Задача A. Паросочетание

Имя входного файла:	`pairs.in`
Имя выходного файла:	`pairs.out`
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 входного файла

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 выходного файла

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

pairs.in	pairs.out
2 2 1 2 0 2 0	2 1 1 2 2

Задача B. Покрытие путями

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

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

Формат входного файла

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 выходного файла

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 C. День рождения

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

Митя знаком с m юношами и n девушками и хочет пригласить часть из них на свой день рождения. Ему известно, с какими девушками знаком каждый юноша, и с какими юношами знакома каждая девушка. Он хочет добиться того, чтобы каждый приглашённый был знаком со всеми приглашёнными противоположного пола, пригласив при этом максимально возможное число своих знакомых. Помогите ему это сделать!

Формат входного файла

Входной файл состоит из одного или нескольких наборов входных данных. В первой строке входного файла записано число наборов k ($1 \le k \le 20$). В последующих строках записаны сами наборы входных данных.

В первой строке каждого набора задаются числа $0 \le m \le 150$ и $0 \le n \le 150$. Далее следуют m строк, в каждой из которых записано одно или несколько чисел — номера девушек, с которыми знаком i-й юноша (каждый номер встречается не более одного раза). Строка завершается числом 0.

Формат выходного файла

Для каждого набора выведите четыре строки. В первой из них выведите максимальное число знакомых, которых сможет пригласить Митя. В следующей строке выведите количество юношей и количество девушек в максимальном наборе знакомых. Следующие две строки должны содержать номера приглашённых юношей и приглашённых девушек соответственно. Если максимальных наборов несколько, то выведите любой из них.

Примеры

birthday.in	birthday.out
2 2 2 1 2 0 1 2 0 3 2 1 2 0 2 0 1 2 0	4 2 2 1 2 1 2 4 2 2 1 3 1 2

Задача D. Такси

Имя входного файла:	`taxi.in`
Имя выходного файла:	`taxi.out`
Ограничение по времени:	0.5 секунды
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 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.

Примеры

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