

Содержание

Must have	2
Задача 2H. Поднимайся и вращай [0.5 sec, 256 mb]	2
Обязательные задачи	3
Задача 2A. Динамический Лес [1.2 sec, 256 mb]	3
Задача 2B. Пещеры и туннели [1 sec, 256 mb]	4
Дополнительные задачи	5
Задача 2C. Count Offline [2.5 sec, 256 mb]	5
Задача 2D. Union [1.5 sec, 256 mb]	6
Задача 2E. Inspector is Coming [1.5 sec, 256 mb]	7
Задача 2F. Distance Sum [5.0 sec, 256 mb]	8
Задача 2G. Прямоугольные запросы [6.5 sec, 256 mb]	9

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

В некоторых задачах большой ввод и вывод. Пользуйтесь **быстрым вводом-выводом**.

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

Обратите внимание на GNU C++ компиляторы с суффиксом `inc`, они позволяют пользоваться **дополнительной библиотекой**. Под ними можно сдать **вот это**.

Must have

Задача 2Н. Поднимайся и вращай [0.5 sec, 256 mb]

Изначально у вас есть n чисел $1, 2, \dots, n$. Каждое живёт само по себе. Далее числа будут объединяться в массивы. Вам нужно реализовать структуру, данных, умеющую отвечать на несколько запросов, за $\mathcal{O}(\log n)$ каждый.

- $+ i j$ – взять массивы, в которых живут числа i и j , и объединить их в один массив именно в таком порядке.
- $! i k$ – взять массив, в котором живёт число i , и повернуть его на k влево.
- $- i k$ – взять массив, в котором живёт число i , и отрезать первые k элементов. Получится два новых массива.

Гарантируется, что запросы корректны. В первом i и j живут в разных массивах, во втором и третьем, длина массива, содержащего i , строго больше k .

Формат входных данных

На первой строке число элементов n ($2 \leq n \leq 100\,000$) и число запросов m ($1 \leq m \leq 100\,000$).
На следующих m строках сами запросы.

Формат выходных данных

После всех запросов нужно вывести получившиеся массивы. На первой строке выведите число массивов. Далее k массивов в формате «число элементов и сами элементы». Отсортируйте массив массивов перед выводом (массивы сравниваются лексикографически).

Пример

stdin	stdout
5 5	3
+ 1 2	1 2
+ 1 3	3 3 4 1
+ 1 4	1 5
! 3 1	
- 2 1	

Подсказка по решению

Вам нужно каждому элементу сопоставить `node*` и научиться подниматься от неё до корня соответствующего декартова дерева.

Замечание

То, что вы напишите в этой задаче, это уже почти ETT ;-)

Обязательные задачи

Задача 2А. Динамический Лес [1.2 sec, 256 mb]

Вам нужно научиться обрабатывать 3 типа запросов:

1. Добавить ребро в граф (`link`).
2. Удалить ребро из графа (`cut`).
3. По двум вершинам a и b , определить, лежат ли они в одной компоненте связности (`get`).

Изначально граф пустой (содержит N вершин, не содержит ребер). Гарантируется, что в любой момент времени граф является лесом. При добавлении ребра гарантируется, что его сейчас в графе нет. При удалении ребра гарантируется, что оно уже добавлено.

Формат входных данных

Числа N и M ($1 \leq N \leq 10^5 + 1$, $1 \leq M \leq 10^5$) — количество вершин в дереве и, соответственно, запросов. Далее M строк, в каждой строке команда (`link` или `cut`, или `get`) и 2 числа от 1 до N — номера вершин в запросе.

Формат выходных данных

В выходной файл для каждого запроса `get` выведите 0, если не лежат, или 1, если лежат.

Пример

stdin	stdout
3 7 get 1 2 link 1 2 get 1 2 cut 1 2 get 1 2 link 1 2 get 1 2	0101
5 10 link 1 2 link 2 3 link 4 3 cut 3 4 get 1 2 get 1 3 get 1 4 get 2 3 get 2 4 get 3 4	110100

Подсказка по решению

ETT — декартово дерево понеявному ключу на массиве «эйлеров обход из ориентированных рёбер».

Задача 2В. Пещеры и туннели [1 сек, 256 mb]

После посадки на Марс учёные нашли странную систему пещер, соединённых туннелями. И учёные начали исследовать эту систему, используя управляемых роботов. Было обнаружено, что существует ровно один путь между каждой парой пещер. Но потом учёные обнаружили специфическую проблему. Иногда в пещерах происходят небольшие взрывы. Они вызывают выброс радиоактивных изотопов и увеличивают уровень радиации в пещере. К сожалению, роботы плохо выдерживают радиацию. Но для исследования они должны переместиться из одной пещеры в другую. Учёные поместили в каждую пещеру сенсор для мониторинга уровня радиации. Теперь они каждый раз при движении робота хотят знать максимальный уровень радиации, с которым придётся столкнуться роботу во время его перемещения. Как вы уже догадались, программу, которая это делает, будете писать вы.

Формат входных данных

Первая строка входного файла содержит одно целое число N ($1 \leq N \leq 100\,000$) — количество пещер. Следующие $N - 1$ строк описывают туннели. Каждая из этих строк содержит два целых числа — a_i и b_i ($1 \leq a_i, b_i \leq N$), описывающие туннель из пещеры с номером a_i в пещеру с номером b_i . Следующая строка содержит целое число Q ($1 \leq Q \leq 100\,000$), означающее количество запросов. Далее идут Q запросов, по одному на строку. Каждый запрос имеет вид « $C U V$ », где C — символ «I» либо «G», означающие тип запроса (кавычки только для ясности). В случае запроса «I» уровень радиации в U -й пещере ($1 \leq U \leq N$) увеличивается на V ($0 \leq V \leq 10\,000$). В случае запроса «G» ваша программа должна вывести максимальный уровень радиации на пути между пещерами с номерами U и V ($1 \leq U, V \leq N$) после всех увеличений радиации (запросов «I»), указанных ранее. Предполагается, что изначальный уровень радиации равен 0 во всех пещерах, и он никогда не уменьшается со временем (потому что период полураспада изотопов много больше времени наблюдения).

Формат выходных данных

Для каждого запроса «G» выведите максимальный уровень радиации.

Пример

stdin	stdout
4	1
1 2	0
2 3	1
2 4	3
6	
I 1 1	
G 1 1	
G 3 4	
I 2 3	
G 1 1	
G 3 4	

Подсказка по решению

Это обычная HLD. Посмотрите конспект.

Дополнительные задачи

Задача 2С. Count Offline [2.5 sec, 256 mb]

Вам дано множество точек на плоскости.

Нужно уметь отвечать на два типа запросов:

- + x y — добавить в множество точку (x, y) .
- ? x_1 y_1 x_2 y_2 — сказать, сколько точек лежит в прямоугольнике $[x_1..x_2] \times [y_1..y_2]$.

Точки на границе и в углах тоже считаются. $x_1 \leq x_2$, $y_1 \leq y_2$.

Формат входных данных

Число точек N ($1 \leq N \leq 50\,000$). Далее N точек. Число запросов Q ($1 \leq Q \leq 100\,000$).

Далее Q запросов. Все координаты от 0 до 10^9 .

Формат выходных данных

Для каждого запроса GET одно целое число — количество точек внутри прямоугольника.

Пример

stdin	stdout
4	2
0 0	4
1 0	1
0 1	
1 1	
5	
? 0 1 1 2	
+ 1 2	
+ 2 2	
? 1 0 2 2	
? 0 0 0 0	

Подсказка по решению

Это относительно сложная задача. В любом случае нужно отсортировать точки по x и получить задачу на массиве. Далее два пути:

Или представить, что изменений нет, сделать дерево отрезков сортированных массивов, а чтобы учесть изменения навесить сверху корневую «отложенные запросы».

Или решать в offline – взять массив всех точек добавленных за всё время и на нём поддерживать дерево отрезков деревьев отрезка.

Кто знает и любит Фенвика – можно Фенвика.

Задача 2D. Union [1.5 sec, 256 mb]

Дано дерево из n вершин. Нужно обработать запросы вида
“количество рёбер на пути от v_i , до u_i , вес которых не более k_i ”.

Формат входных данных

На первой строке числе n ($1 \leq n \leq 10^5$). Следующие $n - 1$ строк описывают рёбра дерева. Рёбро задаётся парой концов a, b и весом w ($1 \leq a, b \leq n, a \neq b, 1 \leq w \leq 10^6$). Следующая строка содержит число запросов q ($1 \leq q \leq 10^5$). Каждый запрос задаётся тройкой чисел v_i, u_i и k_i ($1 \leq v, u \leq n, 1 \leq k \leq 10^6$).

Формат выходных данных

Для каждого запроса выведите одно число.

Примеры

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

Подсказка по решению

Все пути, как всегда, разбиваются на вертикальные.

Задачу нужно решать в offline.

Представьте, что обходите дерево dfs-ом... как ответить на запросы?

Кстати, если вы не писали до этого LCA-offline, самое время попробовать.

Задача 2E. Inspector is Coming [1.5 sec, 256 mb]

Дано дерево из N вершин. Изначально все рёбра не помечены. Поступают Q запросов вида “пометить все рёбра на пути из u в v , вес которых от W_{min} до W_{max} ”. Выведите число помеченных рёбер в конце процесса.

Формат входных данных

Вам даны число N ($2 \leq N \leq 100\,000$) — число вершин в дереве. Следующие $N - 1$ строка содержат описание рёбер дерева a_i, b_i и w_i ($1 \leq a_i, b_i \leq N, 1 \leq w_i \leq 1\,000\,000$). Далее следует число запросов Q ($0 \leq Q \leq 100\,000$). Следующие Q строк описывают запросы, каждая содержит четыре целых числа u, v, W_{min} , and W_{max} ($1 \leq u, v \leq N, 1 \leq W_{min} \leq W_{max} \leq 1\,000\,000$).

Формат выходных данных

Одно число – количество помеченных рёбер дерева.

Примеры

stdin	stdout
3 1 2 20 2 3 10 2 1 3 5 15 1 3 15 25	2
4 1 2 10 1 3 20 3 4 30 1 4 2 11 29	1

Замечание

Эту задачу нужно решать в offline.

Кстати, если не писали до этого LCA-offline, самое время попробовать.

Задача 2F. Distance Sum [5.0 sec, 256 mb]

На некоторой карте обозначены n городов и $n - 1$ дорога, соединяющая эти города таким образом, что полученный граф является деревом. Города занумерованы последовательными целыми числами от 1 до n .

Город 1 является корнем дерева; обозначим для каждого $i > 1$ город, являющийся предком города i , за p_i , а расстояние между городами p_i и i за d_i .

Snuke хочет для каждого $1 \leq k \leq n$ вычислить наименьшую сумму расстояний от некоторого города до городов $1, \dots, k$:

$$\min_{1 \leq v \leq n} \left\{ \sum_{i=1}^k \text{dist}(i, v) \right\}$$

Здесь $\text{dist}(u, v)$ обозначает расстояние между городами u и v .

Формат входных данных

Первая строка входа содержит одно целое число n ($1 \leq n \leq 2 \cdot 10^5$). Далее идут $n - 1$ строк, i -я из которых содержит два целых числа p_{i+1} и d_{i+1} — номер предка города $i + 1$ и расстояние между городом $i + 1$ и этим предком ($1 \leq p_i \leq n$, $1 \leq d_i \leq 2 \cdot 10^5$, p_i образуют дерево).

Формат выходных данных

Выведите n строк. В i -й из этих строк выведите ответ для $k = i$.

Примеры

stdin	stdout
10	0
4 1	3
1 1	3
3 1	4
3 1	5
5 1	7
6 1	10
6 1	13
8 1	16
4 1	19
15	0
1 3	3
12 5	9
5 2	13
12 1	14
7 5	21
5 1	22
6 1	29
12 1	31
11 1	37
12 4	41
1 1	41
5 5	47
10 4	56
1 2	59

Задача 2G. Прямоугольные запросы [6.5 sec, 256 mb]

Даны N точек на плоскости, у каждой точки есть ценность. Нужно быстро обрабатывать запросы двух типов:

- Присвоить всем точкам в области $[x_1..x_2] \times [y_1..y_2]$ ценность K .
- Найти точку с минимальной ценностью в области $[x_1..x_2] \times [y_1..y_2]$.

Формат входных данных

Вам даны число точек N ($1 \leq N \leq 262144$) и N точек.

Каждая точка задается тремя числами — x , y , начальная ценность.

Далее следует число запросов M ($1 \leq M \leq 10^4$) и M запросов в формате

“= $x_1 y_1 x_2 y_2 value$ ” для присваивания и “? $x_1 y_1 x_2 y_2$ ” для взятия минимума.

Все координаты от -10^9 до 10^9 . Все ценности от 0 до 10^9 .

Формат выходных данных

На каждый запрос ? выведите минимальную ценность точек в прямоугольнике.

Если в прямоугольнике нет ни одной точки, выведите NO.

Пример

stdin	stdout
4	2
1 1 1	1
-1 1 1	NO
-1 -1 1	0
1 -1 1	
7	
= 0 0 3 3 2	
= -3 -3 0 0 2	
? 0 0 3 3	
? -3 -3 3 3	
= -1 -1 1 1 0	
? 0 0 0 0	
? -1000 -1000 1000 1000	