Теорема Форда-Фалкерсона. Теорема состоит из нескольких утверждений:

1. Максимальный поток равен минимальному разрезу
2. Поток максимален, если не существует пути $s \to t$ в остаточной сети

Давайте покажем первое утверждение. Для этого сначала скажем, что любой поток не больше любого разреза. Это верно, потому что размер разреза это сумма $\sum c_{v,u}$, а поток это $|f| = \sum f_{v,u}$ по $v \in A, u \notin A$. Теперь мы предъявим разрез, размер которого равен размеру какого-то потока — тогда мы одновременно найдем и минимальный разрез, и максимальный поток.

Пусть мы взяли какой-то существующий поток, в остаточной сети которого не было пути $s \to t$ (условие п.2). Тогда возьмем разрез, образованный насыщенными ребрами — то есть возьмем в множество A все вершины, достижимые из s в остаточной сети. Тогда для всех ребер разреза $c_{v,u} = f_{v,u}$, то есть вес разреза совпал с величиной потока. Значит, такой поток максимален, а соответствующий разрез минимальный.

Алгоритм Форда-Фалкерсона. Самый простой алгоритм для поиска потока — найти произвольный путь в остаточной сети и пустить поток вдоль него. Работает данный алгоритм за $O(|f| \cdot E)$, что не очень хорошо. Стоит заметить, что алгоритм сохраняет дробность решений — то есть поток на целых чисел будет целым числом, для четных целых — четным целым (потому что происходят только операции взятия минимума и вычитания).

Алгоритм Эдмондса-Карпа. Идея алгоритма в том, чтобы дополнять поток вдоль кратчайшего по ребрам пути из s в t. Такой алгоритм имеет уже полиномиальную сложность. По сути, мы дополняем поток вдоль пути, который находится поиском в ширину. Для доказательства времени работы изобразим слоистую сеть, которую найдет bfs (то есть сгруппированные по расстояниям вершины). У нас есть ребра внутри одной компоненты, ребра в следующую, и ребра на сколько угодно назад. Утверждается, что при насыщении вдоль пути расстояние от s до v не уменьшается. Это верно, потому что при обновлении сети у нас удаляются какие-то ребра вдоль пути, и добавляются обратные. Следовательно, при старом разбиении на классы у нас не появляется ребер на два и более классов вправо, а значит расстояния могут только увеличиться.

На каждой итерации алгоритма насыщается какое-то ребро. Надо заметить, что оно может насыщаться еще раз, но для этого сначала надо пустить поток по обратному ребру. Сколько раз это можно сделать? $O(V)$. Потому что мы насыщаем вдоль ребра слева направо (из d в $d+1$). Тогда, поскольку вершины двигаются только вправо, то не получится пустить по ребру $v \to u$ поток раньше, чем переместив v и u на слой вправо. А всего вершины могут быть удалены не более, чем на $O(V)$. Тогда мы делаем $O(VE)$ итераций, на каждой из которых вызываем bfs, получая оценку $O(VE^2)$.

Алгоритм Диница. Идея алгоритма Диница похожа на нашу предыдущую идею в декомпозиции потока. Если у нас уже есть зафиксированная слоистая сеть, то можно пускать поток вдоль пути, пока путь находится по ребрам между слоями. Тогда мы будем искать пути, пока слоистая сеть не потеряет связность, потом перестроим слоистую сеть, и так далее. Перестроек будет $O(V)$, потому что вершина t каждый раз двигается хотя бы на слой вправо. На каждой итерации нам надо найти все пути с насыщениями. Насыщений, как можно заметить, не более $O(E)$. Хочется много раз запустить dfs, который найдет все пути. При этом если каждый раз запускать dfs заново, то мы получим сложность $O(E^2)$. Но если запоминать указатель на первое нерассмотренное ребро, то сложность получится $O(VE)$. Почему? Потому что мы будем либо совершать неудачную попытку и двигать указатели к следующему

в списке элементу, либо найдем путь длины $O(V)$. Операций первого рода суммарно будет как раз $O(E)$ на все запуски.

Масштабирование. Давайте попытаемся улучшить асимптотику другой техникой — будем сначала искать большие потоки, а потом маленькие. А именно, давайте по очереди искать пути, по которым можно пустить хотя бы 2^k единиц потока. Тогда на каждой итерации у нас будет не более $2^k \cdot E$ единиц потока, при этом мы будем находить пути с потоком хотя бы 2^{k-1}, что означает линейное от числа ребер число насыщений. Значит, Форд-Фалкерсон с масштабированием работает за $O(E^2 \log C)$.