

## Задача А. Топологическая сортировка (1 балл) (!)

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

Дан ориентированный невзвешенный граф. Необходимо его топологически отсортировать.

### Формат входного файла

В первой строке входного файла даны два натуральных числа  $N$  и  $M$  ( $1 \leq N \leq 100\,000$ ,  $0 \leq M \leq 100\,000$ ) — 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 выходного файла

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

### Пример

<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 В. Поиск цикла (2 балла)

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

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

### Формат входного файла

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 выходного файла

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

### Пример

cycle.in	cycle.out
2 2 1 2 2 1	YES 2 1
2 2 1 2 1 2	NO

## Задача С. Двудольный граф (1 балл)

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

Двудольным называется неориентированный граф  $\langle V, E \rangle$ , вершины которого можно разбить на два множества  $L$  и  $R$ , так что  $L \cap R = \emptyset$ ,  $L \cup R = V$  и для любого ребра  $(u, v) \in E$  либо  $u \in L, v \in R$ , либо  $v \in L, u \in R$ .

Дан неориентированный граф. Требуется проверить, является ли он двудольным.

### Формат входного файла

Первая строка входного файла содержит два натуральных числа  $n$  и  $m$  — количество вершин и ребер графа соответственно ( $1 \leq n \leq 100\,000$ ,  $0 \leq m \leq 200\,000$ ).

Следующие  $m$  строк содержат описание ребер по одному на строке. Ребро номер  $i$  описывается двумя натуральными числами  $b_i, e_i$  — номерами концов ребра ( $1 \leq b_i, e_i \leq n$ ). Допускаются петли и параллельные ребра.

### Формат выходного файла

В единственной строке выходного файла выведите «YES», если граф является двудольным и «NO» в противном случае.

### Пример

bipartite.in	bipartite.out
4 4 1 2 1 3 2 4 4 2	YES
3 3 1 2 2 3 3 1	NO

## Задача D. Конденсация графа (2 балла)

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

Дан ориентированный невзвешенный граф. Необходимо выделить в нем компоненты сильной связности и топологически их отсортировать.

### Формат входного файла

В первой строке входного файла находятся два натуральных числа  $N$  и  $M$  ( $1 \leq N \leq 20\,000, 1 \leq M \leq 200\,000$ ) — 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 выходного файла

Первая строка выходного файла должна содержать целое число  $k$  — количество компонент сильной связности в графе. Вторая строка выходного файла должна содержать  $n$  чисел — для каждой вершины выведите номер компоненты сильной связности, которой она принадлежит. Компоненты должны быть занумерованы таким образом, чтобы для каждого ребра  $(u, v)$  номер компоненты, которой принадлежит  $u$  не превосходил номер компоненты, которой принадлежит  $v$ .

### Пример

cond.in	cond.out
6 7	2
1 2	1 1 1 2 2 2
2 3	
3 1	
4 5	
5 6	
6 4	
2 4	

## Задача Е. Гамильтонов путь (2 балла)

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

Дан ориентированный граф без циклов. Требуется проверить, существует ли в нем путь, проходящий по всем вершинам.

### Формат входного файла

Первая строка входного файла содержит два целых числа  $n$  и  $m$  — количество вершин и дуг графа соответственно. Следующие  $m$  строк содержат описания дуг по одной на строке. Ребро номер  $i$  описывается двумя натуральными числами  $b_i$  и  $e_i$  — началом и концом дуги соответственно ( $1 \leq b_i, e_i \leq n$ ).

Входной граф не содержит циклов и петель.

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

### Формат выходного файла

Если граф удовлетворяет требуемому условию, то выведите YES, иначе NO.

### Пример

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

## Задача F. Игра (2 балла)

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

Дан ориентированный невзвешенный ациклический граф. На одной из вершин графа стоит «фишка». Двое играют в игру. Пусть «фишка» находится в вершине  $u$ , и в графе есть ребро  $(u, v)$ . Тогда за ход разрешается перевести «фишку» из вершины  $u$  в вершину  $v$ . Проигрывает тот, кто не может сделать ход.

### Формат входного файла

В первой строке входного файла находятся три натуральных числа  $N$ ,  $M$  и  $S$  ( $1 \leq N, S, M \leq 100\,000$ ) — количество вершин, рёбер и вершина, в которой находится «фишка» в начале игры соответственно. Далее в  $M$  строках перечислены рёбра графа. Каждое ребро задаётся парой чисел — номерами начальной и конечной вершин.

### Формат выходного файла

Если выигрывает игрок, который ходит первым, выведите «First player wins», иначе — «Second player wins».

### Пример

game.in	game.out
3 3 1 1 2 2 3 1 3	First player wins
3 2 1 1 2 2 3	Second player wins