Содержание

Вы не умеете читать/выводить данные, открывать файлы? Воспользуйтесь примерами.

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 компилятор `GNU C++11 5.1.0 (TDM-GCC-64) inc`, который позволяет пользоваться дополнительной библиотекой. Под ним можно сдать вот это.

Обязательные задачи

Задача 16A. 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 ($1 \leqslant N \leqslant 10^5 + 1$, $1 \leqslant M \leqslant 10^5$) — 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

Задача 16B. Пещеры и туннели [1 sec, 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 ($1 \leqslant N \leqslant 100\,000$) — количество пещер. Следующие $N-1$ строк описывают туннели. Каждая из этих строк содержит два целых числа — a_i и b_i ($1 \leqslant a_i, b_i \leqslant N$), описывающие туннель из пещеры с номером a_i в пещеру с номером b_i. Следующая строка содержит целое число Q ($1 \leqslant Q \leqslant 100\,000$), 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 ($1 \leqslant U \leqslant N$) увеличивается на V ($0 \leqslant V \leqslant 10\,000$). В случае запроса «`G`» ваша программа должна вывести максимальный уровень радиации на пути между пещерами с номерами U и V ($1 \leqslant U, V \leqslant N$) 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 2 2 3 2 4 6 I 1 1 G 1 1 G 3 4 I 2 3 G 1 1 G 3 4	1 0 1 3

Задача 16C. 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. $x_1 \leqslant x_2$, $y_1 \leqslant y_2$.

Формат входных данных

Число точек N ($1 \leqslant N \leqslant 50\,000$). Далее N точек. Число запросов Q ($1 \leqslant Q \leqslant 100\,000$). 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 0 0 1 0 0 1 1 1 5 ? 0 1 1 2 + 1 2 + 2 2 ? 1 0 2 2 ? 0 0 0 0	2 4 1

Дополнительные задачи

Задача 16D. 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 ($1 \leqslant n \leqslant 10^5$). Следующие $n-1$ строк описывают рёбра дерева. Ребро задаётся парой концов a, b и весом w ($1 \leqslant a, b \leqslant n$, $a \neq b$, $1 \leqslant w \leqslant 10^6$). Следующая строка содержит число запросов q ($1 \leqslant q \leqslant 10^5$). Каждый запрос задаётся тройкой чисел v_i, u_i и k_i ($1 \leqslant v, u \leqslant n$, $1 \leqslant k \leqslant 10^6$).

Формат выходных данных

Для каждого запроса выведите одно число.

Примеры

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

Задача 16E. 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 ($2 \leqslant N \leqslant 100\,000$) — число вершин в дереве. Следующие $N-1$ строка содержат описание рёбер дерева a_i, b_i и w_i ($1 \leqslant a_i, b_i \leqslant N$, $1 \leqslant w_i \leqslant 1\,000\,000$). Далее следует число запросов Q ($0 \leqslant Q \leqslant 100\,000$). Следующие Q строк описывают запросы, каждая содержит четыре целых числа u, v, W_{min}, and W_{max} ($1 \leqslant u, v \leqslant N$, $1 \leqslant W_{min} \leqslant W_{max} \leqslant 1\,000\,000$).

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 16F. 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 $1 \leqslant k \leqslant n$ вычислить наименьшую сумму расстояний от некоторого города до городов $1, \ldots, k$:

$$\min_{1 \leqslant v \leqslant n} \{\sum_{i=1}^{k} dist(i, v)\}$$

Здесь $dist(u, v)$ обозначает расстояние между городами u и v.

Формат входных данных

Первая строка входа содержит одно целое число n ($1 \leqslant n \leqslant 2 \cdot 10^5$). 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 4 1 1 1 3 1 3 1 5 1 6 1 6 1 8 1 4 1	0 3 3 4 5 7 10 13 16 19
15 1 3 12 5 5 2 12 1 7 5 5 1 6 1 12 1 11 1 12 4 1 1 5 5 10 4 1 2	0 3 9 13 14 21 22 29 31 37 41 41 47 56 59

Задача 16G. 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 ($1 \leqslant N \leqslant 262\,144$) и N точек.

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

Далее следует число запросов M ($1 \leqslant M \leqslant 10^4$) 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 1 1 1 -1 1 1 -1 -1 1 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	2 1 NO 0