ribbon tied in a bow. Small pink petals are scattered around the base of the rose.

**СПАСИБО ЗА ВНИМАНИЕ,
ЗА ВАШИ ПОЖЕЛАНИЯ
ЗА СЛОВ ВОЛШЕБНОЕ ТЕПЛО-
ВМИГ СТАЛО НА ДУШЕ СВЕТЛО!**