Домашнее задание по АиСД №4

Эмиль Гарипов M3138

2019-10-14

Задача №1

Для поиска позиции p будем использовать следующий алгоритм:
На i-м шаге будем сравнивать число x с числом a_{2^i}. До тех пор, пока верно $x > 2^i$, увеличиваем i. Тогда мы остановимся, когда $a_{2^i} \leq x$, причем все элементы на префиксе длиной 2^{i-1} меньше x, поскольку массив отсортирован. Тогда бинарным поиском попытаемся найти x на отрезке $[2^{i-1}; 2^i]$. Элемент x, если сущесвтует в массиве a, то находится именно в этом отрезке, так как $a_{2^{i-1}} < x \leq a_{2^i}$ и массив a отсортирован.

Посчитаем время работы алгоритма: пусть p — позиция x в массиве a, тогда поиск отрезка, в котором находится элемент x выполнится не более чем за $\log p$ шагов, так как поиск перебирает позиции, равные степеням двойки, которые не превосходят p, и сравнивает значения в них с x. Степеней двоек не больших чем x ровно $\lfloor \log p \rfloor$.
Бин поиск выполнит не более $\log p$ шагов, поскольку длина отрезка, на котором выполняется поиск $2^{\log p}$.

Итого, алгоритм работает за $\mathcal{O}(\log p)$.

Задача №2

Воспользуемся следующим алгоритмом:
Будем поддерживать три указателя, L, R_1 и R_2. L указывает на начало отрезков таких, что количество различных чисел на них ровно k, заканчивающихся в $R \in [R1; R2]$. Так же необходимо хранить массив a — массив подсчета, i-я ячейка которого будет хранить кол-во числа i для $\forall i$ на отрезках, описанных выше. Чтобы узнать кол-во различных чисел на отрезке, надо посчитать кол-во не нулей в этом массиве (подробнее об этом ниже).

> Шаг № 1. Если мы узнаем такие L, R_1 и R_2, то надо добавить к ответу $R_2 - R_1 + 1$, так как на них ровно k различных чисел, уменьшать $a_{val[l]}$и прибавлять к ответу $R_2 - R_1 + 1$, сдвигая L на один вправо до тех пор, пока кол-во различных чисел на отрезках $[L; R]$ ($R \in [R_1; R_2]$), не станет равно k (о том, как это посчитать сказано ниже), где $val[l]$ — значение исходного массива в позиции l.
>
> Шаг №2. После сдвига левой границы надо найти такие R_1, R_2, что кол-во различных чисел на них было ровно k. Для этого сдвинем R_1 и R_2 в $R_2 + 1$ и будем увеличивать R_2, одновременно увеличивая значения $a_{val[R_2]}$ до тех пор, пока кол-во не нулей в массиве a не будет равно k (в самом начале работы алгоритма будет аналогичным способом искать L, R_1, R_2).
>
> После этого будем проделывать алгоритм начиная с Шага №1, до тех пор пока не дойдем до конца массива.

Так как мы добавляем и удаляем по одному элементу из отрезка, то для подсчета кол-ва не нулей в массиве просто будем хранить переменную, хранящую это кол-во. Увеличивать ее будем при добавлении элемента x, если $a_x = 0$, уменьшать при удалении элемента x, если $a_x = 1$(до уменьшения a_x).

Таким образом алгоритм для каждой позиции L посчитает кол-во отрезков, удовлетворяющих условию за время $\mathcal{O}(n)$.