Содержание

Общая информация:

Вход в контест: `http://contest.yandex.ru/contest/1087/`
Дедлайн на задачи: 2 недели, до 1-го марта 23:59.
К каждой главе есть более простые задачи (base), посложнее (advanced), и сложные (hard).
В скобках к каждой главе написано сколько любых задач из этой главы нужно сдать.
Сайт курса: `https://compscicenter.ru/courses/algorithms-2/2015-spring/`
Семинары ведёт Сергей Владимирович Копелиович,
контакты: burunduk30@gmail.com, vk.com/burunduk1

В каждом условии 2 таймлимита: для `C/C++` и для `Java, Python`.

01.Base [2/2]

Задача 1A. Ancestor. Предок [0.3 sec (0.7 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 ($1 \leqslant n \leqslant 100\,000$) — 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 ($1 \leqslant m \leqslant 100\,000$) — количество запросов. Каждая из следующих m строк содержит два различных числа a и b ($1 \leqslant a, b \leqslant n$).

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 1 1 2 3 3 5 4 1 1 4 3 6 2 6 6 5	0 1 1 0 0

Задача 1B. Общий предок [0.5 sec (1 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 $2 \leqslant i \leqslant N$ 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: $1 \leqslant N \leqslant 5 \cdot 10^4, 0 \leqslant M \leqslant 5 \cdot 10^4$.

Формат выходных данных

M ответов на запросы.

Пример

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

01.Advanced [2/2]

Задача 1C. Самое дешевое ребро [0.5 sec (1 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.

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

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

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

Формат выходных данных

m ответов на запросы.

Пример

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

Задача 1D. Дерево [1 sec (2 sec), 256 mb]

Дано взвешенное дерево. Найти кратчайшее расстояние между заданными вершинами.

Формат входных данных

Первая строка входного файла содержит натуральное число $N \leqslant 150\,000$ — 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 $M \leqslant 75\,000$ — 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 1 0 1 2 0 1 9 0 0 0 1 0 2 1 0 1 1 1 2 2 0 2 1 2 2	0 1 1 1 0 2 1 2 0

01.Hard [0/2]

Задача 1E. Опекуны карнотавров [0.7 sec (1.5 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 динамозавра. 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

Во входном файле записано число M, обозначающее количество запросов ($1 \le M \le 200\,000$). 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 + 2 ? 2 3 ? 1 3 ? 2 4 + 4 + 4 - 4 ? 5 6 ? 5 5	1 1 2 2 5

Задача 1F. LCA-3 [1.5 sec (3 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 u 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 u 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, $1 \leqslant n \leqslant 50\,000$. На следующей строке расположено n чисел — предок каждой вершины в начальной конфигурации, или 0, если соответстующая вершина является корнем. Затем следует число k — количество запросов к вашей программе, $1 \leqslant k \leqslant 100\,000$. 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 12 1 5 3 0 2 5 1 4 2 1 1 5 0 1 5 1 3 4 0 1 5 0 3 1 0 4 2 0 1 4 0 5 2 0 4 1	0 5 5 3 2 3 3 2