Домашнее задание по АиСД №6

Эмиль Гарипов М3138

2019-10-30

Задача №1

Уже есть построенная двоичная куча на минимум, будем называть ее h_1, из n элементов. Требуется найти в куче k-й минимум. Для ответа на такой запрос восползуемся следующим алгоритмом :

> Шаг № 1. Создадим новую кучу, будем называть ее h_2, в ее вершинах будем хранить пару $(value, index)$, значение $index$ хранит номер вершины в исходной куче h_1, $value$ — значение этой вершины в исходной куче h_1. Добавим в h_2 корень кучи h_1.
>
> Шаг № 2. Детей корня кучи h_2 из кучи h_1 добавим в h_2 и удалим корень. Будем проделывать Шаг № 2 проделаем $k-1$ раз. Тогда в корне h_2 будет находится k-й минимум.

Рассмотрим работу алгоритма более подробно: изначально (после выполнения Шага № 1) куча h_2 будет хранить 1-й миниум. Затем мы добавим детей корня h_2 из кучи h_1 и удалим из h_2 корень (который был 1-м минимумом). И в корне нашей кучи h_2 будет храниться минимальный из всех элементов h_2. Таким образом, мы получим в корне второй минимум (первый минимум мы удалили и добавили минимальные значения,большие первого минимума).
Аналогичным образом куча h_2 будет хранить в своем корне k-й максимум после $k-1$ итерации Шага №2.

Этот алгоритм работает за $\mathcal{O}(k \log k)$, так как на каждой итерации Шага №2 в h_2 будет $\mathcal{O}(k)$ вершин в куче, а добавление (добавлений мы делаем не более 2-х, но это не повлияет на асимптотику работы, так как это константа) и удаление в двоичной куче выполняется за $\mathcal{O}(n)$.

Задача №2

Для реализации операции “changeKeys h x” дополнительно будем в каждой вершине u хранить p_u — сколько нужно добавить к значению этой вершины val_u, чтобы значение было актуальным после прибавлений операцией “changeKeys” на каком-то поддереве, которое содержит вершину u. Поддержим следующий инвариант: при посещении вершины в ней записано актуально значение. Для этого при каждом посещении вершины u будем увеличивать (или уменьшать в случае если $p_u < 0$) val_u на значение p_u и увеличивать значение p_v на p_u $\forall v : v$ — ребенок u. Так же необходимо обнулить p_u, так как значение val_u уже актуально. Назовем эту операцию “push”.

Тогда для выполнения “changeKeys h x” нужно просто увеличить p_h на x (выполняется за $\mathcal{O}(1)$). А во все остальные операции на левосторонней куче просто нужно добавить вначало операцию push(в операции merge необходимо делать push в обе входные вершины). Время работы не поменялось ни у одной из операций, так как push выполняется за $\mathcal{O}(1)$.

Задача №3

Для начала найдем медиану входного массива алгоритмом нахождения медианы за $\mathcal{O}(n)$. Затем условно разделим массив на две части a и b, так, что все элементы a больше медианы m (для краткости назовем медиану m), а все элементы b меньше либо равны med, так что в a было либо столько же элементов, либо на один больше(Если элементов, равных медиане несколько, надо будет немного по другому разбить на две части, но этот случай тривиален и основной алгоритм от рассмотрения этого случая не измеится). Алгоритмом построения кучи за линейное время построим две кучи, первую на элементах a, эту кучу назовем h_l, вторую на элементах b, эту кучу назовем h_r. Куча h_l будет на минимум (в корне хранится минимум кучи, для каждой вершины верно что она небольше своих потомков), h_r — на максимум (в корне хранится максимум кучи, для каждой вершины верно, что она неменьше своих потомков). Будем поддерживать инвариант, что в левой на не более чем на один элемент больше.
Разберем работу операций:

1. Балансировка куч: На самом деле такого запроса не предусмотрено условием задачи, но он нужен для реализиации этой струкрутры. Пока в левой куче меньше элементов будем извлекать корень из правой кучи и закидывать в левую. Это будет работать за $\mathcal{O}(\log n)$, так как мы будем перекидывать мало элементов.

2. insert: Будем добавлять элементы в левую кучу, а затем балансировать h_l и h_r.

3. medianElement: вернем максимум в h_r, это и есть медианный элемент.

4. deleteMediad: Удалим корень из правой кучи и вернем его как удаленный элемент.

   Таким образом все операции работают за $\mathcal{O}(\log n)$, построение за $\mathcal{O}(n)$.