Содержание

У вас не получается читать/выводить данные?
Воспользуйтесь примерами (c++) (python).

Обратите внимание, входные данные лежат в **стандартном потоке ввода** (он же stdin), вывести ответ нужно в **стандартный поток вывода** (он же stdout).

Обратите внимание на `GNU C++` компиляторы с суффиксом `inc`.
Подни можно пользоваться дополнительной библиотекой (`optimization.h`).
То есть, использовать быстрый ввод-вывод: **пример про числа и строки**.
И быструю аллокацию памяти (ускоряет vector-set-map-весь-STL): пример.

Для тех, кто хочет разобраться, как всё это работает.
Короткая версия быстрого ввода-вывода (тык) и короткая версия аллокатора (тык).

Must have

Задача 14A. Longpath. Длиннейший путь [1, 256]

Дан ориентированный граф без циклов. Требуется найти в нем длиннейший путь.

Формат входных данных

Первая строка входного файла содержит два натуральных 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 ($1 \leqslant b_i, e_i \leqslant n$).

Входной граф не содержит циклов и петель.

$n \leqslant 10\,000$, $m \leqslant 100\,000$.

Формат выходных данных

Первая строка выходного файла должна содержать одно натуральное число — количество дуг в длиннейшем пути.

Пример

stdin	stdout
5 5 1 2 2 3 3 4 3 5 1 5	3

Задача 14B. Поиск цикла [1, 256]

Дан ориентированный невзвешенный граф без петель и кратных рёбер. 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 ($1 \leqslant N \leqslant 100\,000$, $M \leqslant 100\,000$) — 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

stdin	stdout
2 2 1 2 2 1	YES 1 2
3 3 1 2 2 3 1 3	NO

Обязательные задачи

Задача 14C. Unique Topsort [1, 256]

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

Формат входных данных

Первая строка входных данных содержит число вершин графа n ($1 \leqslant n \leqslant 100\,000$) и число ребер графа m ($0 \leqslant m \leqslant 100\,000$). 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

stdin	stdout
1 0	YES 1
2 1 2 1	YES 2 1
4 2 1 2 4 3	NO

Задача 14D. Condense 2. Конденсация графа [1, 256]

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 ($n \leqslant 10\,000,\ m \leqslant 100\,000$). 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 ($1 \leqslant b_i,\ e_i \leqslant n$). 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

stdin	stdout
4 4 2 1 3 2 2 3 4 3	2

Задача 14E. Из истории банка Гринготтс [1, 256]

Чтобы понять название задачи, можно прочитать красивую легенду.
http://acm.timus.ru/problem.aspx?space=1&num=1441

Задача же заключается в том, чтобы рёбра неориентированного графа разбить на минимальное число путей.

Формат входных данных

Дан граф. На первой строке число вершин n ($1 \leqslant n \leqslant 20\,000$) и число рёбер m ($1 \leqslant m \leqslant 20\,000$). Следующие m строк содержат описание рёбер графа. Каждая строка по два числа a_i b_i ($1 \leqslant a_i, b_i \leqslant n$). Между каждыми двумя вершинами не более одного ребра. Граф связен.

Формат выходных данных

На первой строке минимальное число путей. На каждой следующей описание очередного пути – номера вершин в порядке прохождения.

Примеры

stdin	stdout
7 7 1 2 4 1 6 7 5 7 7 4 2 3 4 2	3 5 7 4 2 1 4 2 3 6 7

Задача 14F. Bridges. Мосты [1, 256]

Дан неориентированный граф. Требуется найти все мосты в нем.

Формат входных данных

Первая строка входного файла содержит два натуральных числа n и m — количество вершин и ребер графа соответственно ($n \leqslant 20\,000$, $m \leqslant 200\,000$).

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

Формат выходных данных

Первая строка выходного файла должна содержать одно натуральное число b — количество мостов в заданном графе. На следующей строке выведите b целых чисел — номера ребер, которые являются мостами, в возрастающем порядке. Ребра нумеруются с единицы в том порядке, в котором они заданы во входном файле.

Пример

stdin	stdout
6 7 1 2 2 3 3 4 1 3 4 5 4 6 5 6	1 3

Задача 14G. Мосты и компоненты [1, 256]

Дан неориентированный граф (не обязательно связный). Граф может содержать петли и кратные ребра.

Выведите все компоненты реберной двусвязности графа (максимальные подмножества вершин, такие что подграф на них не теряет связность при удалении любого ребра).

Формат входных данных

Первая строка содержит числа n и m ($1 \leqslant n \leqslant 100\,000$, $0 \leqslant m \leqslant 100\,000$) — количество вершин и ребер в графе.

Следующие m строк задают ребра графа.

Формат выходных данных

В первой строке выведите количество компонент, в следующих за ней строках выведите сами компоненты, по одной на строку.

Вершины в каждой компоненте должны идти в возрастающем порядке, компоненты нужно вывести в лексикографическом порядке.

Компонента – вектор номеров своих вершин. Лексикографически сравниваются вектора.

Примеры

stdin	stdout
3 2 1 2 2 3	3 1 2 3
3 3 1 2 2 3 3 1	1 1 2 3
2 2 1 2 1 2	1 1 2
7 8 1 5 5 6 1 6 5 4 4 3 4 2 3 2 7 2	3 1 5 6 2 3 4 7

Дополнительные задачи

Задача 14H. Кодовый замок [1, 256]

Петя опоздал на тренировку по программированию! Поскольку тренировка проходит в воскресенье, главный вход в учебный корпус, где она проходит, оказался закрыт, а вахтёр ушёл куда-то по своим делам. К счастью, есть другой способ проникнуть в здание — открыть снаружи боковую дверь, на которой установлен кодовый замок.

На пульте замка есть d кнопок с цифрами $0, 1, \ldots, d-1$. Известно, что код, открывающий замок, состоит из k цифр. Замок открывается, если последние k нажатий кнопок образуют код.

Поскольку Петя не имеет понятия, какой код открывает замок, ему придётся перебрать все возможные коды из k цифр. Но, чтобы как можно скорее попасть на тренировку, нужно минимизировать количество нажатий на кнопки. Помогите Пете придумать такую последовательность нажатий на кнопки, при которой все возможные коды были бы проверены, а количество нажатий при этом оказалось бы минимально возможным.

Формат входных данных

В первой строке входного файла записаны через пробел два целых числа d и k — количество кнопок на пульте и размер кода, соответственно ($2 \leqslant d \leqslant 10$, $1 \leqslant k \leqslant 20$).

Формат выходных данных

В первой строке выходного файла выведите искомую последовательность. Если последовательностей минимальной длины, перебирающих все возможные коды, несколько, можно выводить любую из них. Гарантируется, что d и k таковы, что минимальная длина последовательности не превосходит 1 мебибайта.

Пример

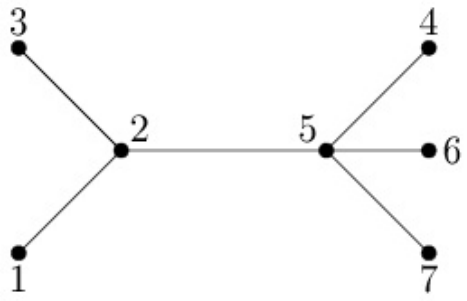
stdin	stdout
2 3	0001011100

Пояснение к примеру

Последовательность в примере перебирает все коды длины 3 в следующем порядке: 000, 001, 010, 101, 011, 111, 110, 100.

Задача 14I. Autotourism [1, 256]

В Байтландии существуют n городов, соединённых $n-1$ дорогами с двусторонним движением таким образом, что из каждого города можно проехать в любой другой по сети дорог. Длина каждой дороги равна 1 километру.



Бензобак автомобиля позволяет проехать без заправки m километров. Требуется выбрать маршрут, позволяющий посетить наибольшее количество различных городов без дозаправки. При этом начинать и заканчивать маршрут можно в произвольных городах.

Формат входных данных

В первой строке входного файла заданы два целых числа n и m ($2 \le n \le 500\,000$, $1 \le m \le 200\,000\,000$) — количество городов в стране и количество километров, которое автомобиль может проехать без дозаправки. В последующих $n-1$ строках описаны дороги. Каждая дорога задаётся двумя целыми числами a и b ($1 \leqslant a, b \leqslant n$) — номерами городов, которые она соединяет. Длина каждой дороги равна 1 км.

Формат выходных данных

Выведите одно число — максимальное количество городов, которое можно посетить без дозаправки.

Пример

stdin	stdout
7 6 1 2 2 3 2 5 5 6 5 7 5 4	5

Пояснение к примеру

5 городов можно посетить, например, по схеме $4 \to 5 \to 7 \to 5 \to 6 \to 5 \to 2$ или по схеме $3 \to 2 \to 1 \to 2 \to 5 \to 6 \to 5$.

Задача 14J. King's Assassination [1, 256]

Дан граф из n вершин и m ребер. Граф ориентированный. Нужно определить число вершин, содержащихся на всех путях из s в t (сами s и t учитывать не нужно).

Формат входных данных

Первая строка содержит n, m, s и t ($2 \leqslant n \leqslant 100\,000$, $1 \leqslant m \leqslant 300\,000$, $1 \leqslant s, t \leqslant n$, $s \neq t$).

Следующие m строк содержат пары чисел x_i и y_i — индексы вершин от 1 до n. Это означает что есть дорога из вершины с номером x_i в вершину с номером y_i.

Формат выходных данных

Число вершин k. Далее k чисел — номера вершин в возрастающем порядке.

Примеры

stdin	stdout
4 3 1 4 1 2 2 3 3 4	2 2 3
4 4 1 4 1 2 2 3 3 4 1 3	1 3
4 5 1 4 1 2 2 3 3 4 1 3 2 4	0