Задача A. Ряд людей

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

На оси O_x стоит N человек. Известно, что i-й человек стоит в некоторой целой точке x_i, находящейся в пределах от 0 до 10^9 включительно. Возможно, что несколько человек стоят на одной координате.

Вам известно, что для некоторых троек-условий (L, R, D) выполнено, что человек с номером R стоит на D правее, чем человек с номером L, то есть $x_R - x_L = D$.

Определите, существуют ли координаты $x_1, \ldots, x_N$, удовлетворяющие всем условиям.

Формат входных данных

В первой строке даны целые числа N и M ($1 \leqslant N \leqslant 100000$, $0 \leqslant M \leqslant 200000$) — число человек и накладываемых условий соответственно.

Далее следуют M строк. В i-й строке даны три целых числа L_i, R_i, D_i ($1 \leqslant L_i, R_i \leqslant N$, $0 \leqslant D_i \leqslant 10000$) — параметры очередного накладываемого условия. Гарантируется, что про каждую пару человек дано не более одного условия.

Формат выходных данных

В единственной строке выходного файла выведите «Yes», если существуют целые числа $x_1, \ldots, x_N$ в пределах от 0 до 10^9 включительно, удовлетворяющие всем наложенным условиям. В ином случае выведите «No».

Примеры

стандартный ввод	стандартный вывод
3 3 1 2 1 2 3 1 1 3 2	Yes
3 3 1 2 1 2 3 1 1 3 5	No
4 3 2 1 1 2 3 5 3 4 2	Yes
10 3 8 7 100 7 9 100 9 8 100	No
100 0	Yes

Задача B. Древнее заклинание

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

Гидеон нашел древний свиток, позволяющий активировать мощнейшее заклинание, которое помогает решить любую задачу. Осталось провести ритуал.

На свитке изображена прямоугольная таблица, содержащая n строк и m столбцов, в каждой ячейке которой написана латинская буква. Заклинание представляет собой строку, также состоящую из латинских букв. Чтобы активировать заклинание, необходимо бесконечное число раз произнести эту строку, перемещая при этом конец волшебной палочки по таблице на свитке вдоль подходящей циклической последовательности ячеек.

Назовем циклической последовательностью ячеек такую последовательность ячеек $(x_1, y_1), (x_2, y_2) \dots (x_k, y_k)$, что каждая пара соседних в последовательности ячеек, а также первая и последняя ячейки в последовательности, не совпадают и имеют общую сторону. Таким образом, при перемещении конца волшебной палочки вдоль последовательности, каждый раз необходимо переместиться в соседнюю ячейку вверх, влево, вправо или вниз, а в конце — вернуться в начальную ячейку. При этом одна и та же ячейка может встречаться в последовательности несколько раз.

Для активации заклинания требуется найти такую циклическую последовательность ячеек, что если одновременно делать следующее: произность бесконечное число раз строку заклинания, по одному символу в секунду, и, начав с первой ячейки, каждую секунду до бесконечности перемещать конец волшебной палочки к следующей ячейке в последовательности (после последней ячейки следует снова перейти к первой ячейке), то буквы заклинания и буквы в ячейках таблицы всегда будут сопадать.

Помогите Гидеону найти подходящую циклическую последовательность, либо выясните, что такой последовательности не существует.

Формат входных данных

Первая строка содержит целые числа n, m и l — высоту и ширину таблицы, а также длину заклинания, соответственно ($2 \leqslant n, m \leqslant 200$, $1 \leqslant l \leqslant 200$).

В следующих n строках содержится по m строчных латинских букв — содержимое таблицы.

В последней строке содержится строка из l строчных латинских букв — заклинание.

Формат выходных данных

Если искомой последовательности не существует, в единственной строке выведите «`NO`».

Иначе, в первой строке выведите «`YES`». Во второй строке выведите k — длину последовательности, k не должно превышать 10^7. Гарантируется, что если ответ существует, то существует ответ с k, не превышающим 10^7.

В третьей строке выведите координаты первой ячейки последовательности: сначала номер строки, а потом номер столбца, на пересечении который находится эта ячейка. Наконец, в последней строке выведите строку длины k, состоящую из символов «`U`», «`L`», «`D`», «`R`», которая описывает последовательность перемещения конца волшебной палочки по ячейкам в искомой циклической последовательности последовательности.

Символ «`U`» соответствует переходу из текущей ячейки в соседнюю сверху, символ «`L`» — в соседнюю слева, «`D`» — в соседнюю снизу, а «`R`» — в соседнюю справа.

После переходов по всем символам необходимо вернуться в исходную ячейку. Выходить за границы поля не разрешается.

Если ответов несколько, выведите любой.

Примеры

стандартный ввод	стандартный вывод
3 2 4 aa bb cc abcb	YES 4 1 1 DDUU
3 3 5 abc dee ghi eeeee	YES 10 2 2 RLRLRLRLRL
3 3 5 abc def ghi eeeee	NO

Замечание

Пояснение к третьему тесту. По определению, последовательность из одной ячейки не является циклической, поэтому решения не существует.

Задача C. Точки сочленения

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

256 мегабайт

Дан неориентированный граф. Требуется найти все точки сочленения в нём.

Формат входных данных

Первая строка входного файла содержит два натуральных числа n и m — количества вершин и рёбер графа соответственно ($1 \leqslant n \leqslant 20\,000$, $1 \leqslant m \leqslant 200\,000$).

Следующие m строк содержат описание рёбер 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 точек сочленения в заданном графе. На следующей строке выведите b целых чисел — номера вершин, которые являются точками сочленения, в возрастающем порядке.

Пример

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

Задача D. Магнитные подушки

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

Город будущего застроен небоскребами, для передвижения между которыми и парковки транспорта многие тройки небоскребов соединены треугольной подушкой из однополярных магнитов. Каждая подушка соединяет ровно 3 небоскреба и вид сверху на нее представляет собой треугольник, с вершинами в небоскребах. Это позволяет беспрепятственно передвигаться между соответствующими небоскребами. Подушки можно делать на разных уровнях, поэтому один небоскреб может быть соединен различными подушками с парами других, причем два небоскреба могут соединять несколько подушек (как с разными третьими небоскребами, так и с одинаковым). Например, возможны две подушки на разных уровнях между небоскребами 1, 2 и 3, и, кроме того, магнитная подушка между 1, 2, 5.

Система магнитных подушек организована так, что с их помощью можно добираться от одного небоскреба, до любого другого в этом городе (с одной подушки на другую можно перемещаться внутри небоскреба), но поддержание каждой из них требует больших затрат энергии.

Требуется написать программу, которая определит, какие из магнитных подушек нельзя удалять из подушечной системы города, так как удаление даже только этой подушки может привести к тому, что найдутся небоскребы из которых теперь нельзя добраться до некоторых других небоскребов, и жителям станет очень грустно.

Формат входных данных

В первой строке входного файла находятся числа N и M — количество небоскребов в городе и количество работающих магнитных подушек соответственно ($3 \leqslant N \leqslant 100000$, $1 \leqslant M \leqslant 100000$). В каждой из следующих M строк через пробел записаны три числа — номера небоскребов, соединенных подушкой. Небоскребы пронумерованы от 1 до N. Гарантируется, что имеющиеся магнитные подушки позволяют перемещаться от одного небоскреба до любого другого.

Формат выходных данных

Выведите в выходной файл сначала количество тех магнитных подушек, отключение которых невозможно без нарушения сообщения в городе, а потом их номера. Нумерация должна соответствовать тому порядку, в котором подушки перечислены во входном файле. Нумерация начинается с единицы.

Примеры

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

Задача E. 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 F. Эйлеров путь

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

Дан неориентированный связный граф, не более трех вершин имеет нечетную степень. Требуется определить, существует ли в нем путь, проходящий по всем ребрам.

Если такой путь существует, необходимо его вывести.

Формат входных данных

Первая строка входного файла содержит натуральное число n — количество вершин графа ($1 \leqslant n \leqslant 100\,000$). Далее следуют n строк, задающих ребра. В i-й из этих строк находится число m_i — количество ребер, инцидентных вершине i. Далее следуют m_i натуральных чисел — номера вершин, в которые ведут ребро из i-й вершины.

Граф может содержать кратные ребра, но не содержит петель.

Граф содержит не более 300 000 ребер.

Формат выходных данных

Если решение существует, то в первую строку выходного файла выведите одно число k — количество ребер в искомом маршруте, а во вторую $k+1$ число — номера вершин в порядке их посещения.

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

Если решений несколько, выведите любое.

Пример

euler.in	euler.out
4 2 2 2 4 1 4 3 1 2 2 4 2 3 2	5 1 2 3 4 2 1

Задача G. Размещение данных

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

Телекоммуникационная сеть крупной IT-компании содержит n серверов, пронумерованных от 1 до n. Некоторые пары серверов соединены двусторонними каналами связи, всего в сети m каналов. Гарантируется, что сеть серверов устроена таким образом, что по каналам связи можно передавать данные с любого сервера на любой другой сервер, возможно с использованием одного или нескольких промежуточных серверов.

Множество серверов A называется отказоустойчивым, если при недоступности любого канала связи выполнено следующее условие. Для любого не входящего в это множество сервера X существует способ передать данные по остальным каналам на сервер X хотя бы от одного сервера из множества A.

На рис. 1 показан пример сети и отказоустойчивого множества из серверов с номерами 1 и 4. Данные на сервер 2 можно передать следующим образом. При недоступности канала между серверами 1 и 2 — с сервера 4, при недоступности канала между серверами 2 и 3 — с сервера 1. На серверы 3 и 5 при недоступности любого канала связи можно по другим каналам передать данные с сервера 4.

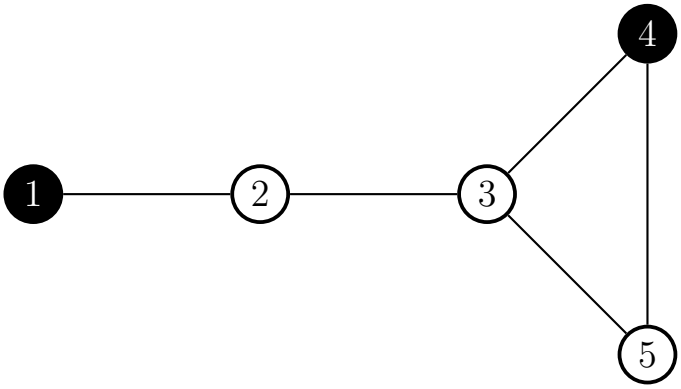


Рис. 1: Пример сети и отказоустойчивого множества серверов.

В рамках проекта группе разработчиков компании необходимо разместить свои данные в сети. Для повышения доступности данных и устойчивости к авариям разработчики хотят продублировать свои данные, разместив их одновременно на нескольких серверах, образующих отказоустойчивое множество. Чтобы минимизировать издержки, необходимо выбрать минимальное по количеству серверов отказоустойчивое множество. Кроме того, чтобы узнать, насколько гибко устроена сеть, необходимо подсчитать количество способов выбора такого множества, и поскольку это количество способов может быть большим, необходимо найти остаток от деления этого количества способов на число $10^9 + 7$.

Требуется написать программу, которая по заданному описанию сети определяет следующие числа: k — минимальное количество серверов в отказоустойчивом множестве серверов, c — остаток от деления количества способов выбора отказоустойчивого множества из k серверов на число $10^9 + 7$

Формат входных данных

Первая строка входного файла содержит целые числа n и m — количество серверов и количество каналов связи соответственно ($2 \leqslant n \leqslant 200\,000$, $1 \leqslant m \leqslant 200\,000$). Следующие m строк содержат по два целых числа и описывают каналы связи между серверами. Каждый канал связи задается двумя целыми числами: номерами серверов, которые он соединяет.

Гарантируется, что любые два сервера соединены напрямую не более чем одним каналом связи, никакой канал не соединяет сервер сам с собой, и для любой пары серверов существует способ

передачи данных с одного из них на другой, возможно с использованием одного или нескольких промежуточных серверов.

Формат выходных данных

Выведите два целых числа, разделенных пробелом: k — минимальное число серверов в отказоустойчивом множестве серверов, c — количество способов выбора отказоустойчивого множества из k серверов, взятое по модулю $10^9 + 7$.

Пример

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

Задача H. Минимизация мостов

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

Добавить в граф $G = \langle V, E \rangle$ (возможно несвязный, с петлями и кратными рёбрами) ровно одно ребро, так чтобы количество мостов в данном графе стало минимально возможным.

Напомним, что мостом в графе называется такое ребро, удаление которого увеличивает число компонент связности графа.

Формат входных данных

Первая строка входного файла содержит два натуральных числа n и m – количества вершин и рёбер графа соответственно ($1 \leqslant n \leqslant 200\,000$, $1 \leqslant m \leqslant 200\,000$).

Следующие m строк содержат описание рёбер по одному на строке. Ребро номер i описывается двумя натуральными числами v_i , u_i – номерами концов ребра ($1 \leqslant v_i, u_i \leqslant n$).

Формат выходных данных

Выведите наименьшее число мостов, которое можно получить добавлением ровно одного ребра.

Пример

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

Задача J. Обновление дата-центров

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мегабайт

У компании BigData Inc. есть n дата-центров, пронумерованных от 1 до n, расположенных по всему миру. В этих дата-центрах хранятся данные клиентов компании (как можно догадаться из названия — большие данные!)

Основой предлагаемых компанией BigData Inc. услуг является гарантия возможности работы с пользовательскими данными даже при условии выхода какого-либо из дата-центров компании из доступности. Подобная гарантия достигается путём использования *двойной репликации* данных. Двойная репликация — это подход, при котором любые данные хранятся в двух идентичных копиях в двух различных дата-центрах.

Про каждого из m клиентов компании известны номера двух различных дата-центров $c_{i,1}$ и $c_{i,2}$, в которых хранятся его данные.

Для поддержания работоспособности дата-центра и безопасности данных программное обеспечение каждого дата-центра требует регулярного обновления. Релизный цикл в компании BigData Inc. составляет один день, то есть новая версия программного обеспечения выкладывается на каждый компьютер дата-центра каждый день.

Обновление дата-центра, состоящего из множества компьютеров, является сложной и длительной задачей, поэтому для каждого дата-центра выделен временной интервал длиной в час, в течение которого компьютеры дата-центра обновляются и, как следствие, могут быть недоступны. Будем считать, что в сутках h часов. Таким образом, для каждого дата-центра зафиксировано целое число u_j ($0 \leqslant u_j \leqslant h-1$), обозначающее номер часа в сутках, в течение которого j-й дата-центр недоступен в связи с плановым обновлением.

Из всего вышесказанного следует, что для любого клиента должны выполняться условия $u_{c_{i,1}} \neq u_{c_{i,2}}$, так как иначе во время одновременного обновления обоих дата-центров, компания будет не в состоянии обеспечить клиенту доступ к его данным.

В связи с переводом часов в разных странах и городах мира, время обновления в некоторых дата-центрах может сдвинуться на один час вперёд. Для подготовки к непредвиденным ситуациям руководство компании хочет провести учения, в ходе которых будет выбрано некоторое непустое подмножество дата-центров, и время обновления каждого из них будет сдвинуто на один час позже внутри суток (то есть, если $u_j = h - 1$, то новым часом обновления будет 0, иначе новым часом обновления станет $u_j + 1$). При этом учения не должны нарушать гарантии доступности, то есть, после смены графика обновления должно по-прежнему выполняться условие, что данные любого клиента доступны хотя бы в одном экземпляре в любой час.

Учения — полезное мероприятие, но трудоёмкое и затратное, поэтому руководство компании обратилось к вам за помощью в определении минимального по размеру непустого подходящего подмножества дата-центров, чтобы провести учения только на этом подмножестве.

Формат входных данных

В первой строке находятся три целых числа n, m и h ($2 \leqslant n \leqslant 100\,000$, $1 \leqslant m \leqslant 100\,000$, $2 \leqslant h \leqslant 100\,000$) — число дата-центров компании, число клиентов компании и количество часов в сутках.

Во второй строке вам даны n чисел $u_1, u_2, \ldots, u_n$ ($0 \leqslant u_j < h$), j-е из которых задаёт номер часа, в который происходит плановое обновление программного обеспечения на компьютерах дата-центра j.

Далее в m строках находятся пары чисел $c_{i,1}$ и $c_{i,2}$ ($1 \leqslant c_{i,1}, c_{i,2} \leqslant n$, $c_{i,1} \neq c_{i,2}$), задающие номера дата-центров, на которых находятся данные клиента i.

Гарантируется, что при заданном расписании обновлений в дата-центрах любому клиенту в любой момент доступна хотя бы одна копия его данных.

Формат выходных данных

В первой строке выведите минимальное количество дата-центров k ($1 \leqslant k \leqslant n$), которые должны затронуть учения, чтобы не потерять гарантию доступности. Во второй строке выведите k различных целых чисел — номера кластеров $x_1, x_2, \ldots, x_k$ ($1 \leqslant x_i \leqslant n$), на которых в рамках учений обновления станут проводиться на час позже. Номера кластеров можно выводить в любом порядке.

Если возможных ответов несколько, разрешается вывести любой из них. Гарантируется, что хотя бы один ответ, удовлетворяющий условиям задачи, существует.

Примеры

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

Замечание

Рассмотрим первый тест из условия. Приведённый ответ является единственным способом провести учения, затронув только один дата-центр. В таком сценарии третий сервер начинает обновляться в первый час дня, и никакие два сервера, хранящие данные одного и того же пользователя, не обновляются в один и тот же час.

С другой стороны, например, сдвинуть только время обновления первого сервера на один час вперёд нельзя — в таком случае данные пользователей 1 и 3 будут недоступны в течение нулевого часа.

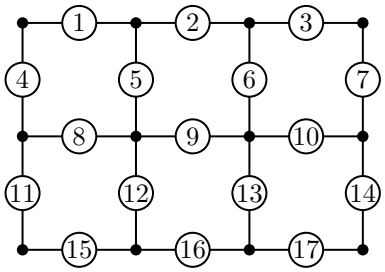
Задача K. Золотые монеты

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

Саша очень любит играть в игры на своем планшете. Недавно он скачал новую игру, которая называется «Золотые монеты».

Игра проходит на поле, имеющем вид сетки с тремя горизонтальными и четырьмя вертикальными рядами. Это поле представляет собой город: линии сетки являются дорогами, а узлы сетки — перекрестками. По дорогам можно перемещаться в обоих направлениях. В начале игры на каждой дороге расположено несколько золотых монет.

Пример игрового поля приведен на рисунке, перекрестки обозначены черными точками, дороги — отрезками, а число на дороге задает начальное количество золотых монет на ней.



Персонажем игры является бородатый электромонтер Томас. Сначала игрок может поместить Томаса на любой перекресток. Затем каждый ход игрок перемещает Томаса с перекрестка, на котором он находится, на соседний перекресток по одной из дорог, на которой еще лежит хотя бы одна монета.

Проходя по дороге, Томас забирает округленную вверх половину лежащих на ней монет. Таким образом, если на дороге лежит x монет, то, пройдя по этой дороге, Томас заберет $\lceil x/2 \rceil$ монет, а на дороге останется $\lfloor x/2 \rfloor$ монет. По дорогам, на которых монет уже нет, перемещать Томаса не разрешается.

Когда Томас оказывается в ситуации, что на всех соседних дорогах не осталось ни одной монеты, игра заканчивается. Очки игрока равны числу монет, которые собрал Томас.

Помогите Саше понять, какое максимальное количество монет он сможет собрать.

Формат входных данных

Входной файл состоит из пяти строк. Первая, третья и пятая строки содержат по три целых числа и описывают соответствующие горизонтальные улицы. Вторая и четвертая строки содержат по четыре целых числа и описывают соответствующие вертикальные улицы. Все числа неотрицательные и не превосходят 10^9.

Формат выходных данных

В единственной строке выходного файла выведите число s — максимальное количество монет, которые удастся собрать.

Примеры

стандартный ввод	стандартный вывод
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17	150
1 1 1 0 0 0 0 1 1 1 0 0 0 1 1 1 1	7

Задача L. 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