Изучаем декартово дерево

Александр Курилкин, ШАД HELPER

1 сентября 2020 г.

Во всех задачах время работы может быть амортизированным или ожидаемым (то есть матожидание времени работы), если не указано иное.

Определение. Пусть дано двоичное дерево поиска, где каждой вершины присвоены значения x_i и y_i, где x — это ключи, а y — приоритеты. Декартовым деревом называется дерево, которое по ключам является двоичным деревом поиска (то есть в левом поддереве вершины v все ключи меньше, чем ключ v, а в правом больше), а по приоритетам является кучей (то есть в поддереве вершины v все приоритеты меньше, чем приоритет v).

№1

Докажите, что набор пар (x_i, y_i) однозначно задает декартово дерево, которое можно построить на таких парах.

Подсказка: выпишите пары в порядке возрастания ключей.

№2**

Докажите, что матожидание глубины вершины в декартовом дереве равно $O(\log n)$. Этого хватит, чтобы дать оценку $O(\log n)$ на базовые операции с декартовым деревом.

Подсказка: введите индикаторную функцию $I(u, v)$ — вершина u является предком v.

№3

Функция $split(v, x)$ разрезает декартово дерево с корнем в вершине v на два декартова дерева: в первом будут все ключи $< x$ из исходного, а во втором все ключи $\geqslant x$.

Функция $merge(v_1, v_2)$ склеивает в одно декартово дерево два декартова дерева с корнями в вершинах v_1 и v_2 таких, что во втором дереве все ключи больше, чем в первом.

Придумайте, как реализовать функции $split$ и $merge$ за $O(\log n)$.

Подсказка: используйте рекурсию.

№4

Функция $insert(v, x)$ вставляет ключ x в декартово дерево с корнем в вершине v.

Функция $delete(v, x)$ удаляет ключ x из декартова дерева с корнем в вершине v.

Придумайте, как реализовать функции $insert$ и $delete$ за $O(\log n)$

Подсказка: выразите их через $split$ и $merge$.

№5*

Функция $get\_kth(v, k)$ находит k-й по величине ключ в декартове дереве с корнем в вершине v.

Функция $get\_less(v, x)$ находит количество ключей $< x$ в декартовом дереве с корнем в вершине v.

Придумайте, как реализовать функции get_kth и get_less за $O(\log n)$.

Подсказка: храните размеры поддеревьев для каждой вершины.

№6**

Предложите структуру данных, которая по смыслу бы представляла из себя массив, и умела делать следующие операции за $O(\log n)$:

- узнать элемент на i-й позиции
- вставить элемент на i-ю позицию
- удалить элемент с i-й позиции

Например, после вставки в массив $[1, 2, 3]$ на первую (в 0-индексации) позицию элемента 4 массив выглядит как $[1, 4, 2, 3]$, а после удаления из массива $[1, 2, 3]$ элемента с первой позиции массив выглядит как $[1, 3]$.

Подсказка: храните массив в декартовом дереве без ключей. Найти i-й элемент можно аналогично функции get_kth.

№7**

Имея структуру данных из предыдущей задачи (кстати, называемую декартовым деревом по неявному ключу), придумайте как в ней реализовать запрос суммы на отрезке за $O(\log n)$. Если придумали, придумайте, как в дополнение к этому реализовать запрос прибавления на отрезке $O(\log n)$.

Подсказка: если вы дошли до этой задачи, сумма на отрезке не должна быть для вас проблемой. Прибавление на отрезке делается методом, схожим с тем, как делаются модификациями на отрезке в дереве отрезков.