Задача A. Pink Floyd

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

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`”.

Пример

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 B. Relocation

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

Фермер Джон переезжает. Он хочет найти лучшее место для новой фермы, так чтобы минимизировать расстояние, которое он будет покрывать каждый день.

Регион, в который ФД планирует переехать, имеет n городов ($1 \leqslant n \leqslant 10\,000$). Там имеется m двунаправленных дорог ($1 \leqslant m \leqslant 50\,000$), соединяющих определенные пары городов. Все города достижимы друг для друга через некоторую комбинацию дорог. Требуется выбрать город, в котором поселится ФД.

В k ($1 \leqslant k \leqslant 5$) из этих городов имеется рынки, которые ФД планирует посещать каждый день. А именно, каждый день ФД планирует выехать со своей новой фермы, посетить все k городов с рынками и вернутся домой. ФД может посещать рынки в произвольном порядке. Город для своей новой фермы он обязательно должен выбрать в одном Из $n - k$ городов, которые не имеют рынка (цена проживания там существенно ниже).

Пожалуйста, помогите ФД вычислить минимальное расстояние, которое он будет ежедневное проезжать, если он выберет наилучший город и наилучший маршрут.

Формат входных данных

Три разделенных пробелом целых числа, n, m, k.

Следующие k строк содержат целое число в диапазоне $1 \ldots n$ Указывающее город, содержащий i-й рынок. Все рынки находятся в различных городах.

Следующие m строк содержат 3 разделенных одиночными пробелами целых числа, i, j ($1 \leqslant i, j \leqslant n$), l ($1 \leqslant l \leqslant 1000$), указывающих наличие дороги длиной l между городами i и j.

Формат выходных данных

Минимальное расстояние, которое ФД будет проезжать каждый день, если он выберет город для проживания оптимально и спланирует маршрут движения оптимально.

Примеры

relocate.in	relocate.out
5 6 3 1 2 3 1 2 1 1 5 2 3 2 3 3 4 5 4 2 7 4 5 10	12

Задача C. На санях

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

В начале XIX века еще не было самолетов, поездов и автомобилей, поэтому все междугородние зимние поездки совершались на санях. Как известно, с дорогами в России тогда было даже больше проблем, чем сейчас, а именно на n существовавших тогда городов имелась ровно $n-1$ дорога, каждая из которых соединяла ровно два города. К счастью, из каждого города можно было добраться в любой другой (возможно, через некоторые промежуточные города). В каждом городе имелась почтовая станция (или, как ее называют, «ям»), на которой можно было пересесть в другие сани. При этом ямщики могли долго запрягать (для каждого из городов известно время, которое ямщики в этом городе тратят на подготовку саней к поездке) и быстро ехать (также для каждого города известна скорость, с которой ездят ямщики из него). Можно считать, что количество ямщиков в каждом городе не ограничено.

Если бы олимпиада проводилась 200 лет назад, то путь участников занимал бы гораздо большее время, чем сейчас. Допустим, из каждого города в Москву выезжает участник олимпиады и хочет добраться до Москвы за наименьшее время (не обязательно по кратчайшему пути: он может заезжать в любые города, через один и тот же город можно проезжать несколько раз). Сначала он едет на ямщике своего города. Приехав в любой город, он может либо сразу ехать дальше, либо пересесть. В первом случае он едет с той же скоростью, с какой ехал раньше. Решив сменить ямщика, он сначала ждет, пока ямщик подготовит сани, и только потом едет с ним (естественно, с той скоростью, с которой ездит этот ямщик). В пути можно делать сколько угодно пересадок.

Жюри стало интересно, какое время необходимо, чтобы все участники олимпиады доехали из своего города в Москву 200 лет назад. Все участники выезжают из своих городов одновременно.

Формат входных данных

В первой строке входного файла дано натуральное число n, не превышающее 2000 — количество городов, соединенных дорогами. Город с номером 1 является столицей. Следующие n строк содержат по два целых числа: t_i и v_i — время подготовки саней в городе i, выраженное в часах, и скорость, с которой ездят ямщики из города i, в километрах в час ($0 \leqslant t_i \leqslant 100, 1 \leqslant v_i \leqslant 100$). Следующие $n-1$ строк содержат описания дорог того времени. Каждое описание состоит из трех чисел a_j, b_j и s_j, где a_j и b_j — номера соединенных городов, а s_j — расстояние между ними в километрах ($1 \leqslant a_j \leqslant n, 1 \leqslant b_j \leqslant n, a_j \neq b_j, 1 \leqslant s_j \leqslant 10000$). Все дороги двусторонние, то есть если из a можно проехать в b, то из b можно проехать в a. Гарантируется, что из всех городов можно добраться в столицу.

Формат выходных данных

Сначала выведите одно вещественное число — время в часах, в которое в Москву приедет последний участник. Далее выведите путь участника, который приедет самым последним (если таких участников несколько, выведите путь любого из них). Выведите город, из которого этот участник выехал первоначально, и перечислите в порядке посещения те города, в которых он делал пересадки. Последовательность должна заканчиваться столицей. При проверке ответ будет засчитан, если из трех величин «время путешествия по выведенному пути», «выведенное время» и «правильный ответ» каждые две отличаются менее чем на 0.0001.

Система оценки

Номер подзадачи	Баллы	Ограничения	Комментарии
		n	
0	0		Примеры из условия.
1	28	$1 \leqslant n \leqslant 40$	Баллы начисляются, если все тесты пройдены.
2	35	$1 \leqslant n \leqslant 400$	Баллы начисляются, если все тесты этой и предыдущих групп пройдены.
3	37	$1 \leqslant n \leqslant 2000$	Баллы начисляются, если все тесты этой и предыдущих групп пройдены.

Примеры

стандартный ввод	стандартный вывод
4 1 1 10 30 5 40 1 10 1 2 300 1 3 400 2 4 100	31.000000000000000 4 2 1
3 1 1 0 10 0 55 1 2 100 2 3 10	3.000000000000000 2 3 1

Задача D. Лабиринт знаний

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

Участникам сборов подарили билеты на аттракцион “Лабиринт знаний”. Лабиринт представляет собой N комнат, занумерованных от 1 до N, между некоторыми из которых есть двери. Когда человек проходит через дверь, показатель его знаний изменяется на определенную величину, фиксированную для данной двери. Вход в лабиринт находится в комнате 1, выход — в комнате N. Каждый участник сборов проходит лабиринт ровно один раз и наибрает некоторое количество знаний (при входе в лабиринт этот показатель равен нулю). Ваша задача — показать наилучший результат.

Формат входных данных

Первая строка входного файла содержит целые числа N ($1 \leqslant N \leqslant 2\,000$) — количество комнат и M ($1 \leqslant M \leqslant 10\,000$) — количество дверей. В каждой из следующих M строк содержится описание двери — номера комнат, из которой она ведет и в которую она ведет (через дверь в лабиринте можно ходить только в одну сторону), а также целое число, которое прибавляется к количеству знаний при прохождении через дверь (это число по модулю не превышает 10 000). Двери могут вести из комнаты в нее саму, между двумя комнатами может быть более одной двери.

Формат выходных данных

В выходной файл выведите “`:)`” — если можно пройти лабиринт и получить неограниченно большой запас знаний, “`:(`” — если лабиринт пройти нельзя, и максимальное количество набранных знаний в противном случае.

Примеры

maze.in	maze.out
2 2 1 2 5 1 2 -5	5

Задача E. Остовное дерево 2

Имя входного файла:	`spantree2.in`
Имя выходного файла:	`spantree2.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Требуется найти в связном графе остовное дерево минимального веса.

Формат входных данных

Первая строка входного файла содержит два натуральных числа n и m — количество вершин и ребер графа соответственно. Следующие m строк содержат описание ребер по одному на строке. Ребро номер i описывается тремя натуральными числами b_i, e_i и w_i — номера концов ребра и его вес соответственно ($1 \leqslant b_i, e_i \leqslant n$, $0 \leqslant w_i \leqslant 100\,000$). $n \leqslant 20\,000, m \leqslant 100\,000$.

Граф является связным.

Формат выходных данных

Первая строка выходного файла должна содержать одно натуральное число — вес минимального остовного дерева.

Пример

spantree2.in	spantree2.out
4 4 1 2 1 2 3 2 3 4 5 4 1 4	7

Задача F. Плотное остовное дерево

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

Требуется найти в графе остовное дерево, в котором разница между весом максимального и минимального ребра минимальна.

Формат входных данных

Первая строка входного файла содержит два натуральных числа n и m — количество вершин и ребер графа соответственно. Следующие m строк содержат описание ребер по одному на строке. Ребро номер i описывается тремя натуральными числами b_i, e_i и w_i — номера концов ребра и его вес соответственно ($1 \leqslant b_i, e_i \leqslant n$, $0 \leqslant |w_i| \leqslant 10^9$). $n \leqslant 1000, m \leqslant 10\,000$.

Формат выходных данных

Если остовное дерево существует, выведите в первой строке выходного файла `YES`, а во второй строке одно целое число — минимальную разность между весом максимального и минимального ребра в остовном дереве.

В противном случае в единственной строке выведите `NO`.

Пример

mindiff.in	mindiff.out
4 5 1 2 1 1 3 2 1 4 1 3 2 2 3 4 2	YES 0

Задача G. Транспортная сеть

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

Вовочка только что был назначен министром транспорта. В стране в настоящее время отсутствуют какие-либо транспортные средства между своими городами, так что Вовочке нужно все построить с нуля. Для этого он может строить дороги, чтобы соединить пары городов, а также может строить аэропорты в городах.

Построить дорогу длиной l стоит $R \times l$. Дорогу можно строить только от одного города до другого, по прямой. Построить аэропорт в городе стоит A рублей.

Вовочка хочет, чтобы из каждого города можно было попасть в каждый. Формально, в город Y можно добраться из города X, если выполняется одно какое-либо из следующих условий:

- Существует прямая дорога между X и Y.
- В X и Y есть аэропорты.
- Существует город Z, такой, что Z достижим из X и Y достижим из Z.

По данным координатам городов и константам R и A, найдите минимальную стоимость транспортной сети, которую министерство транспорта может построить.

Формат входных данных

Первая строка входного файла содержит число n — число городов ($1 \leqslant n \leqslant 150$). Вторая строка содержит n чисел — x-координаты городов. Третья строка содержит n чисел — y-координаты городов. Координаты от 0 до 10^6. Четвертая строка содержит вещественное число R, и пятая строка содержит вещественное число A.

Формат выходных данных

Выведите минимальную стоимость, которую можнно получить, с точностью до 6 знаков.

Примеры

transport.in	transport.out
4 0 0 400 400 0 100 0 100 1.0 150.0	500.0000000000
5 0 0 400 400 2000 0 100 0 100 2000 1.0 500.0	1600.0000000000
8 0 100 200 300 400 2000 2100 2200 0 100 200 300 400 2000 2100 2200 0.5 200.0	824.2640687119