**Задача D-query, spoj**

<https://www.spoj.com/problems/DQUERY/>

При даден масив А с дължина N и Q интервала [l, r], за всеки интервал да се намери броят на различните числа между А[l] и А[r].

Ограничения N <= 30000, Q <= 200000, A[i] <= 1000000

Най-простото решение е да се обиколят елементите от а[l] до a[r] и да се преброят различните, но това би било твърде бавно. Алгоритъмът на Мо се използва точно в такива случаи, когато не можем да отговорим на заявката за [l, r] за O(1), но знаейки отговора за интервала [l,r] можем да направим преход към [l ± 1, r] или [l, r ± 1] и да намерим отговорите за интервалите за О(1).

Алгоритъмът се състои в това да сортираме заявките [l, r] по подходящ начин така, че от отговора за [l1, r1] да намерим отговора за [l2, r2], от [l2, r2] за [l3, r3] и т.н. И това да стане с възможно най-малко преходи. За целта разделяме масива А на B блока, всеки с дължина √N. Първо всички интервали, чиито ляв край влиза в първия блок, ги сортираме по десния край, после всички интервали с ляв край във втория блок, ги сортираме по десния им край и т.н.

bool sortfunc (Query q1, Query q2) {

if(q1.l / block\_size == q2.l / block\_size) return q1.r > q2.r;

return q1.l < q2.l;

}

Сега остава да се измисли как да се направи прехода от текущия интервал [l, r] към [l ± 1, r] или [l, r ± 1] . Това може да стане като в масив count[x] пазим колко пъти се среща числото x в текущия интервал. Когато добавяме A[i] към интервала, ако count[A[i]] = 0, увеличаваме отговора с 1. Аналогично, когато трием А[i] от интервала, aко count[A[i]] е станало 0, намаляваме отговора с 1.

В този алгоритъм има два проблема - А[i] трябва да е по-малко от 106, тъй като не можем да поберем 109 елемента в масива и ако ограничението за А[i] беше различно, би се наложило да компресираме масива. Вторият проблем е, че при всеки преход искаме достъп до стойност в count и променяме тази стойност, което води до голяма константа и като се събмитне задачата дава time limit.

Вместо count, ще използваме два масива pre[] и nxt[] – в pre[i] се пази позицията на най-близкия предишен елемент със същата стойност като А[i]. Съответно в nxt[i] се пази позицията на най-близкия следващ елемент със същата стойност като А[i].

[l,r]→[l,r+1] добавяме 1 към отговора, ако стойността А[r+1] не се среща в [l, r], т.е. pre[r+1] < l

 [l,r]→[l,r−1] изваждаме 1 от отговора, ако pre[r] < l

[l,r]→[l−1,r] добавяме 1 към отговора, ако nxt[l-1] > r

 [l,r]→[l+1,r] изваждаме 1 от отговора, ако nxt[l] > r

**Задача Closest Equals, codeforces 522D**

<https://codeforces.com/contest/522/problem/D>

При даден масив А с дължина N и Q интервала [l, r], за всеки интервал търсим минималното разстояние между два еднакви елемента. Търсим минималната стойност |x - y| за l ≤ x,y ≤ r и а[x] = a[y], x ≠ y.

Решение <https://codeforces.com/blog/entry/16853>

Още материали:

<https://codeforces.com/blog/entry/83248>

<https://codeforces.com/blog/entry/43230> (Алгоритъм на Мо в дърво)

Задачи:

<https://codeforces.com/contest/86/problem/D> (подобна на задача D-query)

<https://www.spoj.com/problems/COT2/>

<https://www.codechef.com/ALKH2016/problems/VLB> (задачи с Алгоритъм на Мо в дърво)