

Задача А. LCA

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

Дан ориентированный граф. Подсчитайте, сколько пар вершин (i, j) имеют общего предка. Общим предком вершин i и j называется такая вершину k , что и i , и j достижимы из k .

Формат входного файла

В первой строке входного файла содержатся целые числа $1 \leq N \leq 10^4, 0 \leq M \leq 10^4$ — количество вершин и рёбер в графе. Далее следуют M строк по два числа от 1 до N . Пара чисел (a, b) означает, что из вершины a есть ребро в вершину b .

Формат выходного файла

Выведите одно целое число — количество пар.

Примеры

<code>lca.in</code>	<code>lca.out</code>
2 1 1 2	4
3 2 2 1 3 1	7

Задача В. LCA - 2

Имя входного файла: `lca2.in`
 Имя выходного файла: `lca2.out`
 Ограничение по времени: 2 секунды
 Ограничение по памяти: 256 мегабайт

Задано подвешенное дерево, содержащее n ($1 \leq n \leq 100\,000$) вершин, пронумерованных от 0 до $n - 1$. Требуется ответить на m ($1 \leq m \leq 10\,000\,000$) 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 (a_1, a_2) . Если ответ на $i - 1$ -й запрос равен v , то i -й запрос имеет вид $((a_{2i-1} + v) \bmod n, a_{2i})$.

Формат входного файла

Первая строка содержит два числа: n и m . Корень дерева имеет номер 0. Вторая строка содержит $n - 1$ целых чисел, i -е из этих чисел равно номеру родителя вершины i . 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 выходного файла

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

Примеры

<code>lca2.in</code>	<code>lca2.out</code>
3 2 0 1 2 1 1 1 0	2
1 2 0 0 1 1 1	0

Задача С. LCA-3

Имя входного файла: `lca3.in`
 Имя выходного файла: `lca3.out`
 Ограничение по времени: 2 секунды
 Ограничение по памяти: 256 мегабайт

Подвешенное дерево — это ориентированный граф без циклов, в котором в каждую вершину, кроме одной, называемой *корнем* ориентированного дерева, входит одно ребро. В корень ориентированного дерева не входит ни одного ребра. *Отцом* вершины называется вершина, ребро из которой входит в данную.

(по материалам Wikipedia)

Дан набор подвешенных деревьев. Требуется выполнять следующие операции:

- 0 u и v Для двух заданных вершин u и v выяснить, лежат ли они в одном дереве. Если это так, вывести вершину, являющуюся их наименьшим общим предком, иначе вывести 0.
- 1 u и v Для корня u одного из деревьев и произвольной вершины v другого дерева добавить ребро (v, u) . В результате эти два дерева соединятся в одно.

Вам необходимо выполнять все операции online, т.е. вы сможете узнать следующий запрос только выполнив предыдущий.

Формат входного файла

На первой строке входного файла находится число n — суммарное количество вершин в рассматриваемых деревьях. На следующей строке расположено n чисел — предок каждой вершины в начальной конфигурации, или 0, если соответствующая вершина является корнем. Затем следует число k — количество запросов к вашей программе. Каждая из следующих строк содержит по три целых числа: вид запроса (0 — для поиска LCA или 1 — для добавления ребра) и два числа x, y . Вершины, участвующие в запросе можно вычислить по формуле: $u = (x - 1 + ans) \bmod n + 1$, $v = (y - 1 + ans) \bmod n + 1$, где ans - ответ на последний запрос типа 0.

Формат выходного файла

Для каждого запроса типа 0, выведите в выходной файл одно число на отдельной строке — ответ за этот запрос.

Примеры

lca3.in	lca3.out
5	0
0 0 0 0 0	5
12	5
1 5 3	3
0 2 5	2
1 4 2	3
1 1 5	3
0 1 5	2
1 3 4	
0 1 5	
0 3 1	
0 4 2	
0 1 4	
0 5 2	
0 4 1	

Задача D. Ожерелье

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

Ювелир должен сделать эксклюзивное ожерелье для Королевы. Ожерелье должно состоять из серебряных золотых и бронзовых бусинок, расположение которых строго специфицировано. Золотые бусинки одинаковые и могут использоваться взаимозаменяемо, аналогично обстоит дело с серебряными и бронзовыми бусинками. Ювелир подготовил бусины для работы и нанизал их на один длинный стержень. Теперь он готов собирать ожерелье снимая бусины одну за одной со стержня и нанизывая на шнурок с любой из сторон, а в завершение процесса соединяя два конца шнурка. Соединение будет незаметно, поэтому оно может быть между любыми двумя бусинами.

К несчастью, бусины на стержне могут не быть в том же порядке, в котором они появятся на ожерелье. Поэтому в процессе сборки ожерелья, ювелир может брать бусины со стержня и откладывать в сторону. Ювелир хочет минимизировать максимальное количество бусин, которые он отложит в сторону в процессе сборки ожерелья.

Формат входного файла

Первая строка ввода содержит одно целое число L ($1 \leq L \leq 1000$) - количество бусин в ожерелье. Следующая строка содержит строку из L букв (каждая из которых либо G , либо

S , либо B , означающая золотую, серебряную или бронзовую бусину), которая описывает финальное состояние ожерелья (разрезанного в произвольной точке и выпрямленного). Третья строка содержит строку из L букв, описывающую порядок бусин на палочке. Ювелир может брать бусины только с левого конца палочки. Гарантируется, что возможно собрать ожерелье из заданного расположения бусин.

Формат выходного файла

Вывод должен содержать одну строку - минимально возможное максимальное количество бусин, которые ювелир отложит в сторону в процессе сборки ожерелья.

Примеры

necklace.in	necklace.out
8 GSGSGSGS SSSSGGGG	3
8 SSSGGGBB GSGSGSBB	0