1 задача: Города

2 модуль, 2 семестр

ФИВТ МФТИ, 2019

Описание by Илья Белов

1. Текст задачи

Требуется отыскать самый выгодный маршрут между городами. Требуемое время работы $O((N+M)logN)$, где N-количество городов, M-известных дорог между ними.

Оптимизируйте ввод

Формат входных данных

Первая строка содержит число N – количество городов.

Вторая строка содержит число M - количество дорог.

Каждая следующая строка содержит описание дороги (откуда, куда, время в пути).

Последняя строка содержит маршрут (откуда и куда нужно доехать).

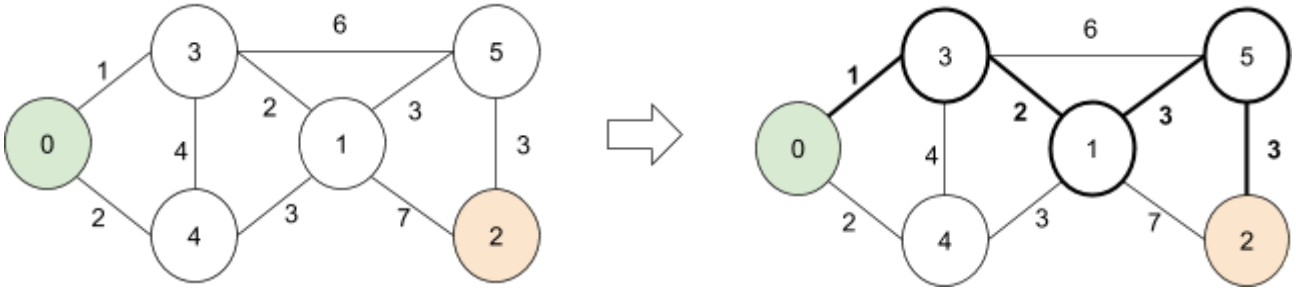
Формат выходных данных

Вывести длину самого выгодного маршрута

Пример:

in	out
6 9 0 3 1 0 4 2 1 2 7 1 3 2 1 4 3 1 5 3 2 5 3 3 4 4 3 5 6 0 2	9

Иллюстрация примера:



2. Описание алгоритма

Алгоритм решает задачу SSSP - он находит кратчайшие пути из одной вершины во все.

В алгоритме поддерживается множество вершин U, для которых уже вычислены длины кратчайших путей до них из s. На каждой итерации основного цикла выбирается вершина $u \in U$ которой на текущий момент соответствует минимальная оценка кратчайшего пути. Вершина u добавляется в множество U и производится релаксация всех исходящих из неё рёбер.

3. Доказательство корректности[1]

Докажем по индукции, что в момент посещения любой вершины u, $d(u) = \rho(s, u)$

На первом шаге выбирается s, для неё выполнено: $d(u) = \rho(s, s) = 0$

Пусть для первых шагов алгоритм сработал верно и на $n + 1$ шагу выбрана вершина u. Докажем, что в этот момент $d(u) = \rho(s, u)$. Для начала отметим, что для любой вершины v, всегда выполняется $d(v) \geq \rho(s, v)$ (алгоритм не может найти путь короче, чем кратчайший из всех существующих). Пусть P — кратчайший путь из s в u, v — первая непосещённая вершина на P, z — предшествующая ей (следовательно, посещённая). Поскольку путь P кратчайший, его часть, ведущая из s через z в v, тоже кратчайшая, следовательно $\rho(s, v) = \rho(s, z) + w(z, v)$. По предположению индукции, в момент посещения вершины z выполнялось $d(z) = \rho(s, z)$, следовательно, вершина v тогда получила метку не больше чем $d(z) + w(z, v) = \rho(s, z) + w(z, v) = \rho(s, v)$, следовательно, $d(v) = \rho(s, v)$. С другой стороны, поскольку сейчас мы выбрали вершину u, её метка минимальна среди непосещённых, то есть $d(u) \leq d(v) = \rho(s, v) \leq \rho(s, u)$, где второе неравенство верно из-за ранее упомянутого определения вершины ***V*** в качестве первой непосещённой вершины на P, то есть вес пути до промежуточной вершины не превосходит веса пути до конечной вершины вследствие неотрицательности весовой функции. Комбинируя это с $d(u) \geq \rho(s, u)$, имеем $d(u) = \rho(s, u)$, что и требовалось доказать.

Поскольку алгоритм заканчивает работу, когда все вершины посещены, в этот момент $d(u) = \rho(s, u)$ для всех u

4. Время работы и дополнительная память

$T = O((V + E)logV)$
$M = O(V)$

5. Доказательство времени работы

Каждая вершина релаксируется не более E раз. После извлечения вершины из очереди она уже никогда не будет положена в неё обратно ⇒ произойдёт V извлечений из очереди. Итого произойдёт $V + E$ обращений к очереди, одно обращение занимает $O(logV)$ времени (при использовании двоичной кучи), значит $T = O((V + E)logV)$

[1] Копипаста с Викиконспектов