

Задача А. Почтальон

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

В городе \mathcal{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 прийти 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 ($1 \leq n \leq 1000$). 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 , общее количество улиц также не превосходит 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

<code>post.in</code>	<code>post.out</code>
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 В. Самое дешёвое ребро

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

Дано подвешенное дерево с корнем в первой вершине. Все ребра имеют веса (стоимости). Вам нужно ответить на M запросов вида «найти у двух вершин минимум среди стоимостей ребер пути между ними».

Формат входных данных

В первой строке файла записано одно число — n (количество вершин).

В i -й из следующих $n - 1$ строк записано два целых числа x и y ($x \leq i$, $|y| \leq 10^6$) — предок вершины i и стоимость ребра.

Далее следуют m ($0 \leq m \leq 5 \cdot 10^4$) запросов вида (x, y) — найти минимум на пути из x в y ($x \neq y$).

Формат выходных данных

Выведите m ответов на запросы.

Примеры

<code>minonpath.in</code>	<code>minonpath.out</code>
5	2
1 2	2
1 3	
2 5	
3 2	
2	
2 3	
4 5	

Задача С. LCA - 2

Имя входного файла: lca2.in
Имя выходного файла: lca2.out
Ограничение по времени: 3 секунды
Ограничение по памяти: 256 мегабайт

Задано подвешенное дерево, содержащее n вершин, пронумерованных от 0 до $n - 1$. Требуется ответить на m запросов о наименьшем общем предке для пары вершин.

Запросы генерируются следующим образом. Заданы числа a_1, a_2 и числа x, y и z .

Числа a_3, \dots, a_{2m} генерируются следующим образом: $a_i = (x \cdot a_{i-2} + y \cdot a_{i-1} + z) \bmod n$. Первый запрос имеет вид $\langle a_1, a_2 \rangle$. Если ответ на $i - 1$ -й запрос равен v , то i -й запрос имеет вид $\langle (a_{2i-1} + v) \bmod n, a_{2i} \rangle$.

Формат входных данных

Первая строка содержит два числа: n ($1 \leq n \leq 100\,000$) и m ($1 \leq m \leq 10\,000\,000$). Корень дерева имеет номер 0. Вторая строка содержит $n - 1$ целых чисел, i -е из этих чисел это предок вершины i

Третья строка содержит целые числа a_1 и a_2 ($0 \leq a_i \leq n - 1$).

Четвёртая строка содержит три целых числа: x, y и z ($0 \leq x, y, z \leq 10^9$)

Формат выходных данных

Выведите в выходной файл сумму номеров вершин — ответов на все запросы.

Примеры

lca2.in	lca2.out
3 2 0 1 2 1 1 1 0	2
1 2 0 0 1 1 1	0

Задача D. Учиться!

Имя входного файла: `moscow.in`
Имя выходного файла: `moscow.out`
Ограничение по времени: 2 секунды
Ограничение по памяти: 64 мегабайта

Каждый год огромное количество выпускников, сдавшие ЕГЭ, выбирают, куда же они пойдут учиться. Не удивительно, что многие из них предпочитают перебраться поближе к столице. Транспортная инфраструктура страны переживает не лучшие времена, и в приемлемом качестве поддерживается минимально возможное число городов, необходимое для того, чтобы от любого города можно было добраться до любого другого.

Каждый выпускник оценивает свои результаты сдачи экзаменов, и решает, насколько далеко от своего родного города в сторону столицы он сможет уехать.

Выпускников настолько много, что вам не требуется выводить для каждого из них, до какого города он сможет доехать. Достаточно вывести сумму ответов для каждого выпускника.

Запросы генерируются следующим образом. Заданы числа a_1, a_2 и числа x, y и z . Числа a_3, \dots, a_{2m} генерируются следующим образом: $a_i = (x \cdot a_{i-2} + y \cdot a_{i-1} + z) \bmod n$. Первый запрос имеет вид $\langle a_1, a_2 \rangle$. Если ответ на $i - 1$ -й запрос равен v , то i -й запрос имеет вид $\langle (a_{2i-1} + v) \bmod n, a_{2i} \rangle$. В i -м запросе первое число соответствует городу, в котором окончил школу i -й выпускник, а второе — насколько далеко от родного города он может уехать. Все выпускники стараются перебраться как можно ближе к столице.

Формат входных данных

Первая строка содержит два числа: n ($1 \leq n \leq 100\,000$) и m ($1 \leq m \leq 10\,000\,000$). Столица имеет номер 0. Вторая строка содержит $n - 1$ целых чисел, i -е из этих чисел равно номеру следующего за городом i на пути к столице. Третья строка содержит два целых числа в диапазоне от 0 до $n - 1$: a_1 и a_2 . Четвертая строка содержит три целых числа: x, y и z , эти числа неотрицательны и не превосходят 10^9 .

Формат выходных данных

Выведите в выходной файл сумму номеров городов — ответов на все запросы.

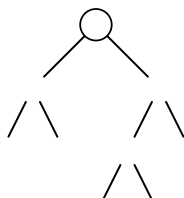
Примеры

<code>moscow.in</code>	<code>moscow.out</code>
3 2 0 1 2 1 1 1 0	1
1 2 0 0 1 1 1	0

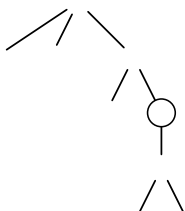
Задача E. Dynamic LCA

Имя входного файла: `dynamic.in`
Имя выходного файла: `dynamic.out`
Ограничение по времени: 5 секунд
Ограничение по памяти: 256 мегабайт

Постановка задачи о *наименьшем общем предке* прежде такова: дано дерево T с выделенным корнем и две вершины u и v , $\text{lca}(u, v)$ — вершина с максимальной глубиной, которая является предком и u , и v . Например, на картинке внизу $\text{lca}(8, 7)$ — вершина 3.



С помощью операции $\text{chroot}(u)$ мы можем менять корень дерева, достаточно отметить u , как новый корень, и направить ребра вдоль пути от корня. Наименьшие общие предки вершин поменяются соответственно. Например, если мы сделаем $\text{chroot}(6)$ на картинке сверху, $\text{lca}(8, 7)$ станет вершина 6. Получившееся дерево изображено внизу.



Вам дано дерево T . Изначально корень этого дерева — вершина 1. Напишите программу, которая поддерживает эти две операции: $\text{lca}(u, v)$ и $\text{chroot}(u)$.

Формат входных данных

Входной файл состоит из нескольких тестов.

Первая строка каждого теста содержит натуральное число n — количество вершин в дереве ($1 \leq n \leq 100\,000$). Следующие $n - 1$ строк содержат по 2 натуральных числа и описывают ребра дерева. Далее идет строка с единственным натуральным числом m — число операций. Следующие m строк содержат операции. Строка $? u v$ означает операцию $\text{lca}(u, v)$, а строка $! u$ — $\text{chroot}(u)$. Последняя строка содержит число 0.

Сумма n для всех тестов не превосходит 100 000. Сумма m для всех тестов не превосходит 200 000.

Формат выходных данных

Для каждой операции $? u v$ выведите значение $\text{lca}(u, v)$. Числа разделяйте переводами строк.

Примеры

dynamic.in	dynamic.out
9	2
1 2	1
1 3	3
2 4	6
2 5	2
3 6	3
3 7	6
6 8	2
6 9	
10	
? 4 5	
? 5 6	
? 8 7	
! 6	
? 8 7	
? 4 5	
? 4 7	
? 5 9	
! 2	
? 4 3	
0	

Задача F. Кварум

Имя входного файла:	stdin
Имя выходного файла:	stdout
Ограничение по времени:	7.5 секунды
Ограничение по памяти:	256 мегабайт

Второе августа 1702 года, среда. На болоте, ныне называемом Санкт-Петербург, собрались лягушки со всех ближайших водоемов. Для каждой из жабок Великим Уквакнителем была приготовлена определенная кувшинка. Оказалось, что лягушки могут общаться только с теми лягушками, кто сидит на лягушке похожего цвета. Давайте соединим такие кувшинки ребрами. Оказалось, что полученный граф является подвешенным деревом на n вершинах (в корне дерева сидит сам Великий Уквакнитель). Все кувшинки пронумерованы от 0 до $n - 1$.

Так как лягушки собрались, чтобы обсудить разрешение людям на строительство на их любимом болоте города, то некоторые пары лягушек хотят обмениваться своими аргументами друг с другом. Для ускорения процесса они просят вас найти их наименьшего общего предка.

Номера лягушек, которые в момент времени k ($1 \leq k \leq m$) хотят получить от вас их наименьшего предка, это a_{2k-1} и a_{2k} генерируются следующим образом:

- Изначально даны a_1, a_2 и числа x, y, z .
- Числа a_3, \dots, a_{2m} генерируются следующим образом: $a_i = (x \cdot a_{i-2} + y \cdot a_{i-1} + z) \bmod n$. Первый запрос имеет вид $\langle a_1, a_2 \rangle$. i -й запрос имеет вид $\langle a_{2i-1}, a_{2i} \rangle$.

Формат входных данных

Первая строка содержит два числа: n ($1 \leq n \leq 10^6$) и m ($1 \leq m \leq 10^7$).

Великий Уквакнитель сидит на кувшинке с номером 0.

Вторая строка содержит $n - 1$ целых чисел, i -е из этих чисел это предок вершины i

Третья строка содержит целые числа a_1 и a_2 ($0 \leq a_i \leq n - 1$).

Четвёртая строка содержит три целых числа: x, y и z ($0 \leq x, y, z \leq 10^9$)

Формат выходных данных

Выведите в выходной файл сумму номеров вершин — ответов на все запросы.

Примеры

stdin	stdout
3 2 0 1 2 1 1 1 0	1
1 2 0 0 1 1 1	0

Замечание

Сюда надо сдать Тарьяна. Остальные решения получают реджект

Задача G. Разбиение на пути

Имя входного файла: `paths.in`
Имя выходного файла: `paths.out`
Ограничение по времени: 1 секунда
Ограничение по памяти: 64 мегабайта

Разбейте рёбра неориентированного графа на минимальное количество рёберно-простых путей.

Формат входных данных

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

Формат выходных данных

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

Примеры

<code>paths.in</code>	<code>paths.out</code>
7 7	3
1 2	2 4 7 6
4 1	7 5
6 7	4 1 2 3
5 7	
7 4	
2 3	
4 2	

Задача Н. Таня и пароль

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

Пока папа был на работе, маленькая девочка Таня решила поиграть с папиным паролем к секретной базе данных. Папин пароль представляет собой строку, состоящую из $n+2$ символов. Она выписала все возможные n трёхбуквенных подстрок пароля на бумажки, по одной на каждую бумажку, а сам пароль выкинула. Каждая трёхбуквенная подстрока была выписана на бумажки столько раз, сколько она встречалась в пароле. Таким образом, в итоге у Тани оказалось n бумажек.

Потом Таня поняла, что папа расстроится, если узнает о ее игре, и решила восстановить пароль или, по крайней мере, хотя бы какую-то строку, соответствующую получившемуся набору трёхбуквенных строк. Вам предстоит помочь ей в этой непростой задаче. Известно, что папин пароль состоял из строчных и заглавных букв латинского алфавита, а также из цифр. Строчные и заглавные буквы латинского алфавита считаются различными.

Формат входных данных

В первой строке следует целое число n ($1 \leq n \leq 2 \cdot 10^5$), количество трёхбуквенных подстрок, которые получились у Тани.

Следующие n строк каждая содержат по три буквы, образующие подстроку пароля папы. Каждый символ во вводе — строчная или заглавная буква латинского алфавита или цифра.

Формат выходных данных

Если во время игры Таня что-то напутала, и строк, соответствующих данному набору подстрок, не существует, то выведите «NO».

Если же возможно восстановить строку, соответствующую данному набору подстрок, то выведите «YES», а затем любой подходящий вариант пароля.

Примеры

стандартный ввод	стандартный вывод
5 aca aba aba cab bac	YES abacaba
4 abc bCb cb1 b13	NO
7 aaa aaa aaa aaa aaa aaa aaa	YES aaaaaaaaa

Задача I. Опекуны карнотавров

Имя входного файла:	carno.in
Имя выходного файла:	carno.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Карнотавры очень внимательно относятся к заботе о своем потомстве. У каждого динозавра обязательно есть старший динозавр, который его опекает. В случае, если опекуна съедают (к сожалению, в юрский период такое не было редкостью), забота о его подопечных ложится на плечи того, кто опекал съеденного динозавра. Карнотавры — смертоносные хищники, поэтому их обычаи строго запрещают им драться между собой. Если у них возникает какой-то конфликт, то, чтобы решить его, они обращаются к кому-то из старших, которому доверяют, а доверяют они только тем, кто является их опекуном или опекуном их опекуна и так далее (назовем таких динозавров суперопекунами). Поэтому для того, чтобы решить спор двух карнотавров, нужно найти такого динозавра, который является суперопекуном для них обоих. Разумеется, беспокоить старших по пустякам не стоит, поэтому спорщики стараются найти самого младшего из динозавров, который удовлетворяет этому условию. Если у динозавра возник конфликт с его суперопекуном, то этот суперопекун сам решит проблему. Если у динозавра нелады с самим собой, он должен разобраться с этим самостоятельно, не беспокоя старших. Помогите динозаврам разрешить их споры.

Формат входных данных

В первой строке содержит целое число M ($1 \leq M \leq 200\,000$) — количество запросов. Далее следуют M запросов, описывающие события:

- $+ v$ — родился новый динозавр и опекунство над ним взял динозавр с номером v . Родившемуся динозавру нужно присвоить наименьший натуральный номер, который до этого еще никогда не встречался.
- $- v$ — динозавра номер v съели.
- $? u v$ — у динозавров с номерами u и v возник конфликт и вам надо найти им третейского судью.

Изначально есть один прадинозавр номер 1. Гарантируется, что он никогда не будет съеден.

Формат выходных данных

Для каждого запроса типа «?» в выходной файл нужно вывести на отдельной строке одно число — номер самого молодого динозавра, который может выступить в роли третейского судьи.

Примеры

carno.in	carno.out
11	1
+ 1	1
+ 1	2
+ 2	2
? 2 3	5
? 1 3	
? 2 4	
+ 4	
+ 4	
- 4	
? 5 6	
? 5 5	