

**Алгоритм поиска min-cost max-flow.** В общем виде решение выглядит так — найти минимальный по весу путь, после чего вдоль него дополнять поток. Поскольку в сети бывают ребра отрицательного веса (и отрицательные циклы!), то искать кратчайший путь надо алгоритмом Форда-Беллмана. Тогда решить задачу можно за  $O(|F| \cdot VE)$ .

**Алгоритм Джонсона.** Во многих задачах сеть имеет более простой вид. Например, в задаче о назначениях в сети нет отрицательных циклов. Если отрицательных цикл нет, то можно воспользоваться идеей про потенциалы. Давайте посчитаем кратчайшие расстояния от  $s$  до всех остальных вершин. Тогда введем новый вес ребра, равный  $w_{v,u} - \rho_u + \rho_v$  (их можно посчитать алгоритмом Форда-Фалкерсона). Таким образом, каждый вес ребра теперь не отрицательный. Поскольку вес любого пути  $s \rightarrow t$  в новом графе — это вес в старом графе, взятый с какой-то константой, то кратчайший путь в новом графе — это кратчайший путь в старом. А такой кратчайший путь можно найти алгоритмом Дейкстры. После обновления вдоль пути у нас появятся обратные ребра, и потенциалы надо будет пересчитать. Как это делать? Заметим, что на кратчайшем пути цена всех ребер нулевая. А тогда цена обратных ребер тоже нулевая, и можно обновить кратчайшие расстояния через кратчайшие расстояния в новом графе. А именно, добавить к потенциалам кратчайшие расстояния в новом графе, которые мы тоже могли найти алгоритмом Дейкстры. В таком случае, сложность получается  $O(|F| \cdot (E + V \log V) + VE)$ , и в задаче о назначениях это дает асимптотику  $O(V^3)$ .

**Строковые алгоритмы.** Строкой мы будем называть конечную последовательность символов какого-то алфавита  $\Sigma$ . Если не сказано иного, то считать размер алфавита константой нельзя.

**Задача о подстановке шаблона.** Задача выглядит так: есть две строки  $s$  и  $t$ . Надо найти позиции, с которых в строке  $s$  можно прочесть строку  $t$ . При этом строка  $t$  состоит из обычных символов алфавита, а также два специальных символа Джокер (символ '?', который означает любой символ, а также астериск '\*', который означает любую строку, возможно пустую). Соответственно, можно ли заменить вопросики на буквы, а звездочки на строки, чтобы получившаяся строка  $\bar{t}$  входила в  $s$ . Решать такую задачу можно с помощью динамического программирования за  $O(|s| \cdot |t|)$ , где состояние — это состояние вида «сколько символов из каждой строки мы прочитали».

**Бор.** Структура данных, которая позволяет делать добавление-удаление-поиск строк за их размер. Она реализуется как дерево, где на каждом ребре написана буква, а строке соответствует путь из корня до вершины. Соответственно, в каждой вершине надо хранить переходы по всем ребрам. Тогда добавление-поиск-удаление делается просто спуском в дереве, с созданием вершин, если пока что переходов соответствующих не было. И надо еще хранить терминальные пометки, которые говорят о том, заканчивалась ли какая-то строка из множества в текущей вершине. Тривиальная реализация будет хранить  $O(L|\Sigma|)$  памяти. Можно хранить список всех детей в списке, и тогда  $O(L|\Sigma|)$  времени, но линейная память. Можно создать хеш-таблицу с переходами в вершине, тогда линейно по обоим параметрам. Если сжать бор (то есть сжать вершины степени 1), то тоже эффективнее будет.