This problem can be efficiently solved by Segment tree.

Since it is stated that, the input will be strictly in increasing order, so a natural technique can be applied.   
  
For this we require 3 arrays :  
1. array **input** store the input numbers,  
2. array **count**  store the frequency of each input numbers,  
3. and array **start** store the index of the input list where a particular input number appeared first.  
  
For example ,  
input = { -1, -1, 1, 1, 1, 1, 3, 10, 10, 10}  
count = { 2, 2, 4, 4, 4, 1, 3, 3, 3 }  
start = { 1, 1, 3, 3, 3, 7, 8, 8, 8 }  
  
At first, a segment tree is constructed where each node will store the value of the maximum count (from the **count** array) of its respective range [ a,b ].

Let, the query range be [ i,j ].

Now 2 cases can occur : The value at index i and j are -  
1. same i.e input[ i ] = input[ j ].  
2. different i.e input[ i ] ≠ input[ j ].  
  
**#Case 1:**  
Solving this case is the easiest. Since input[ i ] = input[ j ] , all the numbers in the range [ i,j ] are same ( since the numbers are non-descending ). So the answer for this **case 1** is **j - i + 1**.  
  
**#Case 2:**  
In this case, there exists an index **x** where input[ i ] = input[ x ] and input[ i ] ≠ input[ x + 1 ]. Let, **k** = x + 1. So, the value of **k** = start [ i ] + count [ i ] .  
So, the frequency of the value input[ i ] in the range [ i,k ] is **cnt1** = k – i .

The frequency of input[ j ] in the range [ i,j ] is **cnt2** = j –start [ j ] + 1 .  
  
The maximum frequency of the values in range [ i,j ] may also exists in the range [ k , start[ j ] - 1 ]. This can be found by querying the segment tree for the maximum value in the range [ k , start[ j ] - 1 ]. Let the maximum count returned by the query be **cnt3** .  
  
Therefore the answer for **case 2** is **max ( cnt1, cnt2 , cnt3 ) .**