Алгоритм Йена

1 Задача:

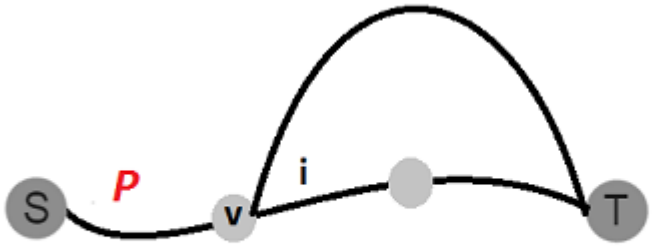
Дан взвешенный неориентированный граф G из n ребер. Найти в нем k-ый кратчайший путь из вершины S в T.
Def : **k-ый кратчайший путь** - k-ый по возрастанию простой путь из S в T.

Предполагается, что у нас есть алгоритм поиска кратчайших путей между двумя вершинами.

2 Алгоритм.

Разберем на примере алгоритма Дейкстры. Пусть вес каждого ребра графа $G \geqslant 0$. Допустим мы нашли кратчайший путь $L1$ из S в T. Теперь пытаемся найти второй кратчайший путь - $L2$. $L2$ должен отличаться хотя бы на одно ребро от $L1$. Тогда переберем все ребра $L1$ и попытаемся каждое из них запретить.

Предположим, что мы запретили i ребро. У нас образовался класс путей с одинаковым префиксом P и выделенным из них минимальным элементом(алгоритмом Дейкстры). Назовем эту операцию расщеплением.



Перебрав все ребра $L1$, мы получим множество классов с выделенным минимальными элементом в каждом. Выберем минимальный из этих элементов и получим $L2$. Заметим, что запрещая i-ое ребро мы выделяем сразу класс путей.

Теперь мы хотим найти $L3$. Он может содержится в классе $L2$, и тогда он должен отличаться от него(и мы расщепляем $L2$) или в каком-либо другом.

Для хранения классов необходимо использовать структуру с функциями getMin, getMax, например, множество. Каждый раз, при расщеплении минимального класса, будем добавлять в эту структуру новые образовавшиеся классы.

Шаг алгоритма выглядит следующим образом:

- взять очередной кратчайший путь
- расщепить его на несколько классов
- добавить образовавшиеся классы в структуру

3 Время работы и память.

Получившееся время работы: O(k * n * T(Время поиска расстояния между 2-мя вершинами)).
Из-за функций getMin и getMax общее время работы изменится на k * n * log(k). Это важно, так как log(k) в теории может оказаться больше чем Dij.

Память: O(k * n).
Чтобы сохранить один класс нам нужно O(n) памяти(Массив с запрещенными ребрами и префикс). Хранить больше чем k классов одновременно не имеет смысла.

Стоит добавить, что существует почти линейное решение за время O((k+n+m)*log(k+n+m)).

Алгоритм A*

Это алгоритм поиска кратчайших путей между 2 веришнами в графе, который часто встречается на практике. Например, нам нужно найти расстояние от одного города до другого. Отличие этого алгоритма от алгорима Дейкстры в том, что алгорим Дейкстры попытается пойти равномерно во все стороны (хотя, очевидно, это неоптимально), а алгоритм A* нет.

A - множество вершин, для которых расстояние посчитано
$d[i]$ - расстояние между начальной вершиной и i
$d[a, b]$ - расстояние между вершиной a и b

Введем понятия:
Черные - вершины, до которых минимальное расстояние уже посчитано.
Серые - вершины, до которых посчитано не обязятельно минимальное расстояние.
Белые - вершины, до которых расстояние еще не найдено.

В алгоритме используется некоторая функция оценки расстояния $f(x)$, которая выбирается самостоятельно(например, расстояние по прямой). Алгоритм Дейкстры на каждом шаге выбирает ту вершину для которой $d[v]$ минимален, мы же будем выбирать ту, для которой $d[v] + f(v)$ минимален.

Для корректной работы алгоритма нужно, чтобы для f выполнялось неравенство треугольника. То есть для любого ребра $[x, y]$ f должно удовлетворять уравнению:
$f(x) \leqslant d[x, y] + f(y)$

Доказательство корректности. Пусть есть вершина v для которой $d[v]$ посчитано. Доказываем, что когда мы его нашли, то оно посчитано корректно и никогда больше не изменится.

От противного. Пусть оно уменьшится, тогда в вершину v придет какой-то другой путь M.

Рассмотрим путь M. Он содержит как минимум одну белую вершину. Возьмем последний непрерывный отрезок пути P, состоящий из белых вершин. Так как начало пути - черная вершина, а конец - серая, рассматриваемый отрезок будет находиться среди серых вершин. Рассмотрим серую вершину u, предшествующую рассматриваемому отрезку.

$d[u] + f(u) \geqslant d[v] + f(v)$, так как v серая и $d[v] + f(v)$ минимально. Перейдем из вершины u в следующую за ним вершину $x \in M$.

По неравенству треугольника $f(x) + d[u, x] \geqslant f(u)$. Заменим $f(u)$ на что-то большее: $d[u] + d[u, x] + f(x) \geqslant d[v] + f(v)$. Для вершины x проделаем те же действия, что и для u, пока не дойдем до v. Получим $D + f(v) \geqslant d[v] + f(v)$, где $D = d[u] + d[u, x] + d[x, x1] + ... + d[k, v]$.

Все отрезки из белых вершин, стоящие до отрезка P, нам не нужны, так как вершина u уже была как-то посчитана, поэтому идти по этим отрезкам было бы невыгодно.