Содержание

У вас не получается читать/выводить данные?
Воспользуйтесь примерами (c++) (python).

Обратите внимание, входные данные лежат в **стандартном потоке ввода** (он же stdin), вывести ответ нужно в **стандартный поток вывода** (он же stdout).

Обратите внимание на `GNU C++` компиляторы с суффиксом `inc`.
Подни можно пользоваться дополнительной библиотекой (`optimization.h`).
То есть, использовать быстрый ввод-вывод: **пример про числа и строки**.
И быструю аллокацию памяти (ускоряет vector-set-map-весь-STL): пример.

Для тех, кто хочет разобраться, как всё это работает.
Короткая версия быстрого ввода-вывода (тык) и короткая версия аллокатора (тык).

Must have

Задача 29A. Ancestor. Предок [0.1 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

stdin	stdout
6 0 1 1 2 3 3 5 4 1 1 4 3 6 2 6 6 5	0 1 1 0 0

Подсказка по решению

Мы умеем очень просто отвечать на такие запросы за $\mathcal{O}(1)$.

Задача 29B0. Общий предок [0.1 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

stdin	stdout
5 1 1 2 3 2 2 3 4 5	1 1

Подсказка по решению

Потренируйтесь в двоичных подъёмах. Они ещё пригодятся.

Задача 29B. Дерево [0.25 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

stdin	stdout
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

Подсказка по решению

Вы же уже сдали LCA, да? Воспользуйтесь им.

Задачи здорового человека

Задача 29C. Самое дешевое ребро [0.1 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

stdin	stdout
5 1 2 1 3 2 5 3 2 2 2 3 4 5	2 2

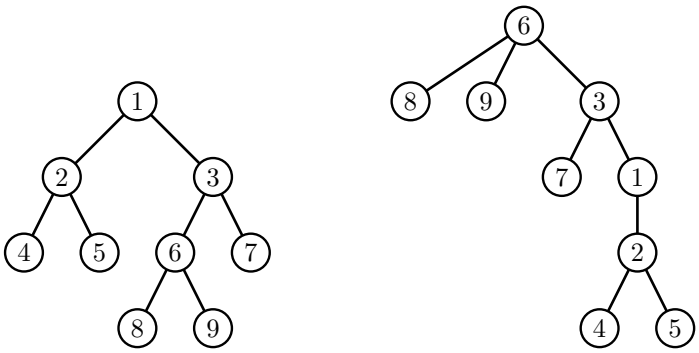
Замечание

Да, мы уже делали что-то такое центроидами

Но именно в этой задаче обязательно сдать двоичные подъёмы.

Задача 29D. Dynamic LCA [0.4 sec, 256 mb]

Наименьший общий предок в дереве для вершин u и v – вершина $lca(u, v)$, которая является предком и u, и v и при этом имеем максимальную глубину. Например, $lca(8, 7) = 3$ в дереве на картинке слева. В дереве можно поменять корень. $chroot(u)$ делает корнем вершину u. Например, после операции $chroot(6)$, как видно на картинке справа, $lca(8, 7) = 6$.



Вам дано дерево с корнем в вершине 1. Напишите программу, отвечающую на запросы $lca(u, v)$ и $chroot(u)$.

Формат входных данных

Во входных данных дано несколько тестов.

Каждый тест начинается с количества вершин в дереве n ($1 \leqslant n \leqslant 100\,000$). Следующие $n - 1$ строк содержат пары целых чисел от 1 до n – рёбра дерева. Далее число запросов m ($1 \leqslant m \leqslant 200\,000$). Каждая из следующих m строк имеет вид "`?` u v" для запроса $lca(u, v)$ или "`!` u" для запроса $chroot(u)$.

Последний тест имеет $n = 0$, его обрабатывать не нужно.

Сумма n по всем тестам не более 100 000. Сумма m по всем тестам не более 200 000.

Формат выходных данных

Для каждой операции вида "`?` u v" выведите $lca(u, v)$.

Примеры

stdin	stdout
9 1 2 1 3 2 4 2 5 3 6 3 7 6 8 6 9 10 ? 4 5 ? 5 6 ? 8 7 ! 6 ? 8 7 ? 4 5 ? 4 7 ? 5 9 ! 2 ? 4 3 0	2 1 3 6 2 3 6 2

Подсказка по решению

Если у вас уже написано обычное LCA, технически больше ничего не нужно. Осталось придумать, как им воспользоваться.

Задача 29E. LCA Problem Revisited [0.6 sec, 256 mb]

Задано подвешенное дерево, содержащее n ($1 \leqslant n \leqslant 100\,000$) вершин, пронумерованных от 0 до $n-1$. Требуется ответить на m ($1 \leqslant m \leqslant 10\,000\,000$) 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 $a_3, \ldots, a_{2m}$ 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

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

Подсказка по решению

Эйлеров обход и спарсы.

Любители могут сдать ФКБ.

Задача 29F. K-инверсии [0.1 sec, 256 mb]

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

Формат входных данных

В первой строке входного файла находятся число n — длина перестановки ($1 \leqslant n \leqslant 20\,000$), и число k ($2 \leqslant k \leqslant 10$). 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

stdin	stdout
3 2 3 1 2	2
5 3 5 4 3 2 1	10

Подсказка по решению

Придумайте сперва динамику. Без всяких деревьев.

Задача 29G. Перестановки [0.5 sec, 256 mb]

Вася выписал на доске в каком-то порядке все числа от 1 по N, каждое число ровно по одному разу. Количество чисел оказалось довольно большим, поэтому Вася не может окинуть взглядом все числа. Однако ему надо всё-таки представлять эту последовательность, поэтому он написал программу, которая отвечает на вопрос — сколько среди чисел, стоящих на позициях с x по y, по величине лежат в интервале от k до l. Сделайте то же самое.

Формат входных данных

В первой строке лежит два натуральных числа — $1 \leqslant N \leqslant 100\,000$ — количество чисел, которые выписал Вася и $1 \leqslant M \leqslant 100\,000$ — количество вопросов, которые Вася хочет задать программе. Во второй строке дано N чисел — последовательность чисел, выписанных Васей. Далее в M строках находятся описания вопросов. Каждая строка содержит четыре целых числа $1 \leqslant x \leqslant y \leqslant N$ и $1 \leqslant k \leqslant l \leqslant N$.

Формат выходных данных

Выведите M строк, каждая должна содержать единственное число — ответ на Васин вопрос.

Пример

stdin	stdout
4 2 1 2 3 4 1 2 2 3 1 3 1 3	1 3

Подсказка по решению

Видите двумерные запросы? Помните, что они решаются сканлайном?

Дерево отрезков сортированных массивов скорее всего тоже зайдёт.

Для искателей острых ощущений

Задача 29H. Черепахи и повороты [0.5 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 ($1 \leqslant W \cdot H \leqslant 100\,000$). 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 ($1 \leqslant Q \leqslant 50\,000$). 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

stdin	stdout
5 4 .#.. ###. ..## .##. 4 1 2 2 1 2 3 4 3 4 2 3 4 1 2 4 2	1 0 2 3

Подсказка по решению

Просто LCA, да? Простая задача.

Задача 29I. LCA-3 [0.8 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

stdin	stdout
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

Задача 29J. Дерево [0.5 sec, 256 mb]

Задано подвешенное дерево, содержащее n ($1 \leqslant n \leqslant 1\,000\,000$) 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 ($1 \leqslant c_i \leqslant n$). Для корня дерева $p_i = 0$.

Формат выходных данных

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

Примеры

stdin	stdout
5 2 1 3 2 0 3 3 3 2 1	1 2 3 1 1

Для мастеров AI

Правила.
Это блок задач про промтинг. Пользоваться можно только бесплатными версиями ИИ. В шапке исходника указывать последовательность промтов, и сайты, куда они были отправлены. Если в процессе использования ИИ вы получили какие-то важные идеи для решения, это тоже часть решения. Если с помощью ИИ вы написали генератор тестов или стресс-тест, это тоже следует указывать. Задокументируйте, пожалуйста, проделанную работу в шапке отосланного решения. Если это нужно, вы можете писать часть кода сами.

Задача 29K. Выстрелы по стенам [1.5 sec, 32 mb]

Производится испытание нового пистолета, который может производить выстрелы с различными скоростями пуль. В некоторые моменты времени происходят выстрелы из начала координат с определенными горизонтальными скоростями, и в некоторые другие моменты строят стены на горизонтальной площадке — невырожденные отрезки, лежащие на прямых, не проходящих через начало координат. При этом стены могут пересекаться. Для обработки результатов эксперимента необходимо определить, сколько времени летела каждая пуля. Пуля летит с постоянной скоростью. Пуля останавливается сразу при касании стены.

Формат входных данных

В начале каждой строки написано одно из трех слов: `shot`, `wall` или `end`. Число строк не превышает 50 000. После слова `shot` следуют две координаты скорости пули. Скорость пули не может равняться нулю. После слова `wall` следуют четыре числа — координаты начала и конца стены. Слово `end` является признаком окончания набора входных данных. Все координаты являются целыми числами, по модулю не превосходящими 10 000. Все события записаны в хронологическом порядке, и интервалы времени между событиями больше, чем время, за которое строится стена, и чем время, за которое пуля пролетает расстояние до ближайшей стены или за границу испытательного полигона.

Формат выходных данных

Для каждого выстрела вывести на отдельной строке одно число — время, которое пролетит пуля, с точностью до 10^{-6}. Если пуля не попадет ни в какую стену, то вместо числа вывести слово `Infinite`.

Пример

stdin	stdout
shot 1 0 wall 1 0 0 1 shot 1 1 shot -1 3 wall 1 0 -1 2 shot -1 3 wall 1 1 -1 1 shot -1 3 wall 2 3 2 -3 wall 3 -2 -3 -2 shot 1 -1 shot 40 -39 shot 9999 -10000 shot -1 -1 shot -3000 -2000 shot -3001 -2000 shot -3000 -2001 shot 1 0 shot 1 1 wall -1 2 10 -10 shot -1 1 shot 0 1 shot 1 1 shot 1 0 shot 1 -1 wall 0 -10000 -10000 0 shot -2 -1 end	Infinite 0.50000000000000000000 Infinite 0.50000000000000000000 0.33333333333333333333 2.00000000000000000000 0.05000000000000000000 0.00020000000000000000 2.00000000000000000000 0.00100000000000000000 Infinite 0.00099950024987506247 1.00000000000000000000 0.50000000000000000000 1.00000000000000000000 0.90909090909090909091 0.43478260869565217391 0.83333333333333333333 2.00000000000000000000 3333.33333333333333333