Задача A. Разрезание графа

Имя входного файла:	`cutting.in`
Имя выходного файла:	`cutting.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Дан неориентированный граф. Над ним в заданном порядке производят операции следующих двух типов:

- `cut` — разрезать граф, то есть удалить из него ребро;
- `ask` — проверить, лежат ли две вершины графа в одной компоненте связности.

Известно, что после выполнения всех операций типа `cut` рёбер в графе не осталось. Найдите результат выполнения каждой из операций типа `ask`.

Формат входных данных

Первая строка входного файла содержит три целых числа, разделённые пробелами — количество вершин графа n, количество рёбер m и количество операций k ($1 \leqslant n \leqslant 50\,000$, $0 \leqslant m \leqslant 100\,000$, $m \leqslant k \leqslant 150\,000$).

Следующие m строк задают рёбра графа; i-я из этих строк содержит два числа u_i и v_i ($1 \leqslant u_i, v_i \leqslant n$), разделённые пробелами — номера концов i-го ребра. Вершины нумеруются с единицы; граф не содержит петель и кратных рёбер.

Далее следуют k строк, описывающих операции. Операция типа `cut` задаётся строкой «`cut` u v» ($1 \leqslant u, v \leqslant n$), которая означает, что из графа удаляют ребро между вершинами u и v. Операция типа `ask` задаётся строкой «`ask` u v» ($1 \leqslant u, v \leqslant n$), которая означает, что необходимо узнать, лежат ли в данный момент вершины u и v в одной компоненте связности. Гарантируется, что каждое ребро графа встретится в операциях типа `cut` ровно один раз.

Формат выходных данных

Для каждой операции `ask` во входном файле выведите на отдельной строке слово «`YES`», если две указанные вершины лежат в одной компоненте связности, и «`NO`» в противном случае. Порядок ответов должен соответствовать порядку операций `ask` во входном файле.

Пример

cutting.in	cutting.out
3 3 7 1 2 2 3 3 1 ask 3 3 cut 1 2 ask 1 2 cut 1 3 ask 2 1 cut 2 3 ask 3 1	YES YES NO NO

Задача B. Расстояние между вершинами

Имя входного файла:	`distance.in`
Имя выходного файла:	`distance.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Коль Дейкстру́ писать без кучи,
То тайм-лимит ты получишь...
А в совсем крутой задаче
Юзай кучу Фибоначчи!

Спектакль преподавателей ЛКШ.июль–2007

Дан неориентированный взвешенный граф. Требуется найти вес минимального пути между двумя вершинами.

Формат входных данных

Первая строка входного файла содержит два натуральных числа n и m — количества вершин и рёбер графа соответственно ($1 \leqslant n \leqslant 100\,000$, $1 \leqslant m \leqslant 200\,000$). Вторая строка входного файла содержит натуральные числа s и t — номера вершин, длину пути между которыми требуется найти ($1 \leqslant s, t \leqslant n$, $s \neq t$).

Следующие m строк содержат описание рёбер по одному на строке. Ребро номер i описывается тремя натуральными числами b_i, e_i и w_i — номерами концов ребра и его вес соответственно ($1 \leqslant b_i, e_i \leqslant n$, $0 \leqslant w_i \leqslant 100$).

Формат выходных данных

Первая строка выходного файла должна содержать одно натуральное число — вес минимального пути между вершинами s и t, или -1, если такого пути нет.

Пример

distance.in	distance.out
4 4 1 3 1 2 1 2 3 2 3 4 5 4 1 4	3

Задача C. Кратчайшие пути

Имя входного файла:	`path.in`
Имя выходного файла:	`path.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Вам дан взвешенный ориентированный граф и вершина s в нём. Для каждой вершины графа u выведите длину кратчайшего пути от вершины s до вершины u.

Формат входных данных

Первая строка входного файла содержит три целых числа n, m, s — количество вершин и ребёр в графе и номер начальной вершины соответственно ($2 \leqslant n \leqslant 2\,000$, $1 \leqslant m \leqslant 5\,000$).

Следующие m строчек описывают рёбра графа. Каждое ребро задаётся тремя числами — начальной вершиной, конечной вершиной и весом ребра соответственно. Вес ребра — целое число, не превосходящее 10^{15} по абсолютной величине. В графе могут быть кратные рёбра и петли.

Формат выходных данных

Выведите n строчек — для каждой вершины u выведите длину кратчайшего пути из s в u. Если не существует пути между s и u, выведите «*». Если не существует кратчайшего пути между s и u, выведите «-».

Примеры

path.in	path.out
6 7 1 1 2 10 2 3 5 1 3 100 3 5 7 5 4 10 4 3 -18 6 1 -1	0 10 - - - *

Задача D. Pink Floyd

Имя входного файла:	`floyd.in`
Имя выходного файла:	`floyd.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	64 мегабайта

Группа Pink Floyd собирается отправиться в новый концертный тур по всему миру. По предыдущему опыту группа знает, что солист Роджер Уотерс постоянно нервничает при перелетах. На некоторых маршрутах он теряет вес от волнения, а на других — много ест и набирает вес.

Известно, что чем больше весит Роджер, тем лучше выступает группа, поэтому требуется спланировать перелеты так, чтобы вес Роджера на каждом концерте был максимально возможным.

Группа должна посещать города в том же порядке, в котором она дает концерты. При этом между концертами группа может посещать промежуточные города.

Формат входных данных

Первая строка входного файла содержит три натуральных числа n, m и k — количество городов в мире, количество рейсов и количество концертов, которые должна дать группа соответственно ($n \leqslant 100$, $m \leqslant 10\,000$, $2 \leqslant k \leqslant 10\,000$). Города пронумерованы числами от 1 до n.

Следующие m строк содержат описание рейсов, по одному на строке. Рейс номер i описывается тремя числами b_i, e_i и w_i — номер начального и конечного города рейса и предполагаемое изменение веса Роджера в миллиграммах ($1 \leqslant b_i, e_i \leqslant n$, $-100\,000 \leqslant w_i \leqslant 100\,000$).

Последняя строка содержит числа $a_1, a_2, ..., a_k$ — номера городов, в которых проводятся концерты ($a_i \neq a_{i+1}$). В начале концертного тура группа находится в городе a_1.

Гарантируется, что группа может дать все концерты.

Формат выходных данных

Первая строка выходного файла должна содержать число l — количество рейсов, которые должна сделать группа. Вторая строка должна содержать l чисел — номера используемых рейсов.

Если существует такая последовательность маршрутов между концертами, что Роджер будет набирать вес неограниченно, то первая строка выходного файла должна содержать строку “`infinitely kind`”.

Примеры

floyd.in	floyd.out
4 8 5 1 2 -2 2 3 3 3 4 -5 4 1 3 1 3 2 3 1 -2 3 2 -3 2 4 -10 1 3 1 2 4	6 5 6 5 7 2 3

Задача E. После финала

Имя входного файла:	`excursion.in`
Имя выходного файла:	`excursion.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Финал Чемпионата Байтландии по программированию впервые проводился в городе $\mathcal{I}$. Дорожная сеть города $\mathcal{I}$ представляет собой N перекрёстков, соединённых M дорогами с двусторонним движением. Программа финала была столь насыщенной, что участники не успели осмотреть город. Более того, церемония закрытия затянулась, так что при отъезде команды-победителя времени оставалось только на то, чтобы добраться до аэропорта кратчайшим по суммарной длине путём. При этом, чтобы хотя бы немного посмотреть город, из всех таких маршрутов выбрали тот, который включает в себя целиком наибольшее количество дорог.

По заданной карте города вычислите длину маршрута и количество дорог, которое удалось посмотреть команде.

Формат входных данных

Первая строка содержит два целых числа N и M — количество перекрёстков в городе $\mathcal{I}$ и количество дорог соответственно ($2 \leqslant N \leqslant 1000$, $1 \leqslant M \leqslant 2 \cdot 10^5$).

Каждая из следующих M строк содержит три целых числа a_i, b_i и c_i ($1 \leqslant a_i, b_i \leqslant N$, $a_i \neq b_i$, $1 \leqslant c_i \leqslant 1000$) — номера перекрёстков, соединяемых i-й дорогой, и длину этой дороги. Два различных перекрёстка могут быть соединены более чем одной дорогой. Место проведения финала находится на перекрёстке с номером 1, аэропорт — на перекрёстке с номером N.

Гарантируется, что существует хотя бы один путь от места проведения финала в аэропорт.

Формат выходных данных

Выведите два целых числа P и Q — длину кратчайшего пути от места проведения финала в аэропорт и максимальное число дорог, которые может включать в себя кратчайший путь.

Примеры

excursion.in	excursion.out
3 3 1 2 1 1 3 2 2 3 1	2 2