H.W: ① Find count of # of subarray with # of distinct ele == k.

② Find sum of length of all subarray with # of distinct element ≥ k.

① Hint :-

Let F (arr, k) : No. of subarray with ≤ k distinct elements.

F (arr, k-1) : No. of subarrays with ≤(k-1) distinct elements.

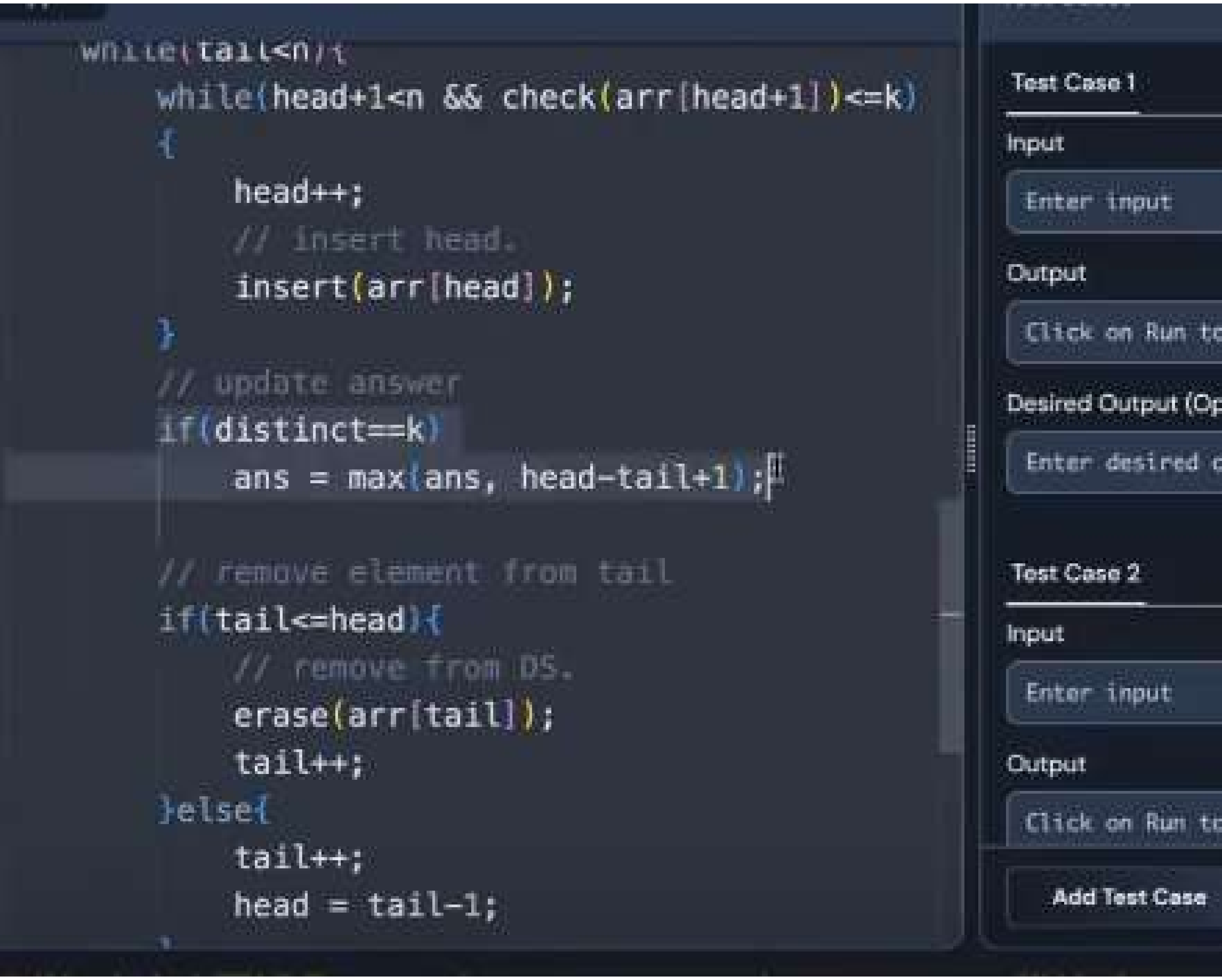
Count of subarrays with exactly K distinct elements

⇒ F(arr, k) - F(arr, k-1)

Variation : Find the max length of a subarray with exactly K distinct elements.

→ similar to Type 1 Form of pointer q2.

→ Before taking max length, check if freq == distinct.



```
while(tail<n){
    while(head+1<n && check(arr[head+1])<=k)
    {
        head++;
        // insert head.
        insert(arr[head]);
    }
    // update answer
    if(distinct==k)
        ans = max(ans, head-tail+1);

    // remove element from tail
    if(tail<=head){
        // remove from DS.
        erase(arr[tail]);
        tail++;
    }else{
        tail++;
        head = tail-1;
    }
```

② If no. of distinct elements > k → holds for subarray (L, R)

Then, it does not necessarily hold for (L+1, R).

Approach :-

1. Partition the array using two pointers.
2. (n-1-head) = No. of subarrays ending after head
3. Count subarrays with ≤ k distinct & subtract from total.

Part 2 : Find the sum of length of these subarrays.

$$len = tail - head + 1$$

$$ans \mathrel{+}= \frac{len(len+1)}{2}$$

Form 2: Two pointers with opp. direction

- 2sum
- 3sum

Q1: Find max value of $[\overbrace{\min(arr[i], arr[j])}^{min} * \overbrace{(j-i)}^{len\ (gap)}]$

$N = 7$

arr =

3	5	8	3	2	7	1

$\boxed{0 \le i < j \le n}$

$N \le 10^5$

$arr[i] \le 10^9$

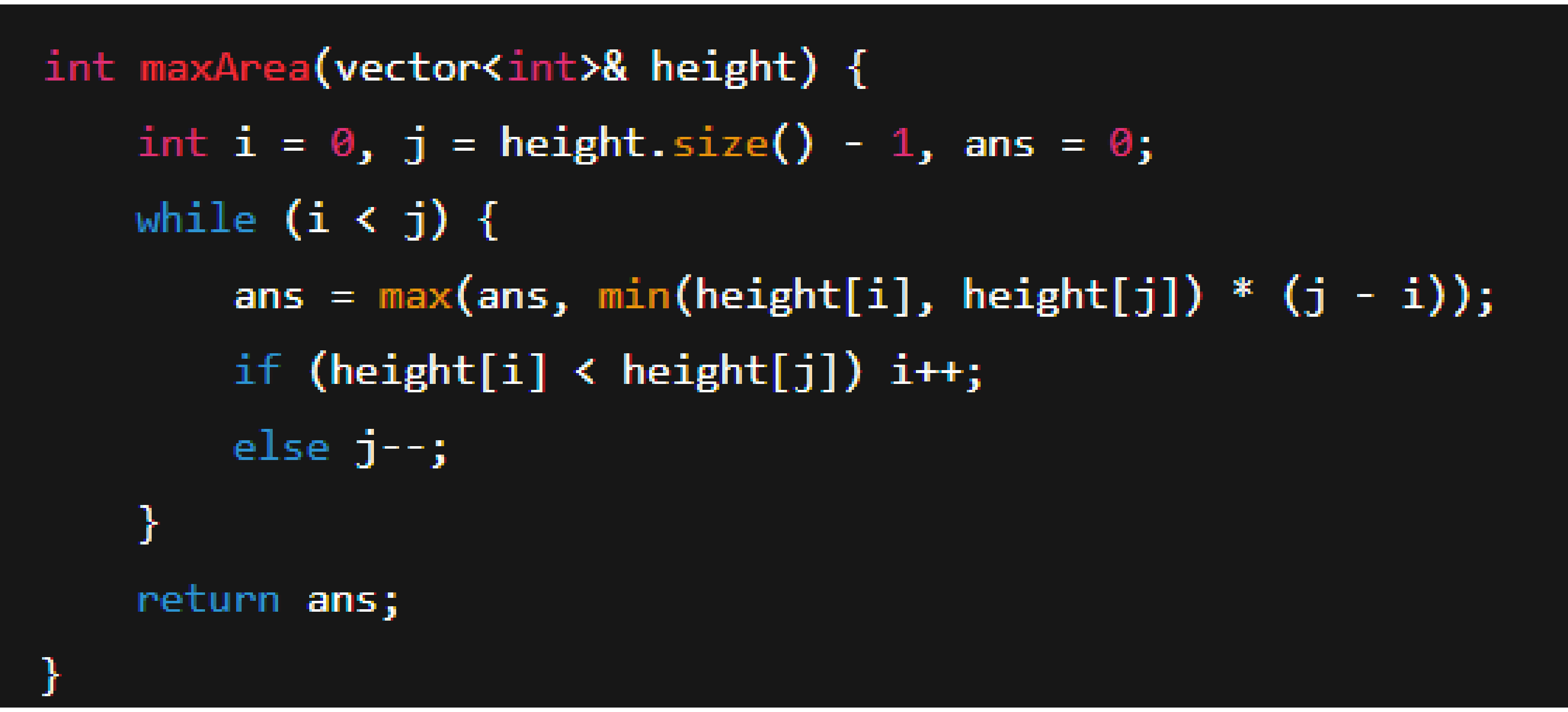
Logic :-

1) start i at the left end, j at the right end.

2) While length dec, inc. min to get better result.

3) Move pointer with the smaller value.

Leetcode Problem NO : 11 → similar



```
int maxArea(vector<int>& height) {
    int i = 0, j = height.size() - 1, ans = 0;
    while (i < j) {
        ans = max(ans, min(height[i], height[j]) * (j - i));
        if (height[i] < height[j]) i++;
        else j--;
    }
    return ans;
}
```

Form 3 : Is S a subsequence of T ?

Exa : T = a c d b a c d a b (size = N)

S = c b a d b (size = M)

Approach :-

Two pointers - one for each sequence

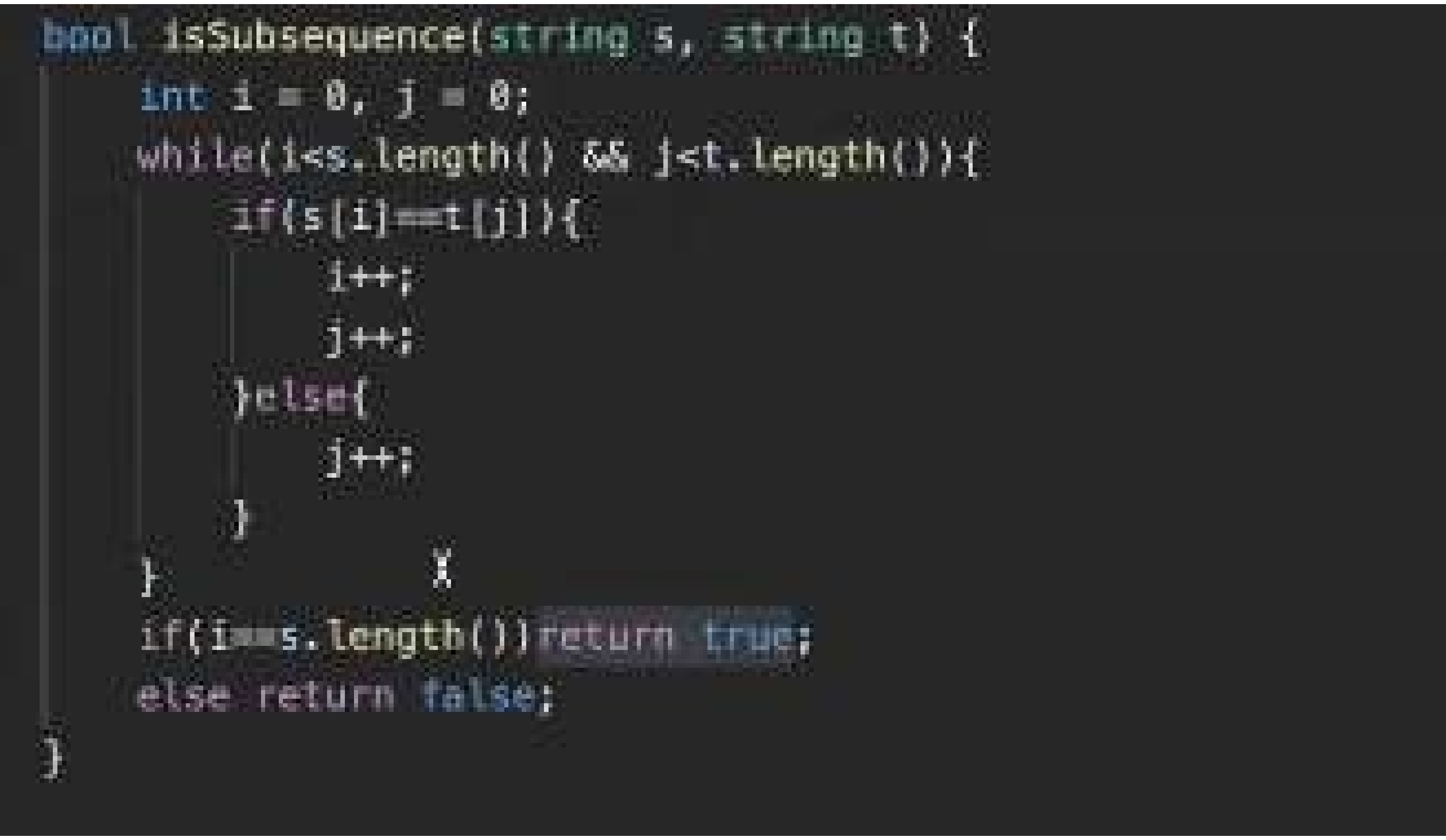
- If match - move both pointers.
- If mismatch - move pointer in T only.

T.C : O(n+m)

Other Applications of Form 3 :-

1. Merge Sort
→ compare smallest elements from two lists and pick smaller one.

2. Intersection of 2 sorted lists
→ If same, add to ans and move both.
→ If different, move ptr with smaller value.



```
bool isSubsequence(string s, string t) {
    int i = 0, j = 0;
    while(i<s.length() && j<t.length()){
        if(s[i]==t[j]){
            i++;
            j++;
        }else{
            j++;
        }
    }
    if(i==s.length())return true;
    else return false;
}
```