

Задача А. Поиск цикла

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

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	YES
1 2	1 2
2 1	
2 2	NO
1 2	
1 2	

Задача В. TopSort. Топологическая сортировка

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

Дан ориентированный невзвешенный граф. Необходимо его топологически отсортировать.

Формат входных данных

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

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	4 6 3 1 2 5
1 2	
3 2	
4 2	
2 5	
6 5	
4 6	
3 3	-1
1 2	
2 3	
3 1	

Задача С. Unique Topsort

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

Дан ориентированный ациклический граф G . Проверить, что существует единственный топологический порядок вершин графа.

Формат входных данных

Первая строка входных данных содержит число вершин графа n ($1 \leq n \leq 100\,000$) и число ребер графа m ($0 \leq m \leq 100\,000$). Следующие m строк содержат пары чисел от 1 до n , задающие начало и конец соответствующего ребра. Гарантируется, что граф не содержит циклов.

Формат выходных данных

Если топологический порядок единственный, выведите на первой строке YES, а на второй номера вершин в топологическом порядке, иначе выведите NO.

Примеры

unitopsort.in	unitopsort.out
1 0	YES 1
2 1 2 1	YES 2 1
4 2 1 2 4 3	NO

Задача D. Condense 2. Конденсация графа

Имя входного файла: condense2.in
Имя выходного файла: condense2.out
Ограничение по времени: 1 секунда
Ограничение по памяти: 256 мегабайт

Требуется найти количество ребер в конденсации ориентированного графа. Примечание: конденсация графа не содержит кратных ребер.

Формат входных данных

Первая строка входного файла содержит два натуральных числа n и m — количество вершин и ребер графа соответственно ($1 \leq n \leq 10\,000$, $1 \leq m \leq 100\,000$). Следующие m строк содержат описание ребер, по одному на строке. Ребро номер i описывается двумя натуральными числами b_i , e_i — началом и концом ребра соответственно ($1 \leq b_i, e_i \leq n$). В графе могут присутствовать кратные ребра и петли.

Формат выходных данных

Первая строка выходного файла должна содержать одно число — количество ребер в конденсации графа.

Пример

condense2.in	condense2.out
4 4	
2 1	2
3 2	
2 3	
4 3	

Задача E. Почтальон

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

В городе \mathfrak{C} есть n площадей, соединенных улицами. При этом количество улиц не превышает ста тысяч и существует не более трех площадей, на которые выходит нечетное число улиц. Для каждой улицы известна ее длина. По улицам разрешено движение в обе стороны. В городе есть хотя бы одна улица. От любой площади до любой можно дойти по улицам.

Почтальону требуется пройти хотя бы один раз по каждой улице так, чтобы длина его пути была наименьшей. Он может начать движение на любой площади и закончить также на любой (в том числе и на начальной).

Формат входных данных

Первая строка входного файла содержит натуральное число n — количество площадей в городе ($1 \leq n \leq 1\,000$). Далее следуют n строк, задающих улицы. В i -й из этих строк находится число m_i — количество улиц, выходящих из площади i . Далее следуют m_i пар положительных чисел. В j -й паре первое число — номер площади, в которую идет j -я улица с i -й площади, а второе число — длина этой улицы.

Между двумя площадями может быть несколько улиц, но не может быть улиц с площади на нее саму.

Все числа во входном файле не превосходят 10^5 .

Формат выходных данных

Если решение существует, то в первую строку выходного файла выведите одно число — количество улиц в искомом маршруте (считая первую и последнюю), а во вторую — номера площадей в порядке их посещения.

Если решений нет, выведите в выходной файл одно число -1.

Если решений несколько, выведите любое.

Примеры

post.in	post.out
4	5
2 2 1 2 2	1 2 4 3 2 1
4 1 2 4 4 3 5 1 1	
2 2 5 4 8	
2 3 8 2 4	

Задача F. Avia. Авиаперелеты

Имя входного файла: avia.in

Имя выходного файла: avia.out

Ограничение по времени: 2 секунды

Ограничение по памяти: 256 мегабайт

Главного конструктора Петю попросили разработать новую модель самолета для компании «Air Бубундия». Оказалось, что самая сложная часть заключается в подборе оптимального размера топливного бака.

Главный картограф «Air Бубундия» Вася составил подробную карту Бубундии. На этой карте он отметил расход топлива для перелета между каждой парой городов.

Петя хочет сделать размер бака минимально возможным, для которого самолет сможет долететь от любого города в любой другой (возможно, с дозаправками в пути).

Формат входных данных

Первая строка входного файла содержит натуральное число n ($1 \leq n \leq 1000$) — число городов в Бубундии. Далее идут n строк по n чисел каждая. j -ое число в i -ой строке равно расходу топлива при перелете из i -ого города в j -ый. Все числа не меньше нуля и меньше 10^9 . Гарантируется, что для любого i в i -ой строчке i -ое число равно нулю.

Формат выходных данных

Первая строка выходного файла должна содержать одно число — оптимальный размер бака.

Пример

avia.in	avia.out
4	
0 10 12 16	
11 0 8 9	
10 13 0 22	
13 10 17 0	10