1. Что такое "Минимальное покрывающее дерево"?

Это остовное дерево этого графа, имеющее минимальный возможный вес, где под весом дерева понимается сумма весов входящих в него рёбер.

**Остовное дерево** графа — ациклический связный подграф данного связного неориентированного графа, в который входят все его вершины.

1. Алгоритм Прима?

Суть самого алгоритма Прима сводится к жадному перебору рёбер, но уже из определенного множества. На входе так же имеется пустой подграф, который и будем достраивать до потенциального минимального остовного дерева.

* Изначально наш подграф состоит из одной любой вершины исходного графа.
* Затем из рёбер инцидентных этой вершине, выбирается такое минимальное ребро, которое связала бы две абсолютно разные компоненты связности, одной из которых и является наш подграф. То есть, как только у нас появляется возможность добавить новую вершину в наш подграф, мы тут же включаем ее по минимально возможному весу.

1. Алгоритм Крускала?

Механизм, по которому работает данный алгоритм, очень прост. На входе имеется пустой подграф, который и будем достраивать до потенциального минимального остовного дерева. Будем рассматривать только связные графы, в другом случае при применении алгоритма Крускала мы будем получать не минимальное остовное дерево, а просто остовной лес.

* Вначале мы производим сортировку рёбер **по не убыванию** по их весам.
* Добавляем i-ое ребро в наш подграф только в том случае, если данное **ребро соединяет две разные компоненты связности**, одним из которых является наш подграф. То есть, на каждом шаге добавляется минимальное по весу ребро, один конец которого содержится в нашем подграфе, а другой - еще нет.
* Алгоритм завершит свою работу после того, как множество вершин нашего подграфа совпадет с множеством вершин исходного графа.

1. Нахождение кратчайших путей в графе?

Задача поиска самого короткого пути между двумя точками на графе, в которой минимизируется сумма весов рёбер, составляющих путь.

1. Алгоритм Форда-Беллмана?

Находит расстояние от одной вершины (дадим ей номер 0) до всех остальных за количество операций порядка **n \* m**. Аналогично предыдущему алгоритму, веса могут быть отрицательными, но у нас не может быть циклов с отрицательной суммой весов рёбер.

Заведём массив d[0… n — 1], в котором на i-ой итерации будем хранить ответ на исходную задачу с ограничением на то, что в путь должно входить строго меньше i рёбер. Если таких путей до вершины j нет, то d[j] = 2000000000 (это должна быть какая-то недостижимая константа, «бесконечность»). В самом начале d заполнен 2000000000. Чтобы обновлять на i-ой итерации массив, надо просто пройти по каждому ребру и попробовать улучшить расстояние до вершин, которые оно соединяет. Кратчайшие пути не содержат циклов, так как все циклы неотрицательны, и мы можем убрать цикл из путя, при этом длина пути не ухудшится (хочется также отметить, что именно так можно найти отрицательные циклы в графе: надо сделать ещё одну итерацию и посмотреть, не улучшилось ли расстояние до какой-нибудь вершины). Поэтому длина кратчайшего пути не больше n — 1, значит, после n-ой итерации d будет ответом на задачу.  
n итераций по m итераций, итого порядка n \* m операций.

1. Алгоритм Дейкстры?

Находит расстояние от одной вершины (дадим ей номер 0) до всех остальных за количество операций порядка **n^2**. Все веса неотрицательны.

На каждой итерации какие-то вершины будут помечены, а какие-то нет. Заведём два массива: mark[0… n — 1] — True, если вершина помечена, False иначе, d[0… n — 1] — для каждой вершины будет храниться длина кратчайшего пути, проходящего *только* по помеченным вершинам в качестве «пересадочных». Также поддерживается инвариант того, что для помеченных вершин длина, указанная в d, и есть ответ. Сначала помечена только вершина 0, а g[i] равно x, если 0 и i соединяет ребро весом x, равно 2000000000, если их не соединяет ребро, и равно 0, если i = 0.  
На каждой итерации мы находим вершину, с наименьшим значением в d среди непомеченных, пусть это вершина v. Тогда значение d[v] является ответом для v.   
Теперь смело помечаем вершину v и пересчитываем d. Так делаем, пока все вершины не станут помеченными, и d не станет ответом на задачу.  
n итераций по n итераций (на поиск вершины v), итого порядка n^2 операций.

1. Нахождение кратчайших путей в бесконтурном графе?

Для бесконтурного графа с произвольными весами дуг задача нахождения кратчайшего пути может быть решена за *О*(*m*) шагов.

Бесконтурном графе -орграф, не содержащий контуров, но, возможно, имеющий циклы.

1. Поиск кратчайших путей между всеми парами вершин?

Нахождение длин кратчайших путей между всеми парами вершин во взвешенном ориентированном графе.

1. Алгоритм Флойда?

Алгоритм нахождения длин кратчайших путей между всеми парами вершин во взвешенном ориентированном графе. Работает корректно, если в графе нет циклов отрицательной величины, а в случае, когда такой цикл есть, позволяет найти хотя бы один такой цикл.

В массиве d[0… n — 1][0… n — 1] на i-ой итерации будем хранить ответ на исходную задачу с ограничением на то, что в качестве «пересадочных» в пути мы будем использовать вершины с номером строго меньше i — 1 (вершины нумеруем с нуля). Пусть идёт i-ая итерация, и мы хотим обновить массив до i + 1-ой. Для этого для каждой пары вершин просто попытаемся взять в качестве пересадочной i — 1-ую вершину, и если это улучшает ответ, то так и оставим. Всего сделаем n + 1 итерацию, после её завершения в качестве «пересадочных» мы сможем использовать любую, и массив d будет являться ответом.  
n итераций по n итераций по n итераций, итого порядка n^3 операций.