# Задача A. Домино

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

Набор домино состоит из прямоугольных костяшек, каждая из которых разделена на две половинки линией, параллельной более короткой стороне. На каждой из половинок нарисованы точки, количество которых соответствует числу от 0 до $M$ включительно. На костяшках полного набора домино обозначены все возможные различные пары чисел, например, если $M$ равно 3, то полный набор содержит 10 костяшек: (0, 0), (0, 1), (0, 2), (0, 3), (1, 1), (1, 2), (1, 3), (2, 2), (2, 3), (3, 3).

Из костяшек можно выкладывать цепочки, соединяя пары костяшек короткими сторонами, если количества точек на соседних с местом соединения половинках костяшек равны. Некоторые костяшки были удалены из полного набора. Требуется определить, какое минимальное количество цепочек нужно выложить из оставшихся в наборе костяшек, чтобы каждая из них принадлежала ровно одной цепочке.

Напишите программу, которая по информации о наборе домино должна ответить, какое минимальное количество цепочек нужно выложить.

## Формат входного файла

В первой строке входного файла содержится одно целое число $M$ ($0 \leqslant M \leqslant 100$), которое соответствует максимально возможному количеству точек на половинке костяшки.

Во второй строке записано одно целое число $N$, равное количеству костяшек, удаленных из полного набора.

Каждая $i$-я из последующих $N$ строк содержит по два числа $A_i$ и $B_i$. Это количества точек на половинках $i$-й удалённой костяшки.

## Формат выходного файла

Единственная строка выходного файла должна содержать одно целое число $L$ - минимальное количество цепочек.

## Примеры

| domino.in | domino.out |
|---|---|
| 7<br>2<br>7 5<br>3 4 | 2 |
| 0<br>0 | 1 |
| 100<br>5<br>10 10<br>17 93<br>25 71<br>51 51<br>55 55 | 2 |

# Задача B. Ориентация графа

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

Задан неориентированный граф с $N$ вершинами, пронумерованными целыми числами от 1 до $N$. Напишите программу, которая последовательно решает следующие задачи:

а) выясняет количество компонент связности графа;

б) находит и выдает все такие ребра, что удаление любого из них ведет к увеличению числа компонент связности;

в) определяет, можно ли ориентировать все ребра графа таким образом, чтобы получившийся граф оказался сильно связным (ориентированный граф называется сильно связным, если из любой его вершины можно пройти в любую другую, двигаясь по ребрам вдоль стрелок);

г) ориентирует максимальное количество ребер, чтобы получившийся граф оказался сильно связным;

д) определяет минимальное количество ребер, которые следует добавить в граф, чтобы ответ на пункт в) был утвердительным.

## Формат входного файла

Во входном файле записано целое число $N$ ($1 \leqslant N \leqslant 100$) и список ребер графа, заданных номерами концевых вершин.

## Формат выходного файла

Ваша программа должна вывести в выходной файл последовательно ответы на пункты а)-д) в следующем формате:

в первой строке запишите ответ на пункт а);

во второй строке запишите количество ребер из ответа на пункт б), а в последующих строках выдайте сами эти ребра;

в следующую строку выведите сообщение «NOT POSSIBLE», если требуемым в пункте в) способом ориентировать граф невозможно, иначе выведите сообщение «POSSIBLE»;

далее выведите максимальное количество ребер графа, которые можно ориентировать (пункт г); в последующие строки выведите список этих ребер;

в качестве ответа на пункт д) выведите количество ребер, которые следует добавить в исходный граф, а далее выведите сами эти ребра.

Ребра задаются указанием номеров своих концевых вершин, а при выводе ответа на пункт г) должна быть указана их ориентация (вначале выводится номер начальной вершины, затем - номер конечной). Если ответ на пункт а) отличен от единицы, то пункты в) и г) решать не следует и ответы на них не выводятся. Баллы за пункт в) в случае утвердительного ответа на него начисляются лишь в том случае, если программа правильным образом ориентировала ребра графа (пункт г).

### Примеры

| orgraph.in | orgraph.out |
|---|---|
| 4<br>1 2<br>2 4<br>3 4<br>4 1 | 1<br>0<br>POSSIBLE<br>4<br>1 4<br>4 2<br>2 1<br>4 3<br>0 |
| 4<br>1 2<br>2 4<br>3 4<br>4 1 | 1<br>1<br>3 4<br>NOT POSSIBLE<br>3<br>1 2<br>2 4<br>4 1<br>1<br>1 3 |

## Задача C. Кратеры на Луне

| | |
|---|---|
| Имя входного файла: | moon.in |
| Имя выходного файла: | moon.out |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 64 мегабайта |

Пролетающие время от времени в опасной близости от нашего спутника Луны астероиды захватываются ее гравитационным полем и, будучи ничем не задерживаемы, врезаются с огромной скоростью в лунную поверхность, оставляя в память о себе порядочных размеров кратеры приблизительно круглой формы. Увлекающийся астрономией профессор З. В. Ездочетов занялся изучением современной карты участка лунной поверхности. Он решил найти на ней максимально длинную цепочку вложенных друг в друга кратеров. Зная о Ваших недюжинных способностях в области построения алгоритмов, за помощью в решении этой непростой задачи он обратился к Вам.

*Идея решения.* Построим граф, вершины которого соответствуют кратерам, а дуга из вершины $i$; в вершину $j$ идет в том и только том случае, когда $i$-ый кратер целиком лежит внутри $j$-го. По условию задачи нам нужно найти длиннейший путь в полученном ориентированном ациклическом графе. Для этого применяем метод динамического программирования, используя схему топологической сортировки.

### Формат входного файла

Первая строка содержит целое число $N$ - количество кратеров, отмеченных на карте ($1 \leqslant N \leqslant 500$). Следующие $N$ строк содержат описания кратеров с номерами от 1 до $N$. Описание каждого кратера занимает отдельную строку и состоит из трех целых чисел, принадлежащих диапазону $[-32768, 32767]$ и разделенных пробелами. Первые два числа представляют собой декартовы координаты его центра, а третье - радиус. Все кратеры различны.

### Формат выходного файла

Первая строка должна содержать длину искомой цепочки кратеров, вторая - номера кратеров из этой цепочки, начиная с меньшего кратера и кончая самым большим. Номера кратеров должны быть разделены пробелами. Если существует несколько длиннейших цепочек, следует вывести любую из них.

### Примеры

| moon.in | moon.out |
|---|---|
| 3<br>0 0 1<br>4 -1 1<br>5 -1 2 | 2<br>2 3 |
| 4<br>0 0 5<br>-2 0 1<br>1 0 3<br>-1 0 3 | 3<br>2 4 1 |
| 4<br>0 0 30<br>-15 15 20<br>15 10 5<br>10 10 10 | 3<br>3 4 1 |