

Задача А. Pairs. Паросочетание

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

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

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

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

Формат входных данных

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

Далее следуют n строк с описаниями ребер. i -я вершина из A описана в $i + 1$ -й строке файла. Каждая из этих строк содержит номера вершин из B , соединенных с i -й вершиной A . 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 ребер в максимальном паросочетании. Далее должны следовать l строк, в каждой из которых должны быть два целых числа u_j и v_j — концы ребер паросочетания в A и B , соответственно.

Пример

<code>pairs.in</code>	<code>pairs.out</code>
2 2	2
1 2 0	1 1
2 0	2 2

Задача В. Минимальное контролирующее множество

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

Требуется построить в двудольном графе минимальное контролирующее множество, если дано максимальное паросочетание.

Формат входных данных

В первой строке файла даны два числа m и n ($1 \leq m, n \leq 4000$) — размеры долей. Каждая из следующих m строк содержит список ребер, выходящих из соответствующей вершины первой доли. Этот список начинается с числа K_i ($0 \leq K_i \leq n$) — количества ребер, после которого записаны вершины второй доли, соединенные с данной вершиной первой доли, в произвольном порядке. Сумма всех K_i во входном файле не превосходит 500 000. Последняя строка файла содержит некоторое максимальное паросочетание в этом графе — m чисел $0 \leq L_i \leq n$ — соответствующая i -й вершине первой доли вершина второй доли, или 0, если i -я вершина первой доли не входит в паросочетание.

Формат выходных данных

Первая строка содержит размер минимального контролирующего множества. Вторая строка содержит количество вершин первой доли S , после которого записаны S чисел — номера вершин первой доли, входящих в контролирующее множество, в возрастающем порядке. Третья строка содержит описание вершин второй доли в аналогичном формате.

Пример

<code>minimal.in</code>	<code>minimal.out</code>
3 2	2
2 1 2	1 1
1 2	1 2
1 2	
1 2 0	

Задача С. Испорченный паркет

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

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 входных данных

Первая строка входного файла содержит 4 числа N , M , A , B ($1 \leq N, M \leq 300$, 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 выходных данных

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

Замечание

Чтобы пройти TL, нужно писать умного Куна.

Или использовать жадную инициализацию.

Или пореже обнулять пометки.

Задача D. Такси

Имя входного файла: `taxi.in`
Имя выходного файла: `taxi.out`
Ограничение по времени: 0.5 секунд
Ограничение по памяти: 256 мегабайт

Управлять службой такси — совсем 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 минимизировать 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.

Пример

<code>taxi.in</code>	<code>taxi.out</code>
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