

Задача А. Расстояние между вершинами

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

Дан неориентированный взвешенный граф.

Найти вес минимального пути между двумя вершинами.

Формат входных данных

Первая строка входного файла содержит натуральные числа N , M , вторая строка числа S и F ($N \leq 5\,000$, $M \leq 100\,000$, $1 \leq S, F \leq N$, $S \neq F$) — количество вершин и ребер графа а также номера вершин, длину пути между которыми требуется найти. Следующие M строк по три натуральных числа b_i , e_i и w_i — номера концов i -ого ребра и его вес соответственно ($1 \leq b_i, e_i \leq n$, $0 \leq w_i \leq 100\,000$).

Формат выходных данных

Первая строка должна содержать одно натуральное число — вес минимального пути между вершинами S и F . Во второй строке через пробел выведите вершины на кратчайшем пути из S в F в порядке обхода. Если путь из S в F не существует, выведите -1 .

Примеры

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

Задача В. Расстояние между вершинами

Имя входного файла: `dist.in`
Имя выходного файла: `dist.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 ($1 \leq n \leq 100\,000$, $1 \leq m \leq 200\,000$). Вторая строка входного файла содержит натуральные числа s и t — номера вершин, длину пути между которыми требуется найти ($1 \leq s, t \leq n$, $s \neq t$).

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 ($1 \leq b_i, e_i \leq n$, $0 \leq w_i \leq 10000$).

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.

Примеры

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

Задача С. 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 ($n \leq 100$, $m \leq 10\,000$, $2 \leq k \leq 10\,000$). 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 ($1 \leq b_i, e_i \leq n$, $-100\,000 \leq w_i \leq 100\,000$).

Последняя строка содержит числа a_1, a_2, \dots, 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	6
1 2 -2	5 6 5 7 2 3
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	

Задача D. Налоги

Имя входного файла: `tax.in`
Имя выходного файла: `tax.out`
Ограничение по времени: 4 секунды
Ограничение по памяти: 256 мегабайт

Король Байтландии следует мировой тенденции и вводит налоги везде, где только может. Недавно он придумал *налог на путешествия*, который взимается с каждого, кто путешествует по стране.

Каждой Байтландской дороге сопоставлен размер налога за нее. При прохождении через город путешественнику необходимо заплатить налог, равный *максимуму* из размеров налога за дорогу, по которой он вошел в город, и дорогу, по которой он вышел из города. Также необходимо заплатить за первый и последний город: для них налог равен единственной соответствующей дороге.

Ваш друг Байтазар собирается в путешествие из Байтгорода в Байтополис. Помогите ему спланировать маршрут таким образом, чтобы размер уплаченного налога был минимален.

Формат входных данных

Первая строка ввода содержит два целых числа n и m ($2 \leq n \leq 100\,000, 1 \leq m \leq 200\,000$) — количество городов и количество дорог в Байтландии. Города пронумерованы от 1 до n .

Следующие m строк содержат описания дорог. i -ая из этих строк содержит три целых числа a_i, b_i, c_i ($1 \leq a_i, b_i \leq n, a_i \neq b_i, 1 \leq c_i \leq 1\,000\,000$). Это означает, что города a_i и b_i соединены двунаправленной дорогой с размером налога, равным c_i байтлей. Каждая пара соединена не более, чем одной дорогой.

Формат выходных данных

В первой и единственной строке вывода должно содержаться одно целое число — минимальный размер налога (в байтлях) на путешествие из Байтгорода (т.е. города номер 1) в Байтополис (т.е. город номер n). Гарантируется, что существует путь, соединяющий эти два города.

Примеры

tax.in	tax.out
4 5 1 2 5 1 3 2 2 3 1 2 4 4 3 4 8	12
6 6 6 2 4 4 2 3 3 5 3 2 1 8 4 1 2 4 6 6	13

Задача Е. Рейсы во времени

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

Между N населёнными пунктами совершаются пассажирские рейсы на машинах времени.

В момент времени 0 вы находитесь в пункте A . Вам дано расписание рейсов. Требуется оказаться в пункте B как можно раньше (то есть в наименьший возможный момент времени).

При этом разрешается делать пересадки с одного рейса на другой. Если вы прибываете в некоторый пункт в момент времени T , то вы можете уехать из него любым рейсом, который отправляется из этого пункта в момент времени T или позднее (но не раньше).

Формат входных данных

Первая строка входного файла содержит число N — количество населённых пунктов ($1 \leq N \leq 1000$). Вторая строка содержит два числа A и B — номера начального и конечного пунктов. Третья строка содержит число K — количество рейсов ($0 \leq K \leq 1000$). Следующие K строк содержат описания рейсов, по одному на строке. Каждое описание представляет собой четвёрку целых чисел. Первое число каждой четвёрки задаёт номер пункта отправления, второе — время отправления, третье — пункт назначения, четвёртое — время прибытия. Номера пунктов — натуральные числа из диапазона от 1 до N . Пункт назначения и пункт отправления могут совпадать. Время измеряется в некоторых абсолютных единицах и задаётся целым числом, по модулю не превышающим 10^9 . Поскольку рейсы совершаются на машинах времени, то время прибытия может быть как больше времени отправления, так и меньше, или равным ему.

Гарантируется, что входные данные таковы, что добраться из пункта A в пункт B всегда можно.

Формат выходных данных

Выведите в выходной файл минимальное время, когда вы сможете оказаться в пункте B .

Примеры

<code>time.in</code>	<code>time.out</code>
2 1 1 2 1 1 2 10 1 10 1 9	0
1 1 1 3 1 3 1 -5 1 -5 1 -3 1 -4 1 -10	-10