

Задача А. Двоичное дерево поиска 1

Имя входного файла: `bst1.in`
 Имя выходного файла: `bst1.out`
 Ограничение по времени: 2 секунды
 Ограничение по памяти: 256 мегабайт

Реализуйте сбалансированное двоичное дерево поиска.

Формат входного файла

Входной файл содержит описание операций с деревом, их количество не превышает 100000. В каждой строке находится одна из следующих операций:

- `insert x` — добавить в дерево ключ x . Если ключ x уже в дереве, то ничего делать не надо.
- `delete x` — удалить из дерева ключ x . Если ключа x в дереве нет, то ничего делать не надо.
- `exists x` — если ключ x есть в дереве, выведите «`true`», иначе «`false`»

Все числа во входном файле целые и по модулю не превышают 10^9 .

Формат выходного файла

Выведите последовательно результат выполнения всех операций `exists`. Следуйте формату выходного файла из примера.

Примеры

bst1.in	bst1.out
<code>insert 2</code>	<code>true</code>
<code>insert 5</code>	<code>false</code>
<code>insert 3</code>	
<code>exists 2</code>	
<code>exists 4</code>	
<code>delete 5</code>	

Задача В. Двоичное дерево поиска 2

Имя входного файла: `bst2.in`
 Имя выходного файла: `bst2.out`
 Ограничение по времени: 2 секунды
 Ограничение по памяти: 256 мегабайт

Реализуйте сбалансированное двоичное дерево поиска.

Формат входного файла

Входной файл содержит описание операций с деревом, их количество не превышает 100000. Формат операций смотрите в предыдущей задаче. В каждой строке находится одна из следующих операций:

- `insert x` — добавить в дерево ключ x .

- `delete x` — удалить из дерева ключ x . Если ключа x в дереве нет, то ничего делать не надо.
- `exists x` — если ключ x есть в дереве, выведите «`true`», иначе «`false`»
- `next x` — выведите минимальный элемент в дереве, строго больший x , или «`none`», если такого нет.
- `prev x` — выведите максимальный элемент в дереве, строго меньший x , или «`none`», если такого нет.
- `kth k` — выведите k -ый по величине элемент (нумерация с единицы). Если такого не существует, то выведите «`none`».

Все числа во входном файле целые и по модулю не превышают 10^9 .

Формат выходного файла

Выведите последовательно результат выполнения всех операций `exists`, `next`, `prev`. Следуйте формату выходного файла из примера.

Примеры

bst2.in	bst2.out
<code>insert 2</code>	<code>true</code>
<code>insert 5</code>	<code>false</code>
<code>insert 3</code>	<code>5</code>
<code>exists 2</code>	<code>3</code>
<code>exists 4</code>	<code>none</code>
<code>next 4</code>	<code>3</code>
<code>prev 4</code>	<code>2</code>
<code>delete 5</code>	<code>none</code>
<code>next 4</code>	
<code>prev 4</code>	
<code>kth 1</code>	
<code>kth 3</code>	

Задача С. И снова сумма...

Имя входного файла: `sum.in`
 Имя выходного файла: `sum.out`
 Ограничение по времени: 3 секунды
 Ограничение по памяти: 64 мегабайта

Реализуйте структуру данных, которая поддерживает множество S целых чисел, с которым разрешается производить следующие операции:

- `add(i)` — добавить в множество S число i (если он там уже есть, то множество не меняется);
- `sum(l, r)` — вывести сумму всех элементов x из S , которые удовлетворяют неравенству $l \leq x \leq r$.

Формат входного файла

Исходно множество S пусто. Первая строка входного файла содержит n — количество операций ($1 \leq n \leq 300\,000$). Следующие n строк содержат операции. Каждая операция имеет вид либо «+ i », либо «? l r ». Операция «? l r » задаёт запрос $sum(l, r)$.

Если операция «+ i » идёт во входном файле в начале или после другой операции «+», то она задаёт операцию $add(i)$. Если же она идёт после запроса «?», и результат этого запроса был y , то выполняется операция $add((i + y) \bmod 10^9)$.

Во всех запросах и операциях добавления параметры лежат в интервале от 0 до 10^9 .

Формат выходного файла

Для каждого запроса выведите одно число — ответ на запрос.

Примеры

sum.in	sum.out
6	3
+ 1	7
+ 3	
+ 3	
? 2 4	
+ 1	
? 2 4	