

## Задача А. Компоненты связности

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

Дан неориентированный невзвешенный граф. Необходимо посчитать количество его компонент связности.

### Формат входных данных

В первой строке входного файла содержится одно натуральное число  $N$  ( $N \leq 100$ ) — количество вершин в графе. Далее в  $N$  строках по  $N$  чисел — матрица смежности графа: в  $i$ -й строке на  $j$ -м месте стоит «1», если вершины  $i$  и  $j$  соединены ребром, и «0», если ребра между ними нет. На главной диагонали матрицы стоят нули. Матрица симметрична относительно главной диагонали.

### Формат выходных данных

Вывести одно целое число — искомое количество компонент связности графа.

### Примеры

<code>components.in</code>	<code>components.out</code>
6 0 1 1 0 0 0 1 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0	3

## Задача В. Долой списывание!

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

Во время теста Павел Олегович заметил, что некоторые лкшата обмениваются записками. Сначала он хотел поставить им всем двойки, но в тот день Павел Олегович был добрым, а потому решил разделить лкшат на две группы: списывающих и дающих списывать, и поставить двойки только первым.

У Павла Олеговича записаны все пары лкшат, обменявшихся записками. Требуется определить, сможет ли он разделить лкшат на две группы так, чтобы любой обмен записками осуществлялся от лкшонка одной группы лкшонку другой группы.

### Формат входных данных

В первой строке находятся два числа  $N$  и  $M$  — количество лкшат и количество пар лкшат, обменивающихся записками ( $1 \leq N \leq 100$ ,  $0 \leq M \leq \frac{N(N-1)}{2}$ ). Далее в  $M$  строках расположены описания пар лкшат: два различных числа, соответствующие номерам лкшат, обменивающихся записками (нумерация лкшат идёт с 1). Каждая пара лкшат перечислена не более одного раза.

### Формат выходных данных

Необходимо вывести ответ на задачу Павла Олеговича. Если возможно разделить лкшат на две группы, выведите «YES»; иначе выведите «NO».

### Примеры

bipartite.in	bipartite.out
3 2 1 2 2 3	YES

## Задача С. Компоненты связности - 2

Имя входного файла: `matrix2.in`  
Имя выходного файла: `matrix2.out`  
Ограничение по времени: 2 секунды  
Ограничение по памяти: 64 мегабайта

Дан неориентированный невзвешенный граф. Необходимо посчитать количество его компонент связности и вывести их.

### Формат входных данных

Во входном файле записано два числа  $N$  и  $M$  ( $0 < N \leq 100\,000$ ),  $0 \leq M \leq 100\,000$ ). В следующих  $M$  строках записаны по два числа  $i$  и  $j$  ( $1 \leq i, j \leq N$ ), которые означают, что вершины  $i$  и  $j$  соединены ребром.

### Формат выходных данных

В первой строчке выходного файла выведите количество компонент связности. Далее выведите сами компоненты связности в следующем формате: в первой строке количество вершин в компоненте, во второй — сами вершины в произвольном порядке.

### Примеры

<code>matrix2.in</code>	<code>matrix2.out</code>
6 4	3
3 1	3
1 2	1 2 3
5 4	2
2 3	4 5
	1
	6

## Задача D. Поиск цикла

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

Дан ориентированный невзвешенный граф. Необходимо определить есть ли в нём циклы, и если есть, то вывести любой из них.

### Формат входных данных

В первой строке входного файла находятся два натуральных числа  $N$  и  $M$  ( $1 \leq N \leq 100\,000$ ,  $M \leq 100\,000$ ) — количество вершин и рёбер в графе соответственно. Далее в  $M$  строках перечислены рёбра графа. Каждое ребро задаётся парой чисел — номерами начальной и конечной вершин соответственно.

### Формат выходных данных

Если в графе нет цикла, то вывести «NO», иначе — «YES» и затем перечислить все вершины в порядке обхода цикла.

### Примеры

<code>cycle.in</code>	<code>cycle.out</code>
2 2 1 2 2 1	YES 1 2
2 2 1 2 1 2	NO

## Задача E. TopSort

Имя входного файла: `topsort.in`  
Имя выходного файла: `topsort.out`  
Ограничение по времени: 3 секунды  
Ограничение по памяти: 64 мегабайта

Дан ориентированный невзвешенный граф. Необходимо его топологически отсортировать.

### Формат входных данных

В первой строке входного файла даны два натуральных числа  $N$  и  $M$  ( $1 \leq N \leq 10^5, 1 \leq M \leq 10^5$ ) — количество вершин и рёбер в графе соответственно. Далее в  $M$  строках перечислены рёбра графа. Каждое ребро задаётся парой чисел — номерами начальной и конечной вершин соответственно.

### Формат выходных данных

Вывести любую топологическую сортировку графа в виде последовательности номеров вершин. Если граф невозможно топологически отсортировать, требуется вывести  $-1$ .

### Примеры

<code>topsort.in</code>	<code>topsort.out</code>
6 6 1 2 3 2 4 2 2 5 6 5 4 6	4 6 3 1 2 5
3 3 1 2 2 3 3 1	-1

## Задача F. Кратчайший путь

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

Дан ориентированный взвешенный ациклический граф. Требуется найти в нем кратчайший путь из вершины  $s$  в вершину  $t$ .

### Формат входных данных

Первая строка входного файла содержит четыре целых числа  $n$ ,  $m$ ,  $s$  и  $t$  — количество вершин, дуг графа, начальная и конечная вершина соответственно. Следующие  $m$  строк содержат описания дуг по одной на строке. Ребро номер  $i$  описывается тремя натуральными числами  $b_i$ ,  $e_i$  и  $w_i$  — началом, концом и длиной дуги соответственно ( $1 \leq b_i, e_i \leq n$ ,  $|w_i| \leq 1000$ ).

Входной граф не содержит циклов и петель.

$1 \leq n \leq 100\,000$ ,  $0 \leq m \leq 200\,000$ ,  $1 \leq s, t \leq n$ .

### Формат выходных данных

Первая строка выходного файла должна содержать одно целое число — длину кратчайшего пути из  $s$  в  $t$ . Если пути из  $s$  в  $t$  не существует, выведите `Unreachable`.

### Примеры

<code>dag-shortpath.in</code>	<code>dag-shortpath.out</code>
2 1 1 2 1 2 -10	-10
2 1 2 1 1 2 -10	Unreachable