Задача A. Топологическая сортировка

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Задан ориентированный ациклический граф с n вершинами и m ребрами. Также задана перестановка вершин графа. Необходимо проверить, является ли данная перестановка топологической сортировкой.

Формат входных данных

В первой строке даны два числа n и m — количество вершин и ребер в графе соответственно ($1 \leqslant n, m \leqslant 10^5$). В следующих m строках заданы пары чисел u_i, v_i, означающие, что в графе есть ребро из вершины u_i в вершину v_i. В последней строке задана перестановка из n элементов.

Формат выходных данных

Выведите «YES» (без кавычек), если данная перестановка является топологической сортировкой и «NO» в противном случае.

Примеры

стандартный ввод	стандартный вывод
3 3 2 3 1 3 1 2 2 1 3	NO
3 3 3 2 1 2 3 1 3 1 2	YES

Задача B. Topsort

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
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 целых числа N и M ($1 \leqslant N \leqslant 100\,000, 0 \leqslant M \leqslant 100\,000$) — количества 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

стандартный ввод	стандартный вывод
6 6 1 2 3 2 4 2 2 5 6 5 4 6	4 6 3 1 2 5

Задача C. Конденсация графа. Light.

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	0.5 секунд
Ограничение по памяти:	256 мегабайт

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

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

стандартный ввод	стандартный вывод
4 4 2 1 3 2 2 3 4 3	2

Задача D. Конденсация графа

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Вам задан ориентированный граф с N вершинами и M ребрами ($1 \leqslant N \leqslant 200\,000$, $1 \leqslant M \leqslant 200\,000$). Найдите компоненты сильной связности заданного графа и топологически отсортируйте его конденсацию.

Формат входных данных

Граф задан во входном файле следующим образом: первая строка содержит числа N и M. Каждая из следующих M строк содержит описание ребра – два целых числа из диапазона от 1 до N – номера начала и конца ребра.

Формат выходных данных

На первой строке выведите число K – количество компонент сильной связности в заданном графе. На следующей строке выведите N чисел – для каждой вершины выведите номер компоненты сильной связности, которой принадлежит эта вершина. Компоненты сильной связности должны быть занумерованы таким образом, чтобы для любого ребра номер компоненты сильной связности его начала не превышал номера компоненты сильной связности его конца.

Пример

стандартный ввод	стандартный вывод
10 19 1 4 7 8 5 10 8 9 9 6 2 6 6 2 3 8 9 2 7 2 9 7 4 5 3 6 7 3 6 7 10 8 10 1 2 9 2 7	2 1 2 2 1 1 2 2 2 2 1

Задача E. 2-SAT

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Формулировка `2-SAT`: нужно подобрать значения n булевых переменных так, чтобы все m утверждений вида $x_{i_1} = e_1 \vee x_{i_2} = e_2$ обратились в истину. В данной задаче вам гарантируется, что решение существует.

Формат входных данных

Входной файл состоит из одного или нескольких тестов.

Каждый тест описывается следующим образом. На первой строке число переменных n и число утверждений m. Каждая из следующих m строк содержит числа i_1, e_1, i_2, e_2, задает утверждение $x_{i_1} = e_1 \vee x_{i_2} = e_2$ ($0 \leqslant i_j < n$, $0 \leqslant e_j \leqslant 1$). Ограничения: сумма всех n не больше 100 000, сумма всех m не больше 300 000.

Формат выходных данных

Для каждого теста выведите строку из n нулей и единиц — значения переменных. Если у данной задачи `2-SAT` есть несколько решений, выведите любое.

Пример

стандартный ввод	стандартный вывод
1 0 2 2 0 0 1 0 0 1 1 1 3 4 0 1 1 0 0 0 2 1 1 1 2 0 0 0 0 1	0 01 000

Задача F. Яблоко от яблони

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

У Пети в саду растет яблоня. Воодушевленный историей об Исааке Ньютоне, который, как известно, открыл закон всемирного тяготения после того, как ему на голову упало яблоко, Петя с целью повысить свою успеваемость по физике часто сидит под яблоней.

Однако, поскольку по физике у Пети твердая тройка, яблоки с его яблони падают следующим образом. В какой-то момент одно из яблок отрывается от ветки, на которой оно висит, и начинает падать строго вниз. Если в некоторый момент оно задевает другое яблоко, то то тоже отрывается от своей ветки и начинает падать вниз, при этом первое яблоко не меняет направление своего падения. Вообще, если любое падающее яблоко заденет другое яблоко на своем пути, то оно также начнет падать.

Таким образом, в любой момент каждое яблоко либо висит на ветке, либо падает строго вниз, причем все яблоки кроме первого, чтобы начать падать, должны сначала соприкоснуться с каким-либо другим падающим яблоком.

Выясните, какие яблоки упадут с Петиной яблони.

Формат входных данных

В первой строке вводится число N — количество яблок на Петиной яблоне ($1 \leqslant N \leqslant 200$). Следующие N строк содержат описания яблок. Будем считать все яблоки шарами. Каждое яблоко задается координатами своей самой верхней точки (той, где оно исходно прикреплено к дереву, длиной черенка пренебрежем) x_i, y_i и z_i и радиусом r_i ($-10000 \leqslant x_i, y_i, z_i \leqslant 10000$, $1 \leqslant r_i \leqslant 10000$, все числа целые). Гарантируется, что изначально никакие яблоки не пересекаются (даже не соприкасаются). Ось OZ направлена вверх.

Вершины в ответе должны быть отсортированы по возрастанию.

Формат выходных данных

В первой строке выведите количество яблок, которые упадут с яблони, если начнет падать первое яблоко. В следующей строке выводите номера упавших яблок. Яблоки нумеруются, начиная с 1, в том порядке, в котором они заданы во входных данных.

Пример

стандартный ввод	стандартный вывод
4 0 0 10 4 5 0 3 1 -7 4 7 1 0 1 2 6	3 1 2 4

Задача G. Предок

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Напишите программу, которая для двух вершин дерева определяет, является ли одна из них предком другой.

Формат входных данных

Первая строка входного файла содержит натуральное число n ($1 \leqslant n \leqslant 100000$) — количество вершин в дереве. Во второй строке находятся n чисел, i-е из которых определяет номер непосредственного родителя вершины с номером i. Если это число равно нулю, то вершина является корнем дерева.

В третьей строке находится число m ($1 \leqslant m \leqslant 100000$) — количество запросов. Каждая из следующих m строк содержит два различных числа a и b ($1 \leqslant a, b \leqslant n$).

Формат выходных данных

Для каждого из m запросов выведите на отдельной строке число 1, если вершина a является одним из предков вершины b, и 0 в противном случае.

Пример

стандартный ввод	стандартный вывод
6 0 1 1 2 3 3 5 4 1 1 4 3 6 2 6 6 5	0 1 1 0 0

Задача H. Введите одностороннее движение

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

В Тридевятом Царстве было N городов, некоторые из которых были соединены дорогами. К сожалению, в последнее время добраться из одного города в другой стало очень сложно из-за возникших автомобильных пробок. В целях борьбы с пробками было решено все дороги сделать односторонними, т.е. разрешить проезд по каждой дороге только в одном направлении. При этом требуется, чтобы по-прежнему можно было из любого города попасть в любой другой.

Формат входных данных

Во входном файле записано сначала число N — количество городов ($1 \leqslant N \leqslant 1000$). Затем записано число M — количество дорог ($1 \leqslant M \leqslant 100000$). Далее идет M пар чисел, задающих дороги (каждая дорога описывается номерами городов, которые она соединяет). Не бывает дорог из некоторого города в тот же город. Между двумя городами может быть несколько дорог. Гарантируется, что до введения одностороннего движения можно было попасть из любого города в любой другой.

Формат выходных данных

В выходной файл нужно выдать M пар чисел, соответствующих дорогам (дороги должны быть выданы в том же порядке, в котором они заданы во входном файле). Для каждой дороги сначала должен быть записан номер города, из которого по ней можно будет уехать после введения одностороннего движения, а затем — номер города, куда эта дорога ведет.

Если ввести одностороннее движение так, чтобы можно было из любого города попасть в любой другой, нельзя, выходной файл должен содержать одно число 0.

Пример

стандартный ввод	стандартный вывод
4 6 1 2 1 2 2 3 2 4 4 3 1 4	1 2 2 1 2 3 4 2 3 4 4 1

Задача I. Авиалинии

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

В Берляндии скоро появятся свои авиалинии. Комитет по разработке берляндских авиалиний уже предложил свой вариант соединения городов авиарейсами. Каждый авиарейс задается парой различных городов. Рейсы односторонние. Президенту понравился план, однако он показался ему чересчур неэкономным. Требуется разработать новый план, который содержит наименьшее количество авиарейсов и удовлетворяет условию: если из города aможно было попасть в город b (возможно, с пересадками) согласно первоначальному плану, то и в новом плане это должно быть возможным. Если же это было сделать невозможно, то и согласно новому плану это не должно быть возможным. Очевидно, что из любого города можно попасть в него самого.

Формат входных данных

В первой строке входного файла записаны целые числа N и M ($1 \leqslant N \leqslant 10^3$, $0 \leqslant M \leqslant 10^4$), где N — количество городов в стране, а M — количество авиарейсов в первоначальном плане. Города нумеруются от 1 до N. Далее записано M пар различных чисел a_i, b_i обозначающих наличие рейса из a_i в b_i в первоначальном плане ($1 \leqslant a_i \leqslant N$, $1 \leqslant b_i \leqslant N$). Пары разделяются пробелами или переводами строк. Между парой городов может быть более одного авиарейса.

Формат выходных данных

В первую строку выходного файла выведите количество рейсов в новом плане. Далее выведите авиарейсы в формате, аналогичном формату входных данных. Пары разделяйте пробелами или переводами строк. Пары выводите в любом порядке. Если существует несколько решений, выведите любое.

Пример

стандартный ввод	стандартный вывод
4 5 1 2 2 3 2 1 3 2 2 4	4 1 2 2 3 3 1 1 4

Задача J. Монополия

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

В Тридесятом государстве есть N фирм, занимающихся разработкой программного обеспечения. Однажды известный олигарх Тридесятого государства Иванушка решил монополизировать эту отрасль. Для этого он хочет купить максимальное число программистских фирм Тридесятого государства.

Он разослал предложения всем N компаниям и через некоторое время получил от каждой их них согласие или отказ. Однако он знает, что в бизнесе очень многое зависит от взаимного доверия партнеров.

В результате небольшого исследования Иванушка установил, между какими компаниями существует взаимное доверие (причем всегда если компания доверяет компании B, то компания B доверяет компании A).

Теперь, при желании, Иванушка может повторно разослать предложения всем компаниям, включив в письма список компаний, давших согласие участвовать в его проекте. При этом каждая компания, независимо от своего первоначального мнения дает согласие, если в списке есть хотя бы одна компания, которой она доверяет, и отказ в противном случае. Таким образом, некоторые компании, которые изначально не согласились участвовать в проекте, могут теперь дать свое согласие, а некоторые из давших согласие — наоборот отказаться. В результате этого у Иванушки формируется новый список, который он опять может разослать фирмам. Он может сколь угодно долго повторять операцию, каждый раз рассылая текущий список. Иванушка может остановить процесс в любой момент и заключить договора с теми, кто после последней рассылки дал согласие.

Напишите программу, которая определит, какое максимальное число компаний может объединить Иванушка под своим началом.

Будем считать, что Иванушка — честный предприниматель и он никогда не подтасовывает рассылаемые им списки.

Формат входных данных

В первой строке входных данных содержится число N — количество фирм ($1 \leqslant N \leqslant 2000$). Далее идут N чисел, описывающих ответ фирмы на первое предложение Иванушки (1 — согласие, 0 — отказ). Далее задается число M ($0 \leqslant M \leqslant 200000$) — количество пар компаний, между которыми существует доверие. Далее следуют M пар чисел, задающих номера фирм, между которыми существует взаимное доверие (числа в паре не могут быть одинаковыми). Любая пара компаний упоминается в этом списке не более одного раза.

Формат выходных данных

Выведите одно число — максимальное число фирм, которое сможет купить Иванушка.

Пример

стандартный ввод	стандартный вывод
7 1 0 0 0 0 0 1 6 1 2 1 3 1 4 4 5 5 6 2 5	4

Задача K. Водостоки

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Карту местности условно разбили на квадраты, и посчитали среднюю высоту над уровнем моря для каждого квадрата.

Когда идет дождь, вода равномерно выпадает на все квадраты. Если один из четырех соседних с данным квадратом квадратов имеет меньшую высоту над уровнем моря, то вода с текущего квадрата стекает туда (и, если есть возможность, то дальше), если же все соседние квадраты имеют большую высоту, то вода скапливается в этом квадрате.

Разрешается в некоторых квадратах построить водостоки. Когда на каком-то квадрате строят водосток, то вся вода, которая раньше скапливалась в этом квадрате, будет утекать в водосток.

Если есть группа квадратов, имеющих одинаковую высоту и образующих связную область, то если хотя бы рядом с одним из этих квадратов есть квадрат, имеющий меньшую высоту, то вся вода утекает туда, если же такого квадрата нет, то вода стоит во всех этих квадратах. При этом достаточно построить водосток в любом из этих квадратов, и вся вода с них будет утекать в этот водосток.

Требуется определить, какое минимальное количество водостоков нужно построить, чтобы после дождя вся вода утекала в водостоки.

Формат входных данных

Во входном файле записаны сначала числа N и M, задающие размеры карты — натуральные числа, не превышающие 100. Далее идет N строк, по M чисел в каждой, задающих высоту квадратов карты над уровнем моря. Высота задается натуральным числом, не превышающим 10000. Считается, что квадраты, расположенные за пределами карты, имеют высоту 10001 (то есть вода никогда не утекает за пределы карты).

Формат выходных данных

В выходной файл выведите минимальное количество водостоков, которое необходимо построить.

Пример

стандартный ввод	стандартный вывод
4 4 1 2 4 1 2 4 4 4 1 4 3 2 1 2 3 2	4