Задача A. Паросочетание

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Двудольным графом называется неориентированный граф (V, E), $E \subseteq V \times V$ такой, что его множество вершин V можно разбить на два множества A и B, для которых $\forall (e_1, e_2) \in E$ $e_1 \in A$, $e_2 \in B$ и $A \cup B = V$, $A \cap B = \varnothing$.

Паросочетанием в двудольном графе называется любой набор его несмежных рёбер, то есть такой набор $S \subseteq E$, что для любых двух рёбер $e_1 = (u_1, v_1)$, $e_2 = (u_2, v_2)$ из S $u_1 \neq u_2$ и $v_1 \neq v_2$.

Ваша задача — найти максимальное паросочетание в двудольном графе, то есть паросочетание с максимально возможным числом рёбер.

Формат входных данных

В первой строке записаны два целых числа n и m ($1 \leqslant n, m \leqslant 250$), где n — число вершин в множестве A, а m — число вершин в B.

Далее следуют n строк с описаниями рёбер — i-я вершина из A описана в $(i+1)$-й строке файла. Каждая из этих строк содержит номера вершин из B, соединённых с i-й вершиной A. Гарантируется, что в графе нет кратных ребер. Вершины в A и B нумеруются независимо (с единицы). Список завершается числом 0.

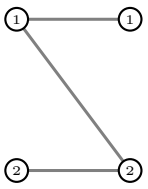
Формат выходных данных

Первая строка выходного файла должна содержать одно целое число l — количество рёбер в максимальном паросочетании. Далее следуют l строк, в каждой из которых должны быть два целых числа u_j и v_j — концы рёбер паросочетания в A и B соотвественно.

Пример

стандартный ввод	стандартный вывод
2 2 1 2 0 2 0	2 1 1 2 2

Замечание



Задача B. Разбиение на пары

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мегабайт

Космические археологи обнаружили на планете в соседней звездной системе n древних артефактов, которые они пронумеровали от 1 до n. Каждый артефакт имеет k различных параметров, каждый параметр характеризуется целым числом. Артефакт i имеет параметры $a_{i,1}, a_{i,2}, \ldots, a_{i,k}$. Оказалось, что первые параметры у всех артефактов различны: для всех $i \neq j$ выполнено $a_{i,1} \neq a_{j,1}$, при этом другие параметры у артефактов могут совпадать.

Учёные также обнаружили текст, в соответствии с которым для активации артефактов их необходимо особым образом разбить на пары и совместить. Разбиение артефактов на пары является корректным, если для каждого t от 1 до k можно выбрать такое число b_t, что оно лежит на отрезке между значениями t-го параметра артефактов каждой пары. То есть, если артефакты i и j образуют пару, должно выполняться условие $a_{i,t} \leqslant b_t \leqslant a_{j,t}$ или условие $a_{i,t} \geqslant b_t \geqslant a_{j,t}$.

Теперь ученые хотят выяснить, верно ли расшифрован текст. Для этого необходимо проверить, существует ли корректное разбиение артефактов на пары. Каждый артефакт должен войти ровно в одну пару в разбиении.

Требуется написать программу, которая по описанию параметров артефактов определяет, можно ли разбить их на пары таким образом, чтобы для каждого параметра существовало значение, лежащее между значениями этого параметра артефактов каждой пары, и в случае положительного ответа выводит такое разбиение.

Формат входных данных

В первой строке заданы целые числа n и k – количество артефактов и количество параметров ($2 \leqslant n \leqslant 2 \cdot 400$, n чётно, $1 \leqslant k \leqslant 7$).

В следующих n строках задано по k целых чисел $a_{i,1}, a_{i,2}, \ldots, a_{i,k}$ – параметры артефактов ($-10^9 \leqslant a_{i,j} \leqslant 10^9$, все значения $a_{i,1}$ различны).

Формат выходных данных

Выведите «NO», если требуемого разбиения на пары не существует.

В противном случае выведите «YES» в первой строке. Далее выведите $n/2$ строк, в каждой строке выведите по два числа – номера артефактов, из которых следует составить пару. Каждый артефакт должен быть выведен ровно один раз.

Если существует несколько корректных разбиений артефактов на пары, разрешается вывести любое из них

Примеры

стандартный ввод	стандартный вывод
6 2 8 6 1 5 6 3 3 1 4 7 7 2	YES 4 1 5 6 2 3
4 3 1 -1 -1 2 1 1 3 -1 1 4 1 -1	NO

Задача C. Покрытие путями

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

64 мегабайта

Задан ориентированный ациклический граф. Требуется определить минимальное количество не пересекающихся по вершинам путей, покрывающих все вершины.

Формат входных данных

Первая строка входного файла содержит целые числа n и m — количества вершин и рёбер графа соответственно ($2 \leqslant n \leqslant 1\,000$, $0 \leqslant m \leqslant 10^5$). В следующих m строках содержатся по два натуральных числа — номера вершин u и v, которые соединяет ребро (u, v).

Формат выходных данных

В первой строке выходного файла выведите натуральное число k — минимальное количество путей, необходимых для покрытия всех вершин.

Пример

paths.in	paths.out
3 3 1 3 3 2 1 2	1

Задача D. Замощение доминошками

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

Дано игровое поле размера $n \times m$, некоторые клетки которого уже замощены. Замостить свободные соседние клетки поля доминошкой размера 1×2 стоит a условных единиц, а замостить свободную клетку поля квадратиком размера 1×1 — b условных единиц.

Определите, какая минимальная сумма денег нужна, чтобы замостить всё поле.

Формат входных данных

Первая строка входного файла содержит 4 целых числа n, m, a, b ($1 \leqslant n, m \leqslant 100$, $|a| \leqslant 1\,000$, $|b| \leqslant 1\,000$). Каждая из последующих n строк содержит по m символов: символ '".“ (точка) обозначает занятую клетку поля, а символ '"*“ (звёздочка) — свободную.

Формат выходных данных

В выходной файл выведите одно число — минимальную сумму денег, имея которую можно замостить свободные клетки поля (и только их).

Пример

dominoes.in	dominoes.out
2 3 3 2 .** .*.	5

Задача E. Такси

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	0.5 секунд
Ограничение по памяти:	256 мегабайт

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 входных данных

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 выходных данных

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

Примеры

стандартный ввод	стандартный вывод
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 F. Фруктовый сок

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Степан недавно купил себе новую соковыжималку. Теперь по утрам он и его братья и сестры пьют свежевыжатый фруктовый сок. А это, между прочим, очень полезно!

Недавно они поняли, что можно пить сок, выжатый не только из одного вида фруктов, как, например, апельсиновый, но и различные смеси, например, виноградно-яблочный.

В семье Степана все очень любят сок, поэтому могут утром выпить не один стакан, причем разных видов сока. Например, его сестра Катя очень любит грейпфрутовый и апельсиновый соки. Степан, как наиболее технически грамотный человек, каждое утро занимается приготовлением соков.

Опишем подробнее, как работает соковыжималка. В нее загружаются фрукты, они проходят отжим в центрифуге, обезвоженная мякоть сбрасывается в отдельный резервуар, а сок попадает в специальную емкость.

Основная проблема состоит в том, что эту емкость иногда приходится мыть. Например, если после приготовления апельсинового сока, необходимо приготовить яблочный, то емкость надо мыть, иначе получится апельсиново-яблочный сок. Более формально, пусть сок A состоит из компонентов $a_1, \ldots, a_n$, а сок B – из компонентов $b_1, \ldots, b_m$. Сок B можно готовить после сока A, если любой из компонентов a_i является компонентом сока B (т.е. $\exists j : b_j = a_i$). В противном случае емкость для сока надо помыть.

Степан не очень любит мыть посуду, поэтому хочет мыть емкость как можно меньшее число раз. Помогите ему.

Формат входных данных

Первая строка входного файла содержит количество N различных соков, которые требуется приготовить ($1 \leqslant N \leqslant 300$). Каждая из последующих N строк описывает один из соков. Описание сока состоит из числа k его компонентов ($1 \leqslant k \leqslant 300$) и списка этих компонентов. Каждый из компонентов сока описывается словом длиной до 30 символов из строчных и прописных букв латинского алфавита. Прописные и строчные буквы различаются. Различные компоненты имеют различные названия.

Формат выходных данных

В выходной файл выведите минимальное количество раз, которое Степану придется помыть емкость для сока. Учитывайте при этом, что емкость для сока надо помыть и после приготовления последней порции сока.

Примеры

стандартный ввод	стандартный вывод
4 1 Apple 2 Apple Orange 1 Orange 2 Orange Pineapple	2
3 1 Apple 1 Orange 1 Mango	3

Задача G. Полезные ископаемые

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Ведется проект по освоению планеты соседней звездной системы. Для добычи полезных ископаемых планируется направить на планету несколько партий роботов.

Участок поверхности планеты, на котором планируется добывать полезные ископаемые, представляет собой клетчатый прямоугольник размером w на h, клетки участка имеют координаты от $(1, 1)$ до (w, h). В некоторых клетках участка находятся базы специалистов, в которые могут быть доставлены партии роботов. Всего на участке размещено s баз, и i-я база находится в клетке с координатами (x_i, y_i).

Каждая партия роботов характеризуется тремя параметрами: j-я партия доставляется на базу b_j, содержит n_j роботов и каждый робот партии обладает мобильностью m_j. Когда партия роботов доставляется на соответствующую базу, каждый робот этой партии перемещается по поверхности планеты от базы до некоторой клетки. Если мобильность робота равна m, он может не более m раз переместиться на одну из восьми соседних клеток, как показано на рис. 1.

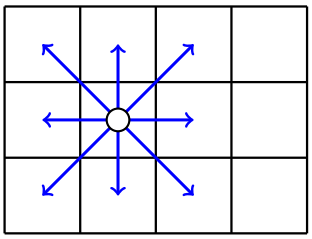


Рис. 1: Возможные перемещения робота в восьми направлениях.

После того как роботы из всех доставленных партий размещаются на участке, они активируются и начинают добычу полезных ископаемых. В процессе перемещения в одной клетке может одновременно находиться произвольное количество роботов. Однако после активации в каждой клетке должно находиться не более q роботов.

Руководством проекта получена информация о t партиях роботов, которые могут быть последовательно отправлены на планету. После доставки всех партий роботов, учитывая их ограниченную мобильность, возможна ситуация, что не удастся разместить роботов на участке так, чтобы в каждой клетке оказалось не больше q роботов. Поэтому руководство должно выбрать k первых партий роботов, где $0 \leqslant k \leqslant t$, которые будут полностью доставлены на соответствующие базы. После этого, если $k < t$, следует дополнительно принять z из n_{k+1} роботов следующей, $(k + 1)$-й партии, $0 \leqslant z < n_{k+1}$.

Все полученные таким образом роботы должны с учетом ограничения на мобильность разместиться на участке таким образом, чтобы в каждой клетке было не более q роботов. После этого они будут активированы и начнут добычу полезных ископаемых. Разумеется, руководство проекта старается максимизировать количество роботов, которые будут доставлены на планету, поэтому, с учетом описанных ограничений, требуется максимизировать k, а затем максимизировать z.

Требуется написать программу, которая по размерам участка, числу q, описанию расположения баз, а также количеству запланированных партий роботов и их описанию определяет максимальное число k — количество партий роботов, и затем – максимальное число z – дополнительное количество роботов из $(k + 1)$-й партии, чтобы, доставив роботов на планету, их можно было разместить на участке таким образом, чтобы в каждой клетке оказалось не более q роботов.

Формат входных данных

Первая строка входного файла содержит числа w, h, s и q ($1 \leqslant w, h \leqslant 10^5$, $1 \leqslant s \leqslant 4$, $1 \leqslant q \leqslant 100$). Последующие s строк содержат по два целых числа x_i, y_i и описывают базы специалистов ($1 \leqslant x_i \leqslant w$, $1 \leqslant y_i \leqslant h$).

Следующая строка содержит число t – количество партий роботов ($1 \leqslant t \leqslant 100$). Последующие t строк описывают партии роботов и содержат по 3 целых числа: b_j, n_j и m_j ($1 \leqslant b_j \leqslant s, 1 \leqslant n_j \leqslant w \cdot h \cdot q$, $0 \leqslant m_j < \max(w, h)$).

Формат выходных данных

Требуется вывести два числа: k и z, $0 \leqslant k \leqslant t$. Если $k = t$, то z должно быть равно 0, иначе должно выполняться условие $0 \leqslant z < n_{k+1}$.

Пример

стандартный ввод	стандартный вывод
4 3 2 1 1 1 3 2 3 1 4 1 2 9 1 1 12 2	1 7

Замечание

В приведенном примере описания входных данных следует полностью принять первую партию роботов и дополнительно принять 7 роботов из второй партии. На рис. 2 показано, как можно разместить этих роботов на участке, чтобы в каждой клетке было не более одного робота. Базы специалистов показаны кружками. Клетки, в которых окажутся роботы с базы 1, показаны зелёным, а клетки, в которых окажутся роботы с базы 2, показаны красным цветом.

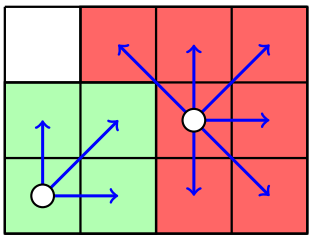


Рис. 2: Возможное размещение роботов на участке в данном примере.