Задача A. Кратчайший путь между вершинами

Имя входного файла:	`dist.in`
Имя выходного файла:	`dist.out`
Ограничение по времени:	4 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 минимальный путь 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 ($1 \leqslant b_i, e_i \leqslant n$, $0 \leqslant w_i \leqslant 10000$).

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. Если путь есть, то вторая строка должна содержать одно целое неотрицательное число k — количество вершин в кратчайшем пути от s до t. В третьей строчке выведите k чисел - сам кратчайший путь. Если кратчайших путей несколько, выведите любой.

Примеры

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

Задача B. 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 C. Цикл отрицательного веса

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

Дан ориентированный граф. Определите, есть ли в нем цикл отрицательного веса, и если да, то выведите его.

Формат входных данных

Во входном файле в первой строке число N ($1 \leqslant N \leqslant 100$) — количество вершин графа. В следующих N строках находится по N чисел — матрица смежности графа. Все веса ребер не превышают по модулю 10 000. Если ребра нет, то соответствующее число равно 100 000.

Формат выходных данных

В первой строке выходного файла выведите «`YES`», если цикл существует или «`NO`» в противном случае. При его наличии выведите во второй строке количество вершин в искомом цикле и в третьей строке — вершины входящие в этот цикл в порядке обхода.

Примеры

negcycle.in	negcycle.out
2 0 -1 -1 0	YES 2 2 1

Задача D. 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 E. Цивилизация

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

Карта мира в компьютерной игре «Цивилизация» версии 1 представляет собой прямоугольник, разбитый на квадратики. Каждый квадратик может иметь один из нескольких возможных рельефов, для простоты ограничимся тремя видами рельефов — поле, лес и вода. Поселенец перемещается по карте, при этом на перемещение в клетку, занятую полем, необходима одна единица времени, на перемещение в лес — две единицы времени, а перемещаться в клетку с водой нельзя.

У вас есть один поселенец, вы определили место, где нужно построить город, чтобы как можно скорее завладеть всем миром. Найдите маршрут переселенца, по которому можно прийти в место строительства города за минимальное время. На каждом ходе переселенец может перемещаться в клетку, имеющую общую сторону с той клеткой, где он сейчас находится.

Формат входных данных

Во входном файле записаны два натуральных числа N и M, не превосходящих 1000 — размеры карты мира (N — число строк в карте, M — число столбцов). Затем заданы координаты начального положения поселенца x и y, где x — номер строки, y — номер столбца на карте ($1 \leqslant x \leqslant N$, $1 \leqslant y \leqslant M$), строки нумеруются сверху вниз, столбцы — слева направо. Затем аналогично задаются координаты клетки, куда необходимо привести поселенца.

Далее идет описание карты мира в виде N строк, каждая из которых содержит M символов. Каждый символ может быть либо «`.`» (точка), обозначающим поле, либо «`W`», обозначающим лес, либо «`#`», обозначающим воду.

Гарантируется, что начальная и конечная клетки пути переселенца не являются водой.

Формат выходных данных

В первой строке выходного файла выведите количество единиц времени, необходимое для перемещения поселенца (перемещение в клетку с полем занимает 1 единицу времени, перемещение в клетку с лесом — 2 единицы времени). Во второй строке выходного файла выведите последовательность символов, задающих маршрут переселенца. Каждый символ должен быть одним из четырех следующих: «`N`» (движение вверх), «`E`» (движение вправо), «`S`» (движение вниз), «`W`» (движение влево). Если таких маршрутов несколько — выведите любой из них.

Если дойти из начальной клетки в конечную невозможно, выведите число -1.

Примеры

civ.in	civ.out
4 8 1 1 4 8 ....WWWW .######. .#..W... ...WWWW.	13 SSSEENEEEEES
4 7 2 2 3 6 ####### #WW#..# #WW#..# #######	-1

Задача F. Путешествие в Метрополис

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

Путешествие по стране никогда не бывает простым, особенно когда не существует прямого сообщения между городами. Группа туристов хочет добраться в город Метрополис, используя сеть железных дорог, которая соединяет n городов, пронумерованных от 1 до n. Город, из которого выезжает группа, имеет номер 1, Метрополис имеет номер n.

На железной дороге постоянно функционируют m маршрутов поездов. Каждый маршрут определяется последовательностью городов, перечисленных в том порядке, в каком их проезжает поезд, обслуживающий этот маршрут. В каждом маршруте для каждой пары соседних городов задано время, за которое поезд этого маршрута проезжает перегон между этими городами. При этом поезда разных маршрутов могут проезжать один и тот же перегон за разное время.

По пути в Метрополис группа может садиться на поезд и сходить с поезда в любом городе маршрута, не обязательно в начальном или конечном. При этом, можно сойти с поезда маршрута i, пересесть на поезд маршрута j, возможно сделать еще несколько пересадок, а потом вновь сесть в поезд того же маршрута i. Туристы предъявляют высокие требования к выбору способа проезда в Метрополис. Во-первых, суммарное время, проведенное в поездах, должно быть минимальным. Во-вторых, среди всех способов с минимальным временем нахождения в поездах предпочти- тельным является тот способ, для которого сумма квадратов промежутков времени, непрерывно проведенных в поезде между двумя пересадками, максимальна. Назовём эту сумму качеством пу- тешествия. Время, проведенное вне поездов, не учитывается.

Требуется написать программу, которая по описаниям имеющихся маршрутов поездов определит минимальное время, которое группе туристов придется провести в поездах, а также максимальное качество путешествия с таким временем.

Формат входных данных

В первой строке входных данных заданы два целых числа ($2 \leqslant n \leqslant 10^4, 1 \leqslant m \leqslant 2 \cdot 10^5$) — количество городов и количество маршрутов соответственно. Далее в m строках содержится описание маршрутов. Описание каждого маршрута начинается с целого числа s_i — количество перегонов в маршруте с номером $i (1 \leqslant s_i \leqslant 10^6)$. Далее следуют $(2s_i + 1)$ целых чисел, описывающих города маршрута и время проезда перегона между соседними городами маршрута, в следующем порядке: $v_{i,1}, t_{i,1}, v_{i,2}, t_{i,2}, v_{i,3}, ..., t_{i,s_i}, v_{i,s_i+1}$, где vi, j — номер j-го города маршрута, ti, j — время проезда перегона между j-м и $(j + 1)$-м городом $(1 \leqslant v_i, j \leqslant n, 1 \leqslant t_i, j \leqslant 1000)$. Гарантируется, что $s_1 + s_2 + ... + s_m \leqslant 10^6$. Никакие два города в описании маршрута не совпадают. Гарантируется, что с помощью имеющихся маршрутов можно добраться из города с номером 1 в город с номером n.

Формат выходных данных

Выходные данные должны содержать два целых числа — минимальное суммарное время, которое придется провести в поездах, и максимальное качество пути с таким временем

Примеры

trains.in	trains.out
2 1 1 1 3 2	3 9
5 2 4 1 3 2 3 3 5 5 10 4 3 4 2 2 1 3 4 1	9 35
5 2 3 1 1 2 2 3 3 4 3 2 2 3 3 4 4 5	10 82

Замечание

В первом примере группа туристов отправится прямым маршрутом в Метрополис. Во втором примере не оптимально проехать напрямую по первому маршруту, так как время в поезде при этом не будет минимальным возможным. Поэтому они отправятся на поезде по маршруту 1 из города 1 в город 2, затем на поезде по маршруту 2 из города 2 в город 3, а затем снова на поезде по маршруту 1 из города 3 в город 5. При этом сумма квадратов промежутков времени, проведенных в поездах между пересадками, равна $3 \cdot 2 + 1 \cdot 2 + 5 \cdot 2 = 35$. В третьем примере добраться из города 1 в город 4 за минимальное время можно, пересаживаясь с маршрута 1 на маршрут 2 в любом из городов 2, 3 или 4. Максимальное качество путешествия достигается при пересадке в городе 2: $1c \cdot 2 + 9 \cdot 2 = 82$.

Задача G. Пятница, тринадцатое

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

Девочка Маша очень суеверная, поэтому, когда наступает пятница, тринадцатое, она начинает вести себя очень неадекватно. К сожалению, ей нужно ходить в школу, и ей приходится ездить на автобусах. В городе есть N автобусных остановок и M автобусных маршрутов. Если в пятницу, тринадцатое она проедет от какой-то остановки до какой-то другой (возможно, используя несколько маршрутов) за время T, причём T делится на 13, то она начинает истошно орать и бегать по автобусу. Помогите ей добраться до школы и остаться в здравом уме.

Формат входных данных

В первой строке входного файла задано одно число T ($1 \leqslant T \leqslant 10$) — число тестов. В первой строке теста заданы два числа N и M ($1 \leqslant N \leqslant 50$, $1 \leqslant M \leqslant 2500$) — число остановок и маршрутов соответственно. Следующие M строк описывают маршруты в формате From To Time ($1 \leqslant$ From, To $\leqslant N$, $1 \leqslant$ Time $\leqslant 100$) — откуда и куда едет автобус и время в пути. На последней строчке теста будет указано, является ли сегодняшний день пятницей, тринадцатым (`True`) или нет (`False`).

Формат выходных данных

Для каждого теста выведите на отдельной строчке минимальное время, за которое Маша сможет доехать от дома (остановка 1) до школы (остановка N), или -1, если она этого сделать не сможет.

Примеры

стандартный ввод	стандартный вывод
3 5 5 1 2 1 1 3 2 2 4 1 3 4 3 4 5 11 True 2 1 1 2 26 True 3 3 1 1 7 1 2 26 2 3 16 False	16 -1 42