Задача A. Коллекционирование монет

Имя входного файла:	`coins.in`
Имя выходного файла:	`coins.out`
Ограничение по времени:	3 секунды
Ограничение по памяти:	64 мебибайта

В клубе коллекционеров монет состоят n человек. В настоящее время клубом проводится конкурс — кто первым соберет полную коллекцию монет `XIX` века. По требованиям клуба, эта коллекция должна состоять из m различных типов монет, причем, можно иметь больше одной монеты каждого типа, но обязательно хотя бы одну.

Вася решил подойти к этому вопросу основательно и хочет рассчитать оптимальную стратегию обмена монет. Он знает, что любой коллекционер с радостью обменяет любую свою монету a на любую Васину монету b, но только в одном случае: если у этого коллекционера больше одной монеты вида a, но еще нет ни одной монеты серии b. Ни на какие другие обмены коллекционеры не согласны. Сам Вася, в отличие от других, ставит глобальную оптимизацию выше локальной, и при желании может нарушать это правило.

Вася хочет с помощью таких обменов набрать как можно больше различных типов монет, чтобы облегчить себе в будущем задачу победы в конкурсе. Помогите ему рассчитать оптимальную стратегию обмена монетами!

Формат входного файла

В первой строке входного файла заданы два числа n и m ($1 \le m, n \le 100$). Далее следуют n строк по m чисел в каждой: j-е число i-й строки — это количество монет j-го типа у i-го коллекционера. Все эти числа неотрицательны и не больше 1 000. Вася имеет номер 1.

Формат выходного файла

В первой строке выходного файла выведите максимальное количество различных типов монет, которые можно собрать. Далее выведите любую из последовательностей обменов, приводящую к такому количеству различных типов. Каждый из обменов выводится в формате x_i u_i v_i, где x_i — номер коллекционера, с которым следует обменяться, u_i — номер монеты, которую Вася от него получает, а v_i — номер монеты, которую Вася ему отдает.

Пример

coins.in	coins.out
7 6 3 0 0 0 0 0 0 2 0 0 0 0 0 2 0 0 0 0 0 2 0 0 0 0 1 0 2 1 1 0 1 0 1 2 1 0 1 0 1 1 2 0	3 2 2 1 3 2 1 4 2 1 5 3 2 6 4 2 7 5 2

Задача B. Охлаждение реактора

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

Известная террористическая группа под руководством знаменитого террориста Бен Гадена решила построить атомный реактор для получения оружейного плутония. Вам, как компьютерному гению этой группы, поручили разработать систему охлаждения реактора.

Система охлаждения реактора представляет собой набор труб, соединяющих узлы. По трубам течет жидкость, причем для каждой трубы строго определено направление, в котором она должна по ней течь. Узлы системы охлаждения занумерованы от 1 до N. Система охлаждения должна быть спроектирована таким образом, чтобы для каждого узла за единицу времени количество жидкости, втекающей в узел, было равно количеству жидкости, вытекающей из узла. То есть если из i-го узла в j-ый течет f_{ij} единиц жидкости за единицу времени (если из i в j нет трубы, то положим $f_{ij} = 0$), то для каждого узла i должно выполняться

$$\sum_{j=1}^{N} f_{ij} = \sum_{j=1}^{N} f_{ji}$$

У каждой трубы имеется пропускная способность c_{ij}. Кроме того, для обеспечения достаточного охлаждения требуется, чтобы по трубе протекало не менее l_{ij} единиц жидкости за единицу времени. То есть для трубы, ведущей из i-го узла в j-ый должно выполняться $l_{ij} \le f_{ij} \le c_{ij}$.

Вам дано описание системы охлаждения, выясните, каким образом можно пустить жидкость по трубам, чтобы выполнялись все указанные условия.

Формат входного файла

Первая строка входного файла содержит числа N и M — количество узлов и труб ($1 \le N \le 200$). Следующие M строк содержат описание труб. Каждая строка содержит четыре целых числа i, j, l_{ij} и c_{ij}. Любые два узла соединены не более чем одной трубой, если есть труба из i в j, то нет трубы из j в i, никакой узел не соединен трубой сам с собой, $0 \le l_{ij} \le c_{ij} \le 10^5$.

Формат выходного файла

Если решение существует, выведите на первой строке выходного файла слово `YES`. Затем выведите M чисел — количество жидкости, которое должно течь по трубам, числа должны быть выведены в том порядке, в котором трубы заданы во входном файле. Если решения не существует, выведите `NO`.

Пример

cooling.in	cooling.out
4 6 1 2 1 2 2 3 1 2 3 4 1 2 4 1 1 2 1 3 1 2 4 2 1 2	NO
4 6 1 2 1 3 2 3 1 3 3 4 1 3 4 1 1 3 1 3 1 3 4 2 1 3	YES 1 2 3 2 1 1

Задача C. Испорченный паркет

Имя входного файла:	`floor.in`
Имя выходного файла:	`floor.out`
Ограничение по времени:	3 секунды
Ограничение по памяти:	64 мебибайта

Пол в некоторой комнате размером $M \times N$ замощен паркетом. При этом некоторые плитки паркета оказались испорчены. Петя решил сделать ремонт в этой комнате, заменив только испорченные клетки. Придя в магазин, он обнаружил, что паркетные плитки бывают двух типов — размера 1×2, которые стоят A рублей (немного подумав, Петя понял, что плитки 1×2 можно поворачивать на 90 градусов, получая тем самым плитки 2×1) и размера 1×1, которые стоят B рублей. Разрезать плитку размера 1×2 на две размера 1×1 Петя не может.

Определите, какая минимальная сумма денег нужна Пете, чтобы сделать ремонт.

Формат входного файла

Первая строка входного файла содержит 4 числа N, M, A, B ($1 \le N, M \le 300$, A, B — целые числа, по модулю не превосходящие 1000). Каждая из последующих N строк содержит по M символов: символ «.» (точка) обозначает неиспорченную плитку паркета, а символ «*» (звездочка) — испорченную. В конце строк могут идти незначащие пробелы. В конце файла могут быть пустые строки.

Формат выходного файла

В выходной файл выведите одно число — минимальную сумму денег, имея которую можно заменить испорченные паркетины (и только их).

Пример

floor.in	floor.out
2 3 3 2 .** .*.	5

Задача D. Мороженое

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

По дороге в школу Петя любит забегать в киоск и покупать себе мороженое. Однако при этом он часто опаздывает в школу. Неожиданно Петя понял — он просто ходит не по кратчайшему пути!

Помогите Пете победить опоздания. Город можно представить как n перекрестков, соединенных m улицами, про каждую улицу известна ее длина. Дом Пети находится на перекрестке a, школа — на перекрестке b, а киоск с мороженым — на перекрестке c. По пути в школу Петя никогда не проходит через один перекресток дважды.

Формат входного файла

Первая строка входного файла содержит числа n и m ($3 \le n \le 30\,000$, $0 \le m \le 50\,000$). Вторая строка содержит три различных числа — a, b и c. Следующие m строк содержат по три целых числа x_i, y_i и l_i — номера перекрестков, соединенных улицей и ее длину (длина — целое положительное число, которое не превышает 10^4).

Формат выходного файла

Если путь из дома Пети до школы, проходящий через перекресток с мороженым и не проходящий по одному перекрестку два раза, существует, выведите на первой строке выходного файла два целых числа k и l — количество улиц, которые проходит Петя в оптимальном пути и длину пути. На второй строке выведите номера перекрестков в том порядке, в котором их посещает Петя.

В противном случае выведите -1 на первой строке выходного файла.

Примеры

icecream.in	icecream.out
5 6 1 5 3 1 2 1 2 3 4 2 4 1 3 4 2 5 1 1 4 5 2	4 9 1 2 3 4 5

Задача E. Сочинитель стихов

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

Начинающий поэт Вася разработал конечный автомат для упрощения процесса сочинения стихов. Этот автомат имеет N состояний, при этом переход из состояния в состояние осуществляется по K возможным рифмам. Автомат имеет одно начальное состояние a и одно конечное состояние b. Результатом работы сочинителя стихов является некоторая последовательность рифм, принимаемая автоматом, на которую Вася накладывает готовые стихи.

Для того, чтобы стихи не были слишком однообразными, после каждого перехода Вася несколько изменяет автомат. А именно, как только происходит переход из состояния i в состояние j по рифме k, Вася стирает все переходы, исходящие из состояния i по рифме k, а также все переходы, ведущие в состояние j по рифме k.

Требуется определить максимальный набор стихов, которые Вася может сочинить, используя свой автомат таким образом. После того, как какой-то переход стерт, он уже больше никогда в истории жизни этого автомата не будет восстановлен, так как Вася не хочет повторяться.

Формат входного файла

Первая строка входного файла содержит четыре целых числа — количество состояний автомата N ($1 \le N \le 50$), количество рифм, которые использует Вася в своих стихах K ($1 \le K \le 50$), а также номера начального и конечного состояний a и b ($1 \le a \ne b \le N$). Вторая строка содержит одно целое число — количество переходов в автомате M ($1 \le M \le 1\,000$). Далее следуют M строк, каждая из которых описывает один переход в формате u_i v_i k_i, означающий, что из состояния u_i можно перейти в состояние v_i по рифме k_i.

Формат выходного файла

В первой строке выходного файла выведите максимальное количество стихов Z, которые Вася может сочинить с помощью своего автомата. Далее выведите Z строк, по одной для каждого стиха. Описание стиха выводите в формате s_1 k_1 s_2 k_2 $\ldots$ s_{l-1} k_{l-1} s_l, где $s_1 = a$, $s_l = b$, k_i — рифмы, по которым производится переход, а s_i — состояния в порядке прохода.

Пример

rhyme.in	rhyme.out
6 3 1 4 9 1 2 1 1 6 1 2 3 1 3 4 3 1 5 2 1 2 2 5 4 1 2 4 3 6 4 2	2 1 1 2 1 3 3 4 1 2 5 1 4

Задача F. Такси

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

Управлять службой такси — сосвем не простое дело. Помимо естественной необходимости централизованного управления машинами для того, чтобы обслуживать заказы по мере их поступления и как можно быстрее, нужно также планировать поездки для обслуживания тех клиентов, которые сделали заказы заранее.

В вашем распоряжении находится список заказов такси на следующий день. Вам необходимо минимимизировать число машин такси, необходимых чтобы выполнить все заказы.

Для простоты будем считать, что план города представляет собой квадратную решетку. Адрес в городе будем обозначать парой целых чисел: x-координатой и y-координатой. Время, необходимое для того, чтобы добраться из точки с адресом (a, b) в точку (c, d), равно $|a - c| + |b - d|$ минут. Машина такси может выполнить очередной заказ, либо если это первый ее заказ за день, либо она успевает приехать в начальную точку из предыдущей конечной хотя бы за минуту до указанного срока. Обратите внимание, что выполнение некоторых заказов может окончиться после полуночи.

Формат входного файла

В первой строке входного файла записано число заказов M ($0 < M < 500$). Последующие M строк описывают сами заказы, по одному в строке. Про каждый заказ указано время отправления в формате `hh:mm` (в интервале с `00:00` по `23:59`), координаты (a, b) точки отправления и координаты (c, d) точки назначения. Все координаты во входном файле неотрицательные и не превосходят 200. Заказы записаны упорядоченными по времени отправления.

Формат выходного файла

В выходной файл выведите единственное целое число — минимальное количество машин такси, необходимых для обслуживания всех заказов.

Пример

taxi.in	taxi.out
2 08:00 10 11 9 16 08:07 9 16 10 11	1
2 08:00 10 11 9 16 08:06 9 16 10 11	2

Задача G. Великая стена

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

У короля Людовика двое сыновей. Они ненавидят друг друга, и король боится, что после его смерти страна будет уничтожена страшными войнами. Поэтому Людовик решил разделить свою страну на две части, в каждой из которых будет властвовать один из его сыновей. Он посадил их на трон в города A и B, и хочет построить минимально возможное количество фрагментов стены таким образом, чтобы не существовало пути из города A в город B.

Страну, в которой властвует Людовик, можно упрощенно представить в виде прямоугольника $m \times n$. В некоторых клетках этого прямоугольника расположены горы, по остальным же можно свободно перемещаться. Кроме этого, ландшафт в некоторых клетках удобен для строительства стены, в остальных же строительство невозможно.

При поездках по стране можно перемещаться из клетки в соседнюю по стороне, только если ни одна из этих клеток не содержит горы или построенного фрагмента стены.

Формат входного файла

В первой строке входного файла содержатся числа m и n ($1 \le m, n \le 50$). Во второй строке заданы числа k и l, где $0 \le k, l, k + l \le mn - 2$, k — количество клеток, на которых расположены горы, а l — количество клеток, на которых можно строить стену. Естественно, что на горах строить стену нельзя. Следующие k строк содержат координаты клеток с горами x_i и y_i, а за ними следуют l строк, содержащие координаты клеток, на которых можно построить стену — x_j и y_j. Последние две строки содержат координаты городов A (x_A и y_A) и B (x_B и y_B) соответственно. Среди клеток, описанных в этих $k+l+2$ строках, нет двух совпадающих. Гарантируется, что $1 \le x_i, x_j, x_A, x_B \le m$ и $1 \le y_i, y_j, y_A, y_B \le n$.

Формат выходного файла

В первой строке выходного файла должно быть выведено минимальное количество фрагментов стены F, которые необходимо построить. В последующих F строках необходимо вывести один из возможных вариантов застройки.

Если невозможно произвести требуемую застройку, то необходимо вывести в выходной файл единственное число -1.

Пример

wall.in	wall.out
5 5 3 8 3 2 2 4 3 4 3 1 1 3 2 3 3 3 4 3 5 3 1 4 1 5 2 1 5 5	3 3 1 1 3 3 3