Задача A. Блинские перестановки...

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1.5 секунд
Ограничение по памяти:	256 мегабайт

Преподаватель алгоритмов Гриб провел экзамен в школе ЦПМ и всех успешно сдавших его школьников пригласил в ресторан, а остальных отправил на завод.

Всего в ресторане было n школьников. Известно, что за i-й стол сел ученик под номером b_i. Но из-за странностей людей, отлично сдавших этот экзамен, несколько раз за вечер все школьники дружно вставали и садились за другие столы, а точнее, человек с i-го стола перемещается за стол с номером p_i.

Но к концу вечера Гриб увидел совсем не то, что ожидал: за i-м столом сидел ученик под номером a_i. Поэтому после мысли «`Что за блинская перестановка`» Гриб заинтересовался, а какая же была изначальная рассадка учеников за столами. Но, так как за вечер Гриб забыл ее, он просит вас ее найти. Но, чтобы усложнить вам жизнь, злой Гриб просит вывести лексикографически минимальную перестановку b.

Формат входных данных

Первая строка содержит одно целое число n ($1 \leqslant n \leqslant 500\,000$) — количество учеников Гриба.

Вторая строка содержит n целых чисел: $p_1, p_2, \ldots, p_n$ ($1 \leqslant p_i \leqslant n$) — за какой стол пересаживается школьник из-за i-го стола каждый раз. Гарантируется, что числа p_i образуют перестановку.

Третья строка содержит n целых чисел: $a_1, a_2, \ldots, a_n$, ($1 \leqslant a_i \leqslant n$) — итоговая блинская рассадка учеников. Гарантируется, что числа a_i образуют перестановку.

Формат выходных данных

В единственной строке выведите n целых чисел — лексикографически минимальную перестановку, из которой могло получиться конечное расположение учеников.

Примеры

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

Замечание

В первом примере, если с самого начала была рассадка $[2, 3, 4, 1]$, то после первой итерации рассадка будет $[2, 3, 4, 1]$. И не сложно показать, что эта перестановка лексикографически минимальная, среди подходящих.

Во втором примере, если сначала была рассадка $[1, 2, 4, 3]$, то после первой итерации будет рассадка $[3, 2, 1, 4]$, а после второй — $[4, 2, 3, 1]$.

Задача B. Очередная задача про запросы на дереве

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

Как-то раз Никита гулял по Липецку и обнаружил в Нижнем парке примечательное *дерево* на n вершинах, пронумерованных целыми числами от 1 до n. Дерево подвешено за вершину с номером 1. Напомним, что деревом называется связный неориентированный граф без циклов.

В каждой вершине дерева v записаны два целых числа t_v и d_v. Также изначально в некоторых вершинах дерева лежат фишки.

Никита дал вам q запросов, каждый из которых характеризуется целым числом c. Для выполнения запроса вы должны рассмотреть все вершины, в которых лежит фишка, у которых $t_v = c$. Пусть эти вершины имеют номера $v_1, v_2, \ldots, v_k$. После этого для каждой вершины **одновременно** независимо выполняется процесс, описанный ниже.

Рассмотрим вершину v_i и путь от данной вершины до корня дерева (то есть до вершины с номером 1). Пусть этот путь состоит из вершин $u_0, u_1, \ldots, u_m$, где $u_0 = v_i$, $u_m = 1$, а вершина u_i является предком вершины u_{i-1}. Заберем фишку из вершины v_i и переместим ее в вершину с номером $u_{d_{v_i}}$. Если $d_{v_i} > m$, то фишка перемещается в корень дерева. Для лучшего понимания процесса обратите внимание на примеры ниже.

Вы должны выполнять запросы последовательно и найти минимальный номер запроса, после выполнения которого все фишки окажутся в корне дерева, либо определить, что после выполнения всех запросов Никиты не все фишки окажутся в корне дерева.

Формат входных данных

Первая строка содержит два целых числа n и q ($2 \leqslant n \leqslant 200\,000$, $1 \leqslant q \leqslant 200\,000$) — количество вершин в дереве и запросов Никиты, соответственно.

Вторая строка содержит $n - 1$ целых чисел $p_2, p_2, \ldots, p_n$ ($1 \leqslant p_i < i$). Данные числа означают, что предком вершины i в дереве является вершина p_i.

Третья строка содержит n целых чисел t_i ($1 \leqslant t_i \leqslant n$).

Четвертая строка содержит n целых чисел d_i ($1 \leqslant d_i \leqslant n$).

Пятая строка содержит n целых чисел a_i ($a_i \in \{0, 1\}$). Если $a_i = 1$, то в i-й вершине дерева изначально находится фишка. Гарантируется, что изначально в дереве существует хотя бы одна фишка, которая находится не в корне.

Шестая строка содержит q целых чисел c_i ($1 \leqslant c_i \leqslant n$) — параметры запросов.

Формат выходных данных

Если существует минимальный номер запроса i ($1 \leqslant i \leqslant q$), после выполнения которого все фишки окажутся в корне дерева, выведите число i.

В противном случае выведите число -1.

Примеры

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

Задача C. Невероятные приключения ДжоДжо

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

Ты думал, что тут будет легенда про ДжоДжо? Но нет, это был я, Дио!

Дана бинарная строка s длины n, состоящая из символов `0` и `1`. Построим **квадратную** таблицу размера $n \times n$, состоящую из символов `0` и `1` следующим образом.

В первую строку таблицы запишем исходную строку s. Во вторую строку таблицы запишем *циклический сдвиг* строки s на один вправо. В третью строку таблицы запишем циклический сдвиг строки s на два вправо. И так далее. Таким образом, в строке с номером k будет записан циклический сдвиг строки s на k вправо при **нумерации строк с нуля**.

В получившейся таблице требуется найти *прямоугольник*, состоящий только из единиц, имеющий наибольшую площадь.

Прямоугольником назовем множество всех клеток таблицы (i, j) таких, что $x_1 \leqslant i \leqslant x_2$ и $y_1 \leqslant j \leqslant y_2$ для некоторых целых чисел $0 \leqslant x_1 \leqslant x_2 < n$ и $0 \leqslant y_1 \leqslant y_2 < n$.

Напомним, что циклическим сдвигом строки s на k вправо называется строка $s_{n-k+1} \dots s_n s_1 s_2 \dots s_{n-k}$. Например, циклическим сдвигом строки «`01011`» на 0 вправо является сама строка «`01011`», а ее циклическим сдвигом на 3 вправо является строка «`01101`».

Формат входных данных

Каждый тест состоит из нескольких наборов входных данных. Первая строка содержит одно целое число t ($1 \leqslant t \leqslant 100\,000$) — количество наборов входных данных. Далее следует описание наборов входных данных.

Каждый набор входных данных описывается единственной бинарной строкой s ($1 \leqslant |s| \leqslant 2 \cdot 10^6$), состоящей из символов `0` и `1`.

Гарантируется, что суммарная длина строк по всем наборам входных данных не превосходит $2 \cdot 10^6$.

Формат выходных данных

Для каждого набора входных данных выведите одно целое число — максимальную площадь прямоугольника, состоящего только из единиц.

Пример

стандартный ввод	стандартный вывод
5 0 1 101 011110 101010	0 1 2 6 1

Замечание

В первом наборе входных данных получается таблица 1×1, состоящая из единственного символа «`0`», таким образом нет прямоугольников, состоящих из единиц, и ответ 0.

Во втором наборе входных данных получается таблица 1×1, состоящая из единственного символа «`1`».

Таблицы, которые получаются в третьем, четвертом и пятом наборах входных данных изображены ниже, прямоугольники с максимальной площадью выделены жирным шрифтом:

1	0	1
1	1	0
0	1	1

0	1	1	1	1	0
0	0	1	**1**	**1**	**1**
1	0	0	**1**	**1**	**1**
1	1	0	0	1	1
1	1	1	0	0	1
1	1	1	1	0	0

1	0	1	0	1	0
0	1	0	1	0	1
1	0	1	0	1	0
0	1	0	**1**	0	1
1	0	1	0	1	0
0	1	0	1	0	1

Задача D. Тяп-ляп, фигак, в релиз!

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	3 секунды
Ограничение по памяти:	1024 мегабайта

Вы долго об этом мечтали и наконец устроились работать в крупную IT-компанию. Вы долго-долго учились всем существующим современным технологиям и уже готовы применить все свои знания на практике. Но тут вы садитесь за свой рабочий стол и видите лист с распечатанным крупными буквами девизом компании: `abcdabcdabcdabcd...`.

В девизе компании указаны четыре главных принципа — `a` (Тяп), `b` (Ляп), `c` (Фигак) и `d` (Релиз). Поэтому вы рассматриваете строки длины n, состоящие из указанных четырех букв латинского алфавита. **Неупорядоченные** пары букв «`ab`», «`bc`», «`cd`» и «`da`» в этом девизе соседние, поэтому будем называть такие пары символов *хорошими*. Итак, если вам дана строка s длины n и известно, что неупорядоченная пара символов $\{x, y\}$ хорошая, то со строкой можно совершить одну из указанных операций:

- если $s_n = x$, то разрешается заменить этот символ на y,
- если существует такое $1 \leqslant i < n$, что $s_i = x$ и $s_{i+1} = \ldots = s_n = y$, то разрешается заменить i-й символ строки на y, а все последующие — на x.

Например, строку `bacdd` можно заменить на одну из строк `bacda`, `bacdc` или `badcc`, а строку `aac` можно заменить на `aab` или `aad`.

Непустую последовательность операций для строки s будем называть *корректной*, если выполняются следующие два условия:

1. после совершения всех операций снова получится строка s,
2. никакая строка, кроме s, в ходе совершения операций не возникнет более одного раза. При этом строка s должна возникнуть ровно два раза — перед началом выполнения операций и после выполнения всех операций.

Теперь мы готовы перейти к условию задачи! У вас есть множество строк, которое изначально пусто. Далее q раз в множество добавится очередная строка t_i, либо строка t_i удалится из множества. После каждого запроса требуется вывести минимальный и максимальный размер корректной последовательности операций, в ходе которой каждое слово возникает хотя бы один раз. Выбор начальной строки s остается за вами.

Формат входных данных

Первая строка содержит два целых числа n и q ($1 \leqslant n \leqslant 20$, $1 \leqslant q \leqslant 100\,000$) — длина рассматриваемых строк и количество запросов изменения множества строк.

Каждая из следующих q строк содержит строку t_i ($|t_i| = n$). Все строки состоят из символов «`a`», «`b`», «`c`» и «`d`». Если до этого запроса строка t_i не находилась в множестве, то она добавляется в него, а иначе она удаляется из множества.

Формат выходных данных

Для каждого из q запросов выведите два целых числа — минимальный и максимальный размер корректной последовательности операций, в ходе которой каждое слово из множества возникает хотя бы один раз.

Если не существует ни одной последовательности операций, удовлетворяющей условию задачи, выведите одно число -1.

Примеры

стандартный ввод	стандартный вывод
2 4 aa ac dd ac	2 12 4 4 -1 12 12
3 2 acc bdd	2 44 28 44

Замечание

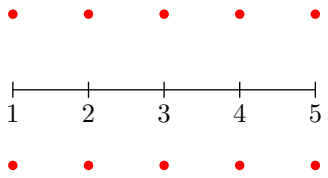
Рассмотрим первый пример.

- После первого запроса множество строк равно {aa}, минимальная последовательность действий имеет следующий вид: aa, ab, aa. В качестве максимальной последовательности действий подходит aa, ab, ba, bb, bc, cb, cc, cd, dc, dd, da, ad, aa.
- После второго запроса множество строк равно {aa, ac}. Минимальная и максимальная последовательности действий: aa, ab, ac, ad, aa.
- После третьего запроса множество строк равно {aa, ac, dd}. Не существует последовательности действий, которая подходит под условие, поэтому следует вывести -1.
- После четвертого запроса множество строк равно {aa, dd}. Минимальная и максимальная последовательности действий таковы: aa, ab, ba, bb, bc, cb, cc, cd, dc, dd, da, ad, aa.

Задача E. Реклама

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

В замечательном городе Ельце есть только одна дорога. По обе стороны от нее стоят по n кофеен. А именно, если представить улицу как отрезок с координатами начала $(1, 0)$ и конца $(n, 0)$, то с одной стороны от нее кофейни располагаются в точках с координатами $(1, 1), (2, 1), \ldots, (n, 1)$, а с другой стороны — в точках с координатами $(1, -1), (2, -1), \ldots, (n, -1)$.



Пример расположения кофеен при $n = 5$

Для каждой кофейни известно, сколько человек ее посещают каждый день. Для кофейни с координатами $(i, 1)$ это количество равно a_i, а для кофейни с координатами $(i, -1)$ это количество равно b_i, где $1 \leqslant i \leqslant n$.

Вова решил открыть новый чайный магазинчик в другом месте, но для этого ему нужно привлечь клиентов. Для этого он решил повесить рекламные плакаты на окна некоторых кофеен. Если плакат весит на окне какой-то кофейни, то его видно из всех кофеен с **другой** стороны улицы, у которых x-координата отличается не более, чем на d, где d — фиксированное число. Но при этом, если на окне какой-то кофейни размещен плакат, то из нее ничего не видно, и ее посетители ничего не узнают о кофейне Вовы.

Более формально, если рекламный плакат весит на окне кофейни с координатами (x, y), то его видно из всех кофеен с координатами $(x', -y)$, для которых $|x - x'| \leqslant d$, на окнах которых плакат не весит.

Помогите Вове определить, какое максимальное количество людей из всех кофеен смогут увидеть его плакаты, если у него есть неограниченное количество плакатов и он развесит их оптимально.

Обратите внимание, что каждый человек, который видит плакат, учитывается в ответе **ровно один раз**. Другими словами, не важно, сколько плакатов видит человек, если он видит хотя бы один из них.

Формат входных данных

Первая строка содержит два целых числа n и d ($0 \leqslant d < n \leqslant 1\,500$) — количество кофеен с каждой стороны улицы и максимальная разность x-координат, при которых видно плакат.

Вторая строка содержит n целых чисел $a_1, a_2, \ldots, a_n$ ($1 \leqslant a_i \leqslant 100\,000$), где i-е число обозначает количество посетителей в кофейне с координатами $(i, 1)$.

Третья строка содержит n целых чисел $b_1, b_2, \ldots, b_n$ ($1 \leqslant b_i \leqslant 100\,000$), где i-е число обозначает количество посетителей в кофейне с координатами $(i, -1)$.

Формат выходных данных

Выведите одно целое число — максимальное количество людей, которые могут увидеть плакаты.

Примеры

стандартный ввод	стандартный вывод
5 2 1 2 2 4 3 5 1 3 2 2	20
4 1 100 1 100 1 1 100 1 1	302
3 0 1 2 3 3 2 1	8
4 3 1 1 1 1 1 1 1 1	6

Замечание

В первом примере Вова может разместить рекламу на кофейнях с координатами $(3, 1)$ и $(3, -1)$, которую будет видно во всех оставшихся кофейнях, так как $d = 2$. Тогда всего плакаты увидят $1 + 2 + 4 + 3 + 5 + 1 + 2 + 2 = 20$ человек.

Во втором примере Вова может разместить рекламу на кофейнях с координатами $(1, 2)$, $(1, -1)$ и $(4, -1)$. Тогда, так как $|1 - 1| \leqslant 1$, $|3 - 4| \leqslant 1$ и $|4 - 4| \leqslant 1$, посетителям в кофейнях с координатами $(1, 1)$, $(1, 3)$ и $(1, 4)$ видно рекламу. А так как $|2 - 2| \leqslant 1$ и $|3 - 2| \leqslant 1$, то посетителям в кофейнях $(2, -1)$ и $(3, -1)$ видно рекламу. Тогда всего $100 + 100 + 1 + 100 + 1 = 302$ человека видят рекламу.

В третьем примере Вова может разместить рекламу на кофейнях с координатами $(1, 1)$, $(1, 2)$ и $(3, -1)$, в которых есть 1, 2 и 1 посетитель соответственно. Так как $d = 0$, то рекламу видно только из кофейни напротив, поэтому всего плакаты увидят $3 + 2 + 3 = 8$ человек.

В четвертом примере Вова может разместить рекламу на одной кофейне с каждой стороны улицы. Тогда реклама будет видна из трех оставшихся кофеен с другой стороны улицы, потому что $d = 3$. А тогда всего рекламу увидят 6 человек.

Задача F. Лисица и полный обход древа

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

Лисица Яэ забралась на *древо* Священной сакуры. Древом называют связный неориентированный граф, не содержащий циклов.

Для передвижения по древу лисица применяет свои магические способности. За один прыжок Яэ может перепрыгнуть из вершины v в вершину u, если вершины v и u соединены ребром, либо если существует такая вершина w, что вершины v и w соединены ребром, а также вершины u и w соединены ребром.

После того, как Яэ смогла заполучить лепесток сакуры, она задумалась, существует ли **циклический** маршрут в древе $v_1, v_2, \ldots, v_n$, такой что:

- между вершинами v_i и v_{i+1} лисица может совершить прыжок,
- между вершинами v_n и v_1 лисица может совершить прыжок,
- все v_i попарно различны.

Помогите лисице определить, существует ли требуемый обход.

Формат входных данных

Первая строка содержит одно целое число n ($2 \leqslant n \leqslant 200\,000$) — количество вершин древа.

Каждая из следующих $n - 1$ строк содержит два целых числа u и v ($1 \leqslant u, v \leqslant n$, $u \neq v$) — вершины, соединенные ребром. Гарантируется, что данные ребра задают древо.

Формат выходных данных

В первой строке выведите «`Yes`» (без кавычек), если требуемый обход древа существует, либо «`No`» (без кавычек) в противном случае.

Если требуемый обход древа существует, во второй строке выведите n различных целых чисел $v_1, v_2, \ldots, v_n$ ($1 \leqslant v_i \leqslant n$) — вершины древа в порядке обхода.

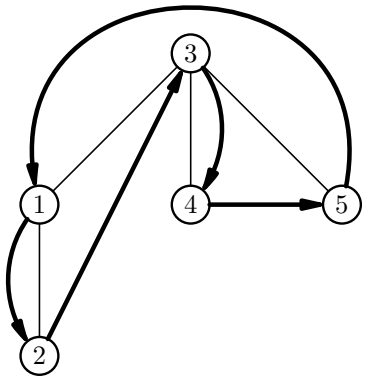
Если существует несколько корректных обходов, выведите любой из них.

Примеры

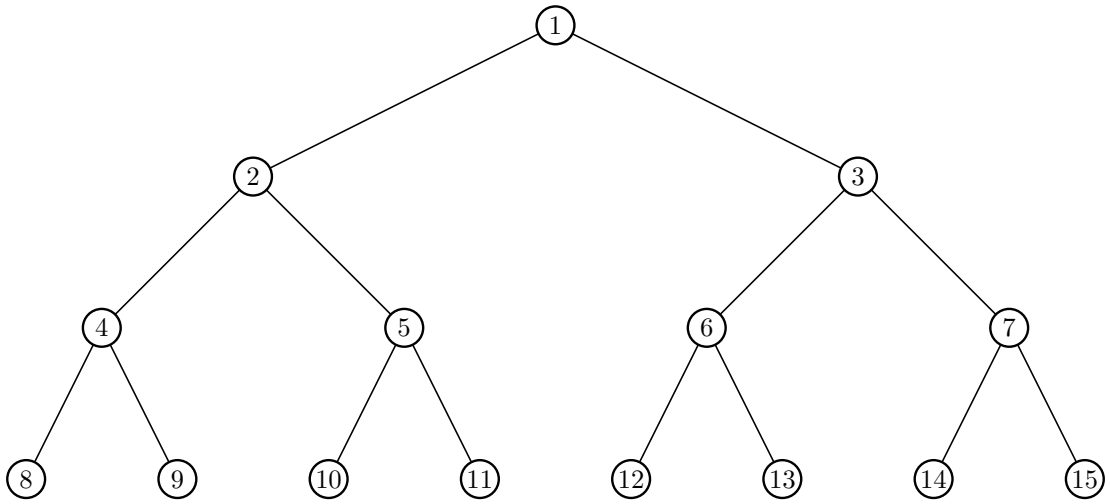
стандартный ввод	стандартный вывод
5 1 2 1 3 3 4 3 5	Yes 4 5 1 2 3
3 1 2 1 3	Yes 1 2 3
15 1 2 1 3 2 4 2 5 3 6 3 7 4 8 4 9 5 10 5 11 6 12 6 13 7 14 7 15	No

Замечание

Древо из первого примера изображено ниже. Жирными стрелками обозначен маршрут лисицы.



Древо из третьего примера изображено ниже. Можно показать, что для него не существует требуемого маршрута.


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

Задача G. Конструктивная задача

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

Как известно, любая задача, в которой не требуется использовать сложные структуры данных, считается конструктивной. Вам предлагается решить одну из таких задач.

Дан массив a из n целых неотрицательных чисел. Вам разрешается **ровно один раз** выполнить следующую операцию: выбрать какой-то непустой подотрезок $a_l, a_{l+1}, \ldots, a_r$ массива a и целое неотрицательное число k, и присвоить значение k всем элементам массива на выбранном подотрезке.

Требуется выяснить, можно ли увеличить $\operatorname{mex}(a)$ ровно на единицу, проделав такую операцию. Другими словами, если до выполнения операции выполнялось $\operatorname{mex}(a) = m$, то после ее выполнения должно быть верно, что $\operatorname{mex}(a) = m + 1$.

Напомним, что mex набора целых чисел $c_1, c_2, \ldots, c_k$ определяется как наименьшее неотрицательное целое число x, которое не встречается в наборе чисел c.

Формат входных данных

Каждый тест состоит из нескольких наборов входных данных. Первая строка содержит одно целое число t ($1 \leqslant t \leqslant 50\,000$) — количество наборов входных данных. Далее следует описание наборов входных данных.

Первая строка каждого набора входных данных содержит единственное целое число n ($1 \leqslant n \leqslant 200\,000$) — размер массива a.

Вторая строка каждого набора входных данных содержит n целых неотрицательных чисел $a_1, a_2, \ldots, a_n$ ($0 \leqslant a_i \leqslant 10^9$) — элементы массива a.

Гарантируется, что сумма n по всем наборам входных данных не превосходит 200 000.

Формат выходных данных

Для каждого набора входных данных выведите «`Yes`» (без кавычек), если можно увеличить $\operatorname{mex}(a)$ ровно на единицу, выполнив операцию из условия ровно один раз, иначе выведите «`No`» (без кавычек).

Пример

стандартный ввод	стандартный вывод
4 3 1 2 1 4 0 2 2 0 4 3 2 0 2 1 0	Yes Yes No No

Замечание

В первом наборе входных данных $\operatorname{mex}(a) = 0$. Если присвоить всем элементам a значение 0, то mex полученного массива будет равен 1, и тем самым увеличится на единицу.

Во втором наборе входных данных $\operatorname{mex}(a) = 1$. Если присвоить значение 1 элементам a на отрезке от 2 до 3, то получится массив $[0, 1, 1, 0]$, для которого mex равен 2, и тем самым увеличился на единицу по сравнению с изначальным.

Можно показать, что в третьем и четвертом наборах входных данных невозможно выполнить операцию, чтобы значение $\operatorname{mex}(a)$ увеличилось ровно на единицу.

Задача H. Дороги в Ельце

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

Елец, Елец, не город, а ...

Фольклор

Это интерактивная задача.

Как хорошо известно, в городе Елец за его полуторатысячелетнюю историю ни разу не ремонтировали дороги. И только недавно руководство города отремонтировало некоторые из них.

Известно, что всего в Ельце есть n перекрестков и m дорог, перемещаться по которым можно в обе стороны, пронумерованных целыми числами от 1 до m. i-я дорога соединяет перекрестки с номерами a_i и b_i.

Среди всех m дорог было отремонтировано некоторое подмножество дорог, но вам не известно, какое именно. Единственная информация, которую вы смогли получить от дорожных служб города, это то, что от любого перекрестка можно доехать до любого другого, двигаясь только по отремонтированным дорогам.

Вы — молодой предприниматель, решили организовать службу доставки свежего сырого мяса в Ельце (в самом городе такое мясо называют «стейками», оно пользуется большой популярностью у местных жителей). Вы уже набрали штат курьеров, однако курьеры готовы перемещаться только по отремонтированным дорогам. Теперь вам предстоит выяснить, какие дороги уже отремонтированы.

Городская администрация предоставила вам город на некоторое время, поэтому вы можете делать действия одного из трех типов:

1. Заблокировать дорогу с номером x. В этом случае перемещение по дороге для курьеров будет запрещено. Исходно все дороги разблокированы.

2. Разблокировать дорогу с номером x. В этом случае курьеры смогут двигаться по дороге x, если она отремонтирована.

3. Попробовать доставить заказ на перекресток с номером y. В этом случае один из ваших курьеров начнет двигаться от неизвестного вам перекрестка s и доставит заказ на перекресток с номером y в том случае, если существует путь по разблокированным отремонтированным дорогам от перекрестка s до перекрестка y. При этом гарантируется, что перекресток s будет выбран заранее.

К сожалению, город предоставлен в ваше полное распоряжение не надолго, поэтому вы можете сделать не более $100 \cdot m$ запросов.

Протокол взаимодействия

Ваша программа будет взаимодействовать с программой жюри с использованием стандартных потоков ввода и вывода. Каждое взаимодействие будет состоять из решения задачи для нескольких наборов входных данных.

Сначала ваша программа должна считать целое число t ($1 \leqslant t \leqslant 1\,000$) — количество наборов входных данных в рамках одного взаимодействия с программой жюри. Затем t раз необходимо выполнить взаимодействие по решению задачи для набора входных данных.

Рассмотрим протокол взаимодействия для одного набора входных данных.

Сначала ваша программа должна считать данные в следующем формате.

Первая строка содержит два целых числа n и m ($2 \leqslant n \leqslant 2\,000$, $n - 1 \leqslant m \leqslant 2\,000$) — число перекрестков и дорог в Ельце.

Каждая из следующих m строк содержит описание одной дороги. i-я из этих строк содержит два целых числа a_i и b_i ($1 \leqslant a_i, b_i \leqslant n$) — номера перекрестков, которые соединяет i-я дорога. Гарантируется, что никакая дорога не соединяет перекресток сам с собой. При этом возможно, что между парой различных перекрестков есть несколько дорог.

После того, как вы считали описание набора входных данных, вы можете задавать запросы. Запросы могут быть трех типов:

1. «`- x`» ($1 \leqslant x \leqslant m$). В этом случае дорога с номером x блокируется, если она ещё не была заблокирована.

2. «`+ x`» ($1 \leqslant x \leqslant m$). В этом случае дорога с номером x разблокируется. **Обратите внимание**, что дорога x должна быть заблокирована ранее. Исходно все дороги разблокированы.

3. «`? y`» ($1 \leqslant y \leqslant n$). В этом случае программа жюри выбирает некоторый город s. В случае, если от города s до города y можно добраться по разблокированным отремонтированным дорогам, программа жюри выведет 1, иначе программа жюри выведет 0. **Обратите внимание**, что город s будет выбран до получения информации о городе y, однако при выборе города s могут учитываться ваши предыдущие запросы.

Всего вы можете задать не более $100 \cdot m$ запросов для каждого набора входных данных.

После того, как вы нашли все отремонтированные дороги, выведите «`! c1 c2 c3 ... cm`», где c_i равно 1, если дорога i отремонтирована, и 0, если дорога не отремонтирована. Этот вывод не будет считаться в общем числе запросов. На это программа жюри выведет 1, если ваш ответ является правильным, и 0 в противном случае. В случае, если ответ оказался не правильным, ваша программа должна немедленно завершить работу.

Обратите внимание, что вам не обязательно разблокировать все дороги на момент вывода ответа. Гарантируется, что все отремонтированные дороги зафиксированы изначально и не будут меняться программой жюри в зависимости от запросов.

Гарантируется, что суммарно по всем наборам входных данных в одном тесте, сумма n и сумма m не превосходят 2 000.

Если вы используете «`cout << ... << endl`» в C++, «`System.out.println`» в Java, «`print`» в Python, «`writeln`» в Pascal, то сброс потока вывода происходит автоматически, дополнительно ничего делать не требуется.

Если вы используете другой способ вывода, рекомендуется делать сброс буфера потока вывода. Обратите внимание, что перевод строки надо выводить в любом случае. Для сброса буфера потока вывода можно использовать «`fflush(stdout)`» в C++, «`flush(output)`» в Pascal, «`System.out.flush()`» в Java, «`sys.stdout.flush()`» в Python.

Пример

стандартный ввод	стандартный вывод
2 2 2 1 2 2 1	
	- 1 ? 1
1	
	? 2
0	
	- 2 + 1 ? 1
1	
	! 1 0
1 3 3 1 2 2 3 3 1	
	- 1 ? 2
1	
	? 1
1	
	- 2 ? 3
1	
	? 3
0	
	+ 1 ? 3
1	
	? 2
1	
	? 1
1	
	! 1 1 1
1	

Замечание

В первом наборе входных данных дорога 1 отремонтирована, а дорога 2 — нет. Для первого запроса доставки в качестве s был выбран перекресток 1, поэтому путь от перекрестка 1 до 1 есть. Для второго запроса доставки в качестве s был выбран перекресток 1. Так как в городе заблокирована единственная отремонтированная дорога, то пути между перекрестками 1 и 2 нет. Для третьего запроса доставки в качестве s был выбран перекресток 2, путь между перекрестками 2 и 1 есть по дороге 1, которая отремонтирована и разблокирована.

Во втором наборе входных данных для запросов доставки в качестве стартовых перекрестков были выбраны перекрестки 1, 3, 1, 2, 2, 3, 1.

Задача I. Досмотр перед вылетом

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

Егор решил выбраться из своей страны в путешествие по другим городам нашей необъятной планеты. Для перемещения он использует самолеты. К сожалению, по какой-то причине каждый раз ему приходится проходить полный досмотр багажа. Обычно все проходило спокойно, но в какой-то момент работники аэропорта обнаружили у него в чемодане задачу, которую он придумал, но не смог решить. Теперь из-за этой задачи его могут не выпустить из страны, поэтому вы должны помочь ему решить ее до вылета!

В вещах Егора была найдена следующая задача. Дан массив, состоящий из n целых чисел. Разрешается не более одного раза развернуть любой отрезок массива от l до r. После разворота отрезка от l до r массив $a_1, \ldots, a_n$ превращается в массив $a_1, \ldots, a_{l-1}, a_r, a_{r-1}, \ldots, a_{l+1}, a_l, a_{r+1}, \ldots, a_n$.

Ваша задача состоит в том, чтобы после **не более одного** применения описанной выше операции максимизировать значение выражения $\max\limits_{1 \leqslant i \leqslant j \leqslant n} (a_i + \ldots + a_j)$. Иными словами, необходимо максимизировать значение подотрезка с максимальной суммой.

Формат входных данных

Первая строка содержит одно целое число n ($1 \leqslant n \leqslant 10^6$) — количество элементов в массиве.

Вторая строка содержит n целых чисел $a_1, a_2, \ldots, a_n$ ($-10^9 \leqslant a_i \leqslant 10^9$) — элементы массива.

Формат выходных данных

В единственной строке выведите максимальное значение среди сумм всех подотрезков массива после не более одного применения описанной операции.

Примеры

стандартный ввод	стандартный вывод
3 1 -100 2	3
6 1 2 3 -4 5 -6	11

Замечание

Рассмотрим первый пример. Можно развернуть отрезок с первого до второго элемента и получить массив $[-100, 1, 2]$. В нем подотрезок с максимальной суммой имеет сумму 3. Эта сумма является максимально возможной.

Рассмотрим второй пример. Если развернуть отрезок с первого до четвертого элемента, то получится массив $[-4, 3, 2, 1, 5, -6]$. В нем отрезок с максимальной суммой имеет сумму 11. Можно показать, что невозможно получить сумму больше.

Задача J. Мясник

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

Антон играет в свою любимую игру «Защита древних 2» за своего любимого героя — Мясника. Сейчас он хочет приготовить себе ужин. Для этого он возьмет прямоугольник высоты h и ширины w, затем проведет вертикальный или горизонтальный разрез так, чтобы обе получившиеся части имели целочисленные длины сторон. После этого одну из частей он положит в ящик, а другую снова разрежет, и так далее.

Более формально, прямоугольник размера $h \times w$ можно разрезать на две части с размерами $x \times w$ и $(h-x) \times w$, где x является целым числом от 1 до $(h-1)$, или на две части с размерами $h \times y$ и $h \times (w-y)$, где y является целым числом от 1 до $(w-1)$.

Эту операцию он повторит $n-1$ раз, после чего оставшийся прямоугольник тоже положит в ящик. Таким образом, в ящике будут лежать n прямоугольников, из которых $n-1$ прямоугольник был отложен в результате разрезов, а n-й прямоугольник — тот, что остался в руках Мясника после всех $n-1$ разрезов.

К сожалению, Мясник забыл числа h и w, однако у него сохранились n прямоугольников, перемешанных в произвольном порядке. **Обратите внимание**, что Мясник **не поворачивал прямоугольники**, но, возможно, переупорядочил их. Теперь он хочет узнать все возможные пары (h, w), из которых можно получить данный набор прямоугольников. А вы должны ему в этом помочь!

Гарантируется, что существует хотя бы одна пара (h, w), из которой можно получить данный набор прямоугольников.

Формат входных данных

Первая строка содержит одно целое число n ($1 \leqslant n \leqslant 200\,000$) — количество полученных прямоугольников.

Каждая из следующих n строк содержит два целых числа a_i и b_i ($1 \leqslant a_i, b_i \leqslant 10^6$) — высота и ширина i-го прямоугольника.

Формат выходных данных

В первой строке выведите одно целое число m — количество пар (h, w), обозначающих размеры прямоугольников, из которых можно получить данные прямоугольники. Два прямоугольника считаются различными, если у них отличается высота или ширина.

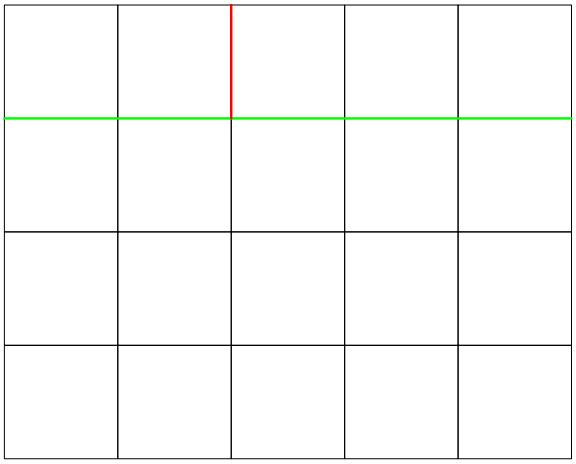
В каждой из следующих m строк выведите два целых числа h_i и w_i — высоту и ширину прямоугольника, из которого можно получить данные прямоугольники.

Примеры

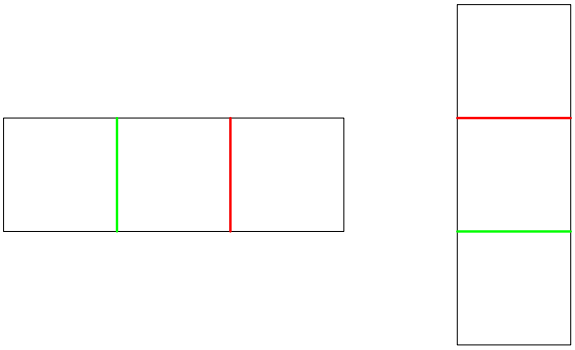
стандартный ввод	стандартный вывод
3 1 2 3 5 1 3	1 4 5
3 1 1 1 1 1 1	2 1 3 3 1

Замечание

В первом примере у Мясника изначально мог быть прямоугольник размером 4×5. Тогда разрезы могли бы выглядеть следующим образом (сначала был сделан зеленый разрез, затем красный):



Во втором примере у Мясника мог быть либо прямоугольник размера 1×3, либо 3×1. Разрезы выглядели бы так (сначала был сделан зеленый разрез, затем красный):



Задача K. Миша и яблоки

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

Школьник Миша устал заниматься спортивным программированием, и поэтому решил все бросить и уйти в магический лес торговать магическими яблоками.

Его друг Даня пришел в этот магический лес, чтобы навестить Мишу. Какого же было его удивление, когда он узнал, что Миша нашел там много друзей, таких же бывших спортивных программистов. И у всех них, как и у Миши, есть своя лавка, где они продают магические яблоки. Чтобы поддержать друзей, так кардинально поменявших свою жизнь, он решил скупить у них весь ассортимент фруктов.

Процесс покупки устроен следующим образом: всего есть n лавок, пронумерованных целыми числами от 1 до n, и m видов магических яблок, пронумерованных целыми числами от 1 до m. В каждой лавке продается какое-то множество видов яблок. Даня посещает все лавки в порядке возрастания номера, начиная с первой. Заходя в лавку, он покупает по одному магическому яблоку каждого вида, который продается в этой лавке, и кладет его к себе в рюкзак.

Однако, магические яблоки не были бы магическими, если бы с ними все было в порядке. Дело в том, что когда два яблока одного типа оказываются вместе в рюкзаке, все яблоки в нем магическим образом исчезают. **Обратите внимание**, что исчезновение происходит уже после того, как Даня положил яблоки в рюкзак и покинул лавку.

Вернувшись домой, Даня понял, что где-то в лесу он успел потерять свой рюкзак. Помня для некоторых лавок, какие виды магических яблок в них продавались, он хочет узнать, какое максимальное количество яблок могло оказаться у него в рюкзаке после всех покупок в лучшем случае.

Формат входных данных

Каждый тест состоит из нескольких наборов входных данных. Первая строка содержит одно целое число t ($1 \leqslant t \leqslant 200\,000$) — количество наборов входных данных. Далее следует описание наборов входных данных.

Первая строка каждого набора данных содержит два целых числа n и m ($1 \leqslant n, m \leqslant 200\,000$) — количество лавок и видов яблок.

Каждая из следующих n строк описывает ассортимент очередного прилавка в формате, описанном ниже.

Каждая строка начинается с целого числа k_i ($0 \leqslant k_i \leqslant 200\,000$). За ней следуют k_i различных целых чисел a_{ij} ($1 \leqslant a_{ij} \leqslant m$) — виды яблок, продаваемых в i-й лавке. Если $k_i = 0$, то Даня не помнит какой ассортимент был в этой лавке, и множество видов яблок может быть каким угодно (**в том числе и пустым**).

Гарантируется, по что сумма всех k_i по всем наборам входных данных не превосходит 200 000 и сумма n по всем наборам входных данных не превосходит 200 000.

Формат выходных данных

Для каждого набора данных выведите одно целое число — максимальное количество яблок, которое могло оказаться в рюкзаке Дани после посещения всех лавок в лучшем случае.

Пример

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

Замечание

В первом наборе входных данных Даня помнит про все лавки, поэтому процесс покупки определяется однозначно. В первой лавке он возьмет два яблока, во второй лавке — тоже два яблока. После того, как он положит их в рюкзак, все яблоки исчезнут. Поэтому в конце останется только два яблока, которые он возьмет в третьей лавке.

Во втором наборе входных данных, если в третьей лавке будет пусто, то после посещения четвертой лавки все яблоки исчезнут. В любом другом случае яблоки исчезнут после посещения третьей лавки, и в четвертой лавке Даня сможет купить одно яблоко, поэтому ответ равен единице.

В третьем наборе входных данных в первой лавке могут продаваться все виды яблок, а во второй лавке может ничего не продаваться. Тогда в конце останутся все пять яблок.

Задача L. Новое имя Юры

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

После проведения ЛКОШП Юра очень устал и захотел изменить свою жизнь и переехать в Японию. В честь такой перемены Юра решил сменить свое имя на что-то милое.

Загоревшись этой идеей, он уже придумал себе имя s, состоящее из символов «_» и «^». Но вот незадача: Юра очень любит смайлики «^_^» и «^^». Поэтому любой символ имени должен быть частью хотя бы одного такого смайлика. **Обратите внимание**, что смайликом могут являться только **подряд идущие** символы имени.

Более формально, рассмотрим все вхождения строк «^_^» и «^^» в строку s. Тогда все такие вхождения должны покрывать строку s целиком, возможно, с пересечениями. Например, в строке «^^__^_^^__^» символы на позициях 3, 4, 9, 10 и 11 не содержатся внутри ни одного смайлика, а остальные символы на позициях 1, 2, 5, 6, 7 и 8 содержатся внутри смайликов.

За одну операцию Юра может вставить один из символов «_» и «^» в свое имя s (вставить символ можно на любую позицию в строке). Он попросил вас вычислить минимальное количество операций, которые нужно сделать, чтобы имя подходило под критерий Юры.

Формат входных данных

Каждый тест состоит из нескольких наборов входных данных. Первая строка содержит одно целое число t ($1 \leqslant t \leqslant 100$) — количество наборов входных данных. Далее следует описание наборов входных данных.

Единственная строка каждого набора входных данных содержит строку s ($1 \leqslant |s| \leqslant 100$), состоящую из символов «_» и «^», — имя, придуманное Юрой.

Формат выходных данных

Для каждого набора входных данных выведите ровно одно целое число — минимальное количество символов, которые нужно добавить в имя, чтобы оно подходило для Юры. Если имя менять не нужно, выведите число 0.

Пример

стандартный ввод	стандартный вывод
7 ^______^ ___^_^^^_^___^ ^_ ^ ^_^^^^^_^_^^ ___^^ -	5 5 1 1 0 3 2

Замечание

В первом наборе входных данных можно получить следующее имя, добавив шесть символов: ^_^_^_^_^_^_^.

В третьем наборе входных данных можно добавить один символ «^» в конец имени, тогда получится имя: ^_^.

В четвертом наборе входных данных можно добавить один символ «^» в конец имени, тогда получится имя ^^.

В пятом наборе входных данных все символы уже содержатся в смайликах, поэтому ответ равен нулю.

В седьмом наборе входных данных можно добавить один символ «^» в начало имени и один символ «^» в конец имени, тогда получится имя: ^_^.