Алгоритмы и Структуры Данных ДЗ-6

Гарипов Роман М3138

03.11.2019

Задача №1

Куча с k-ым минимумом за $\mathcal{O}(klog(k))$

Как устроена такая куча

Рассмотрим момент, когда приходит запрос взять k-ый минимум. Сначала закинем корень во вторую(будем называть ее буферная) кучу. Теперь в буферной куче лежит 1-ый минимум. Удалим эту вершину, добавим ее детей из первой кучи. Теперь в буферной лежит 2-ой и 3-ий минимумы. И так далее.

Более формально

Как базу нашей структуры данных добавим 1-ый минимум - это вершина исходной кучи. Потом будем добавлять/удалять вершины из буферной кучи. пока не получим k-ый минимум. Пусть у нас есть $i-1$ минимум. Удалим его, добавим его детей, выберем минимум в куче, это и будет i-ый минимум. И так проделаем до k. Вернём k-ый минимум - вершину буферной кучи.
Ещё вместе с каждый элементом, нужно хранить его индекс в первой куче, чтобы уметь получать индексы и значения его детей.

Ассимптотика

После того как запомнили k-ый минимум, все оставшиеся элементы можно удалить, это не повлияет на ассимптотику, так как мы добавили k элементов и удалили k элементов, каждая из операций за $\mathcal{O}(log(k))$, что в сумме даёт $\mathcal{O}(klog(k))$.
$\log(k)$ - потому что в буферной куче в любой момент времени $\mathcal{O}(k)$ элементов.

Задача №2

Левосторонняя куча с операцией увеличения всех ключей в куче h на заданное значение x

Как устроена такая куча

Воспользуемся методом проталкиваний. Если приходит запрос на увеличение, запишем в корень конкретной кучи, что для всего поддерева корня надо увеличить значения на x. Когда мы в следующий раз обратимся в корень, перед тем как выполнять какие-то действия, обновим значение корня, увеличив его на x. После этого надо будет записать в детей этой вершины, что потом в них нужно будет увеличить значение на x.

Более формально

Будем в каждой вершине v хранить одну переменную $p[v]$. Пусть изначально все $p[i] = 0$, ничего не нужно прибавлять. А когда придет запрос увеличения значений в куче h на x запишем $p[v] = x$ для корня v кучи h. В начале каждой функции, где мы работает с вершиной или парой вершин, будем обновлять информацию о вершине примерно таким кодом :

```
push(v) :
    val[v] += p[v]
    p[2 * v + 1] += p[v]  // важным моментом является то,
    p[2 * v + 2] += p[v]  // что нужно делать += p[v],
    p[v] = 0 // чтобы запросы могли "накладываться"
```

Итого

Таким образом, в начале каждой операции в обрабатываемой вершине будет актуальная информация. Ассимптотика не изменится ни для какого из методов нашей структуры данных, так как $push(v)$ работает за $\mathcal{O}(1)$, так же все методы будут работать корректно.

Задача №3

Куча с медианой

Как устроена такая куча

Заведём фиктивную вершину - корень нашей кучи. Пусть его левый ребёнок - куча, в которой каждый отец больше (либо равен) своего ребёнка, а правый ребёнок - обычная куча, в которой каждый отец меньше (либо равен) своего ребёнка.
В каждую из них положим поровну элементов, причем так, чтобы в правой все были меньше либо равны медианного элемента, а в левой - больше либо равны, если кол-во элементов n не делится на 2 то положим в левую кучу на один элемент больше.
Тогда, чтобы получить медианный элемент, надо взять максимум в левой куче. Это действительно будет медианный элемент, так как в левой куче $\frac{n}{2}$ чисел, каждое из которых $\leq$ медианы. Понятно, что максимальное из них и будет медианой.

Балансировка

Теперь важно после каждой операции добавления/удаления балансировать левую и правую кучи, чтобы разность их размеров была не больше 1, а именно чтобы в правой было на 1 элемент меньше.
Балансировать достаточно просто, надо либо удалить корень левой кучи и

добавить в правую - и так несколько раз пока размеры не сбалансируются, либо удалить корень правой и добавить в левую, тоже повторяя этот процесс пока размеры не сбалансируются.

Операции add и delete

Тогда, чтобы добавить элемент - добавим его в любую из куч, потом сбалансируем их. Так же с удалением - удалим элемент из той кучи, в которой он лежит(это можно понять по корням куч, если удаляемый элемент x меньше корня правой, значит он лежит в левой и т.д.) и сбалансируем их.

Построение за линейное время

А для того чтобы построить такую структуру данных за $\mathcal{O}(n)$, найдем медиану за $\mathcal{O}(n)$ алгоритмом нахождения k-ой порядковой статистики, который разбивает массив на блоки размером по 5 для $k = \frac{n}{2}$ и запишем это значение в переменную $median$. Теперь построим правую кучу за $\mathcal{O}(\frac{n}{2})$ для первых $\frac{n}{2}$ элементов $\leq median$ и левую кучу за такую же ассимптотику для всех остальных. Этот алгоритм был рассмотрен на практике. Таким образом, мы смогли построить нашу структуру данных за линейное время.