АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА
МЕДИАНИ

Ако за всяка операция „G” сортираме последователността и търсим средния елемент, то тогава изпълнението на програмата би било твърде бавно тъй като можем да имаме над 500000 операции „G”. Най-грубо сложността би била O(n^2*logn). Затова решението е да използваме структура, която пази елементите в сортиран вид. Такава структура е двоично дърво за търсене, при което левият наследник е по-малък или равен на родителя, а десният по-голям от родителя. Нещо повече за да можем да намерим i-я елемент в сортираната последователност ефективно за всеки връх в дървото пазим и броя елементи в поддървото, чийто корен е този връх.

Търсенето на i-я елемент в така конструираната структура става по следния начин.

1. Започваме търсенето от корена
2. Ако имаме ляво поддърво и
    a. Броят елементи в лявото поддърво е i, то търсеният елемент е коренът
    b. Броят на елементите в лявото поддърво е > i, то за да намерим търсеният елемент, прилагаме същият алгоритъм върху лявото поддърво
    c. Броят елементи в лявото поддърво е < i, то за да намерим търсеният елемент прилагаме същият алгоритъм върху дясното поддърво, но този път търсим индекс (i – leftSubTreeChildrenCount - 1)
3. Ако не сме намерили стойности в 2 и имаме дясно поддърво тогава:
    a. Ако i = 0, то търсената стойност е коренът
    b. Ако i > 0, то търсената стойност може да се намери като се приложи алгоритъмът за индекс (i-1) в дясното поддърво.
4. Ако не е открита стойността в 2 и 3, то връщаме коренът.

Както се вижда този подход би дал доста добро бързодействие ако имаме малка дълбочина на двоичното дърво, ето защо трябва да си осигурим, че то е балансирано. За целта реализираме дървото като Red Black Tree, което гарантира сложност на операциите добавяне и търсене O(logn). Трябва да се забележи, че не е необходимо да се имплементира изтриване в дървото тъй като нямаме такава операция в задачата.

По този начин общата сложност става O(nlogn).

Важно: Тъй като операциите за вход изход са много, използвайте *scanf* и *printf* вместо *cin* и *cout*. Разликата в бързодейсвието е огромна и няма да влезе решението във времето с бавно четене и писане.

Автор: Тодор Петров