

Содержание

Сканирующей прямой много не бывает	2
Задача А. <i>K</i> -инверсии [0.2 sec, 256 mb]	2
LCA и RMQ, первое знакомство	3
Задача В. Ancestor. Предок [0.2 sec, 256 mb]	3
Задача С. Общий предок [0.2 sec, 256 mb]	4
Задача D. Дерево [0.6 sec, 256 mb]	5
Задача Е. Самое дешёвое ребро [0.3 sec, 256 mb]	6
Задача F. LCA Problem Revisited [2 sec, 256 mb]	7
Бонус	8
Задача G. Черепахи и повороты (задача на 4) [0.4 sec, 256 mb]	8
Задача H. LCA-3 [2 sec, 256 mb]	9
Задача I. Опекуны карнотавров [0.6 sec, 256 mb]	10
Задача J. Дерево [3 sec, 512 mb]	11

В некоторых задачах большой ввод и вывод. Имеет смысл пользоваться супер быстрым вводом-выводом:

http://acm.math.spbu.ru/~sk1/algo/input-output/fread_write.cpp.html

В некоторых задачах нужен STL, который активно использует динамическую память (set-ы, map-ы) переопределение стандартного аллокатора ускорит вашу программу:

<http://acm.math.spbu.ru/~sk1/algo/memory.cpp.html>

Сканирующей прямой много не бывает

Задача А. K -инверсии [0.2 sec, 256 mb]

Пусть дана перестановка a_1, a_2, \dots, a_n . Назовем k -инверсией набор чисел i_1, i_2, \dots, i_k таких, что $1 \leq i_1 < i_2 < \dots < i_k \leq n$ и $a_{i_1} > a_{i_2} > \dots > a_{i_k}$. Ваша задача — подсчитать количество различных k -инверсий в заданной перестановке.

Формат входных данных

В первой строке входного файла находятся число n — длина перестановки ($1 \leq n \leq 20\,000$), и число k ($2 \leq k \leq 10$). Во второй строке n чисел — сама перестановка.

Формат выходных данных

В выходной файл выведите единственное число — количество k -инверсий в заданной перестановке по модулю 10^9 .

Пример

kinverse.in	kinverse.out
3 2	2
3 1 2	
5 3	10
5 4 3 2 1	

LCA и RMQ, первое знакомство

Задача В. Ancestor. Предок [0.2 sec, 256 mb]

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 ($1 \leq n \leq 100\,000$) — количество вершин в дереве. Во второй строке находится n чисел. При этом i -ое число второй строки определяет непосредственного родителя вершины с номером i . Если номер родителя равен нулю, то вершина является корнем дерева.

В третьей строке находится число m ($1 \leq m \leq 100\,000$) — количество запросов. Каждая из следующих m строк содержит два различных числа a и b ($1 \leq a, b \leq n$).

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

ancestor.in	ancestor.out
6	0
0 1 1 2 3 3	1
5	1
4 1	0
1 4	0
3 6	
2 6	
6 5	

Задача С. Общий предок [0.2 sec, 256 mb]

Дано подвешенное дерево с корнем в 1-й вершине и M запросов вида “найти у двух вершин наименьшего общего предка”.

Формат входных данных

В первой строке файла записано одно число N — количество вершин. В следующих $N - 1$ строках записаны числа. Число x на строке $2 \leq i \leq N$ означает, что x — отец вершин i . ($x < i$). На следующей строке число M . Следующие M строк содержат запросы вида (x, y) — найти наименьшего предка вершин x и y . Ограничения: $1 \leq N \leq 5 \cdot 10^4, 0 \leq M \leq 5 \cdot 10^4$.

Формат выходных данных

M ответов на запросы.

Пример

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

Задача D. Дерево [0.6 sec, 256 mb]

Дано взвешенное дерево. Найти кратчайшее расстояние между заданными вершинами.

Формат входных данных

Первая строка входного файла содержит натуральное число $N \leq 150\,000$ — количество вершин в графе. Вершины нумеруются целыми числами от 0 до $N - 1$. В следующих $N - 1$ строках содержится по три числа u, v, w , которые соответствуют ребру весом w , соединяющему вершины u и v . Веса — целые числа от 0 до 10^9 . В следующей строке содержится натуральное число $M \leq 75\,000$ — количество запросов. В следующих M строках содержится по два числа u, v — номера вершин, расстояние между которыми необходимо вычислить.

Формат выходных данных

Для каждого запроса выведите на отдельной строке одно число — искомое расстояние. Гарантируется, что ответ помещается в знаковом 32-битном целом типе.

Пример

tree.in	tree.out
3	0
1 0 1	1
2 0 1	1
9	1
0 0	0
0 1	2
0 2	1
1 0	2
1 1	0
1 2	
2 0	
2 1	
2 2	

Задача E. Самое дешёвое ребро [0.3 sec, 256 mb]

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 числ — n (количество вершин).

В следующих $n - 1$ строках записаны два числа — x и y . Число x на строке i означает, что x — предок вершины i , y означает стоимость ребра.

$x < i, |y| \leq 10^6$.

Далее m запросов вида (x, y) — найти минимум на пути из x в y ($x \neq y$).

Ограничения: $2 \leq n \leq 5 \cdot 10^4, 0 \leq m \leq 5 \cdot 10^4$.

Формат выходных данных

m ответов на запросы.

Пример

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

Задача F. LCA Problem Revisited [2 sec, 256 mb]

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 $\langle a_1, a_2 \rangle$. 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 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 выходных данных

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

Примеры

lca_rmq.in	lca_rmq.out
3 2	2
0 1	
2 1	
1 1 0	

Бонус

Задача G. Черепахи и повороты (задача на 4) [0.4 sec, 256 mb]

Для тренировки боевых черепах военные построили прямоугольный полигон размером $W \times H$ клеток. Некоторые клетки проходимы для черепах, а некоторые — нет. Черепахи могут перемещаться только параллельно сторонам полигона. Полигон сконструирован таким образом, что существует единственный способ добраться от любой проходимой клетки до любой другой проходимой клетки, не проходя при этом по одной и той же клетке дважды. Известно, что черепахи очень быстро бегают по прямой, но испытывают трудности при повороте на 90 градусов. Поэтому сложность маршрута определяется как количество поворотов, которое придётся совершить черепахе при переходе от начальной до конечной клетки маршрута. Вы должны написать программу, вычисляющую сложность маршрута по его начальной и конечной клетке.

Формат входных данных

В первой строке через пробел записаны два целых числа H и W — размеры полигона ($1 \leq W \cdot H \leq 100\,000$). Далее задаётся карта полигона — H строк по W символов в каждой. Символ '#' обозначает проходимую клетку, а '.' — непроходимую. В $H+2$ -й строке записано целое число Q — количество маршрутов, для которых нужно посчитать сложность ($1 \leq Q \leq 50\,000$). В каждой из следующих Q строк через пробел записаны четыре целых числа: номер строки и номер столбца начальной клетки маршрута, номер строки и номер столбца конечной клетки маршрута. Гарантируется, что начальная и конечная клетки маршрута являются проходимыми. Строки занумерованы числами от 1 до H сверху вниз, а столбцы — числами от 1 до W слева направо.

Формат выходных данных

Для каждого маршрута выведите в отдельной строке одно число — его сложность.

Примеры

turtles.in	turtles.out
5 4	1
.#..	0
###.	2
..##	3
.##.	
....	
4	
1 2 2 1	
2 3 4 3	
4 2 3 4	
1 2 4 2	

Задача Н. LCA-3 [2 sec, 256 mb]

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:

1. 0 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 .
2. 1 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 входных данных

На первой строке входного файла находится число n — суммарное количество вершин в рассматриваемых деревьях, $1 \leq n \leq 50\,000$. 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, $1 \leq k \leq 100\,000$. 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 ($ans = 0$ для первого запроса).

Формат выходных данных

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 I. Опекуны карнотавров [0.6 sec, 256 mb]

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 неладит с самим собой, он должен разобраться с этим самостоятельно, не беспокоя старших. Помогите динозаврам разрешить их споры.

Формат входных данных

Во входном файле записано число M , обозначающее количество запросов ($1 \leq M \leq 200\,000$). Далее на отдельных строках следуют M запросов, обозначающих следующие 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	

Задача J. Дерево [3 сек, 512 mb]

Задано подвешенное дерево, содержащее n ($1 \leq n \leq 1\,000\,000$) вершин. Каждая вершина покрашена в один из n цветов. Требуется для каждой вершины v вычислить количество различных цветов, встречающихся в поддереве с корнем v .

Формат входных данных

В первой строке входного файла задано число n . Последующие n строк описывают вершины, по одной в строке. Описание очередной вершины i имеет вид $p_i c_i$, где p_i — номер родителя вершины i , а c_i — цвет вершины i ($1 \leq c_i \leq n$). Для корня дерева $p_i = 0$.

Формат выходных данных

Выведите n чисел, обозначающих количества различных цветов в поддеревьях с корнями в вершинах $1, \dots, n$.

Примеры

tree.in	tree.out
5	1 2 3 1 1
2 1	
3 2	
0 3	
3 3	
2 1	