**D-Query / 1188 - Fast Queries – Solution Approach Using Segment Tree**

**1.** [**http://www.spoj.com/problems/DQUERY/**](http://www.spoj.com/problems/DQUERY/) **2.** [**http://lightoj.com/volume\_showproblem.php?problem=1188**](http://lightoj.com/volume_showproblem.php?problem=1188)

This is an interesting problem to solve using Segment Tree. The complexity as like as standard segment tree.

So, Let’s begin the solution approach. Let, an array with 8 elements (Like Light OJ),

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Value | 1 | 1 | 1 | 2 | 3 | 5 | 1 | 2 |

To solve this, we should have a slight knowledge at Offline Query. So, what is an **offline Query**? Is it hardest method? No! I know, you know about Query. Right? Hmm. So, what is a query? A query is a search to you to give a solution. It may be a range or may not be a range (single element). So, Query can be two type. One and the mostly know is **online Query** and another is **Offline Query**.  
An Online Query is that for which we reply or return value at the time of query. And, the other is we can store the queries and return the values as their sequence. So, is it hardest to do? No, obviously!

So, if we mix the original array with the query, is it hardest to recognize which is array and which is query? No! How we mix!! Make the array element’s as their key and queries right range as the key of query. And sort the mixed ascendingly with their key. Now, you are ready to solve the problem. How!!

Let, the query and array element like as, Light OJ. (**1188 - Fast Queries)**

8 5

1 1 1 2 3 5 1 2

1 8

2 3

3 6

4 5

4 8   
Which has 8 elements and 5 queries. Let’s make a for the mix. ☺

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Key | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 8 | 3 | 6 | 5 | 8 |
| Left Range |  |  |  |  |  |  |  |  | 1 | 2 | 3 | 4 | 4 |
| Index of Query |  |  |  |  |  |  |  |  | 1 | 2 | 3 | 4 | 5 |
| Value of array | 1 | 1 | 1 | 2 | 3 | 5 | 1 | 2 | 0 | 0 | 0 | 0 | 0 |
| Flag for Query | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

Now, sort it increasingly as their Key but descending with their value. I think you can do that. Do it as your assignment. ☺

Now, see the process,

When I get an array element, check it was in before or not right? You can use map for that with constant time. If not then its resultant value is 1, else 0. And when a query do a query for range of query (Left, Key). So, what is the benefit here of the offline query. When I make a value of before to zero, it’s no needed up to the Key (Right). Look the process below:

First, we will be at the Key 1 (sorted), Array Element is 1, it was not in past. So, 1

Key 2, Array element will also 1, Was in past, so, resultant value for the past is one and, Key 2 is 1,  
Key 3, Array element will 1, In past, so past same element will be 0, and Key 3 is 1,

All those will be do update and query in segment tree at the Key as node. When a query, do the query and save the result with the queries original index. So, on…

So, after all the update and Query,

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Key for array | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Array Element | 1 | 1 | 1 | 2 | 3 | 5 | 1 | 2 |
| Resultant donation to Node of Tree | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

Happy Coding!! ☺