Задача A. Расстояние от корня

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

В заданном корневом дереве найдите вершины, максимально удалённые от корня. Расстоянием между вершинами считается количество рёбер в пути.

Формат входных данных

В первой строке задано n — количество вершин в дереве ($1 \leqslant n \leqslant 100$). В следующих $n-1$ строках заданы вершины, являющиеся предками вершин 2, 3, …, n. Вершина 1 является корнем дерева.

Формат выходных данных

В первой строке выведите максимальное расстояние от корня до остальных вершин дерева. Во второй строке выведите, сколько вершин дерева находятся от корня на таком расстоянии. В третьей строке выведите номера этих вершин через пробел в порядке возрастания.

Примеры

rootdist.in	rootdist.out
3 1 1	1 2 2 3
3 1 2	2 1 3

Задача B. Связность

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

В этой задаче требуется проверить, что граф является *связным*, то есть что из любой вершины можно по рёбрам этого графа попасть в любую другую.

Формат входных данных

В первой строке входного файла заданы числа N и M через пробел — количество вершин и рёбер в графе, соответственно ($1 \leqslant N \leqslant 100$, $0 \leqslant M \leqslant 10\,000$). Следующие M строк содержат по два числа u_i и v_i через пробел ($1 \leqslant u_i, v_i \leqslant N$); каждая такая строка означает, что в графе существует ребро между вершинами u_i и v_i.

Формат выходных данных

Выведите "`YES`", если граф является связным, и "`NO`" в противном случае.

Примеры

connect.in	connect.out
3 2 1 2 3 2	YES
3 1 1 3	NO

Задача C. Дейкстра

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

Дан ориентированный взвешенный граф. Найдите кратчайшее расстояние от одной заданной вершины до другой.

Формат входных данных

В первой строке входного файла три числа: N, S и F ($1 \leqslant N \leqslant 1000$, $1 \leqslant S, F \leqslant N$), где N — количество вершин графа, S — начальная вершина, а F — конечная. В следующих N строках по N чисел — матрица смежности графа, где -1 означает отсутствие ребра между вершинами, а любое неотрицательное число — присутствие ребра данного веса. На главной диагонали матрицы всегда стоят нули. Веса всех рёбер целые и не превосходят 10 000.

Формат выходных данных

Вывести искомое расстояние или -1, если пути не существует.

Пример

dijkstra.in	dijkstra.out
3 1 2 0 -1 2 3 0 -1 -1 4 0	6

Задача D. Флойд

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

Дан ориентированный взвешенный граф. В нём вам необходимо найти пару вершин, кратчайшее расстояние от одной из которых до другой максимально среди всех пар вершин (u, v), для которых v достижима из u. Гарантируется, что хотя бы одна такая пара существует.

Формат входных данных

В первой строке входного файла единственное число: N ($1 \leqslant N \leqslant 100$) — количество вершин графа. В следующих N строках по N чисел — матрица смежности графа, где -1 означает отсутствие ребра между вершинами, а любое неотрицательное число — присутствие ребра данного веса. На главной диагонали матрицы всегда нули.

Формат выходных данных

Вывести искомое максимальное кратчайшее расстояние.

Пример

floyd.in	floyd.out
4 0 5 9 -1 -1 0 2 8 -1 -1 0 7 4 -1 -1 0	16

Задача E. Порядок циклов

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

Дан неориентированный граф из n вершин, заданный матрицей смежности `a` (`a[u][u] = True`; `a[u][v] = a[v][u]`; `a[u][v] = True` тогда и только тогда, когда есть ребро между вершинами `u` и `v`). На нём запускают следующий алгоритм:

```
for x := 1 to n do
  for y := 1 to n do
    for z := 1 to n do
      if a[i][k] and a[k][j] then
        a[i][j] := True;
```

Перед запуском буквы `x`, `y` и `z` заменяют буквами `i`, `j` и `k` в некотором порядке. Утверждается, что после работы этого алгоритма `a[u][v] = True` тогда и только тогда, когда в исходном графе существует путь между вершинами `u` и `v`. Выясните, верно ли это, и если нет, приведите пример исходного графа, на котором это неверно.

Формат входных данных

В первой строке входного файла записаны через пробел три буквы — '`i`', '`j`' и '`k`' — в некотором порядке. Первая буква подставляется в программу вместо '`x`', вторая — вместо '`y`', третья — вместо '`z`'.

Формат выходных данных

Если искомый граф существует, в первой строке выходного файла выведите через пробел целые числа n и m — количество вершин и рёбер в графе, соответственно ($1 \leqslant n \leqslant 10$, $0 \leqslant m \leqslant 45$). В следующих m строках выведите пары вершин, соединённых рёбрами, по одной паре на строке. Номера вершин в паре должны быть упорядочены по возрастанию; вершины нумеруются с единицы. Кратные рёбра и петли не допускаются.

Если же программа с заданным порядком циклов корректно работает на любом графе, вместо n и m выведите в первой строке два нуля через пробел.

Пример

order.in	order.out
k i j	0 0

Задача F. Цикл отрицательного веса

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

Дан ориентированный граф. Определите, есть ли в нём цикл отрицательного веса, и если да, то выведите его.

Формат входных данных

Во входном файле в первой строке число N ($1 \leqslant N \leqslant 100$) — количество вершин графа. В следующих N строках находится по N чисел — матрица смежности графа. Все веса рёбер не превышают по модулю 10 000. Если ребра нет, то соответствующее число равно 100 000.

Формат выходных данных

В первой строке выходного файла выведите «`YES`», если цикл существует или «`NO`» в противном случае. При его наличии выведите во второй строке количество вершин в искомом цикле и в третьей строке — вершины, входящие в этот цикл, в порядке обхода.

Пример

negcycle.in	negcycle.out
2 0 -1 -1 0	YES 2 1 2