arr →

→ 5 6 9 9 . . . .

Brute force

```
for ( i=0 ; i<n ; i++) {
    for (j=i+1 ; j<n ;j++) {
        if (a[j] > a[i]) {
            output [i] = a[j] ;
            break;
        }
    }
}
```

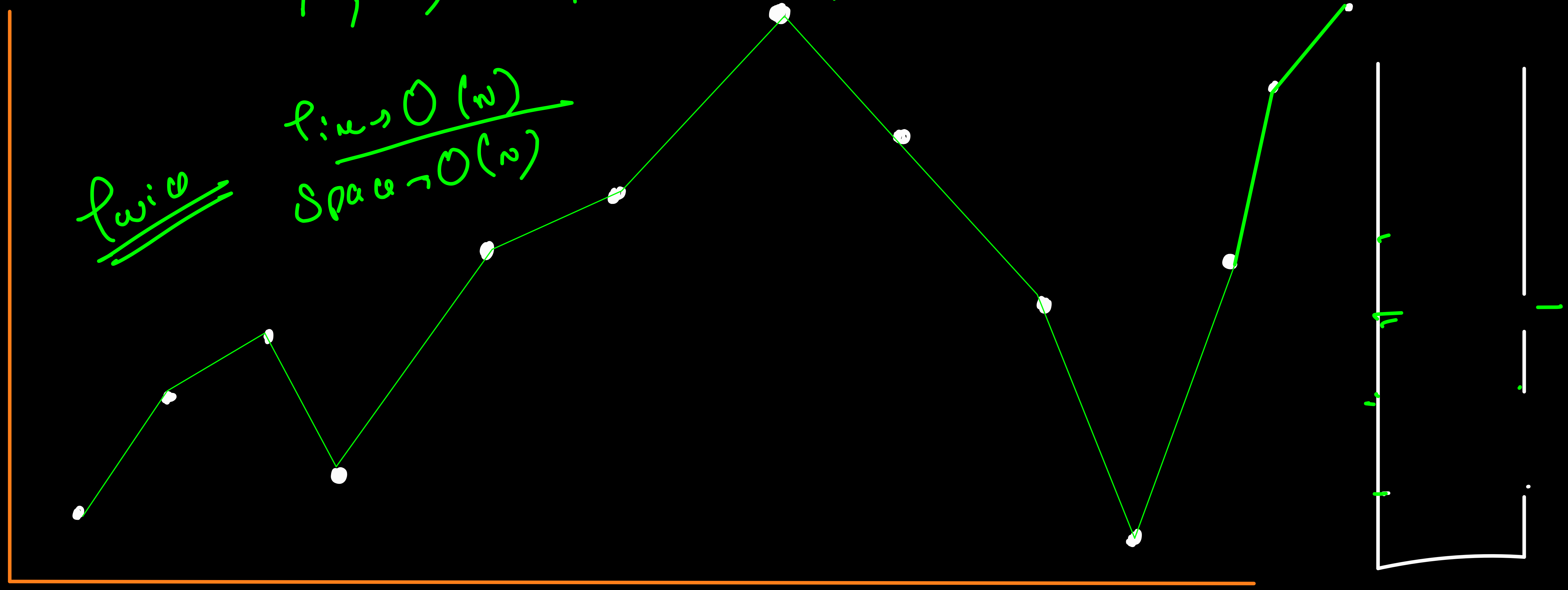
time $O(n^2)$

Space → $O(1)$

2

[2, 5, 6, 3, 9, 12, 20, 13, 7, 1, 8, 16, 19]

oup → [5, 6, 9, 9, 12, 20, -1, 16, 8, 8, 16, 19, -1]



time → O(n)

Space → O(n)

twice

Keep track of last visited elements

↓

Stacks

x → inc cur

x will be definitely the ans of just prev element first element to the right of curr elemt greater than curr element.

for inc curr → a[i+1] > a[i] so a[i+1] is the ans for a[i] → next elemt is the NGE

↓

But we can't say that for sure for a dec curv.

after the dip of dec curve, when we get an inc curve the elements of inc curve might help us to find ans.

We will store the elements of the array because their ans will be coming later.

↳ we should store indexes

index → element

element $\nrightarrow$ index

Qn Prev greater element ??

-1	5	3	-1	6	6
5	3	1	6	2	4

↓ reverse array → O(n)

4 2 6 1 3 5

6 6 -1 3 5 -1 ← nge → O(n)

↓ reverse the output → O(n)

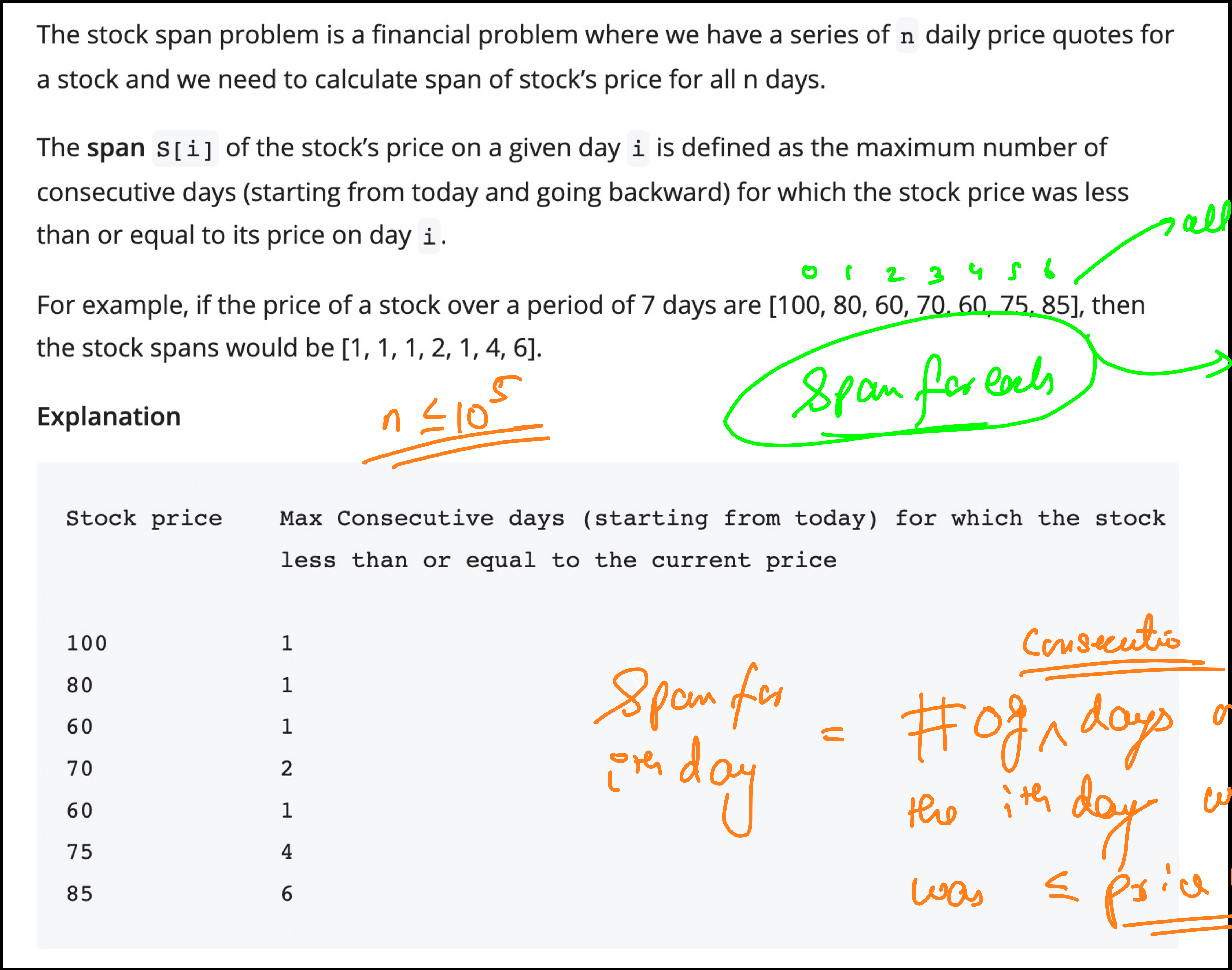
-1 5 3 -1 6 6 → T.C → O(n)

Q.) Next Smaller element → O(n)

```
for ( i=1 ; i<n ; i++) {
    while ( !st.isEmpty() and arr[i] < arr[st.top()])
        output[st.top()] = arr[i]
        st.pop()
    }
    st.push(i)
}
```

Qn prev smaller element → $O(n)$

Stock Span



The stock span problem is a financial problem where we have a series of `n` daily price quotes for a stock and we need to calculate span of stock's price for all n days.

The **span** `S[i]` of the stock's price on a given day `i` is defined as the maximum number of consecutive days (starting from today and going backward) for which the stock price was less than or equal to its price on day `i`.

0 1 2 3 4 5 6

all the prices is of one stock

For example, if the price of a stock over a period of 7 days are [100, 80, 60, 70, 60, 75, 85], then the stock spans would be [1, 1, 1, 2, 1, 4, 6].

Span for each

there are prices for n days

$n \leq 10^5$

Explanation

Stock price	Max Consecutive days (starting from today) for which the stock less than or equal to the current price
100	1
80	1
60	1
70	2
60	1
75	4
85	6

Span for i^{th} day = # of consecutive days on or before the i^{th} day where price was ≤ price[i] nge

reverse

reverse output

output

100, 80, 60, 70, 60, 75, 85

1 1 1

span → index − index pge

O(n)

O(n)

span → 1 1 1 2 1 4 6

index of pge → -1 -1 -1 2 -1 1 0

100, 80, 60, 70, 60, 75, 85

index → 0 1 2 3 4 5 6

O(n)

for any ith day if we calc pge (first elmt to left of i having value > arr[i])

$n \leq 10^5$

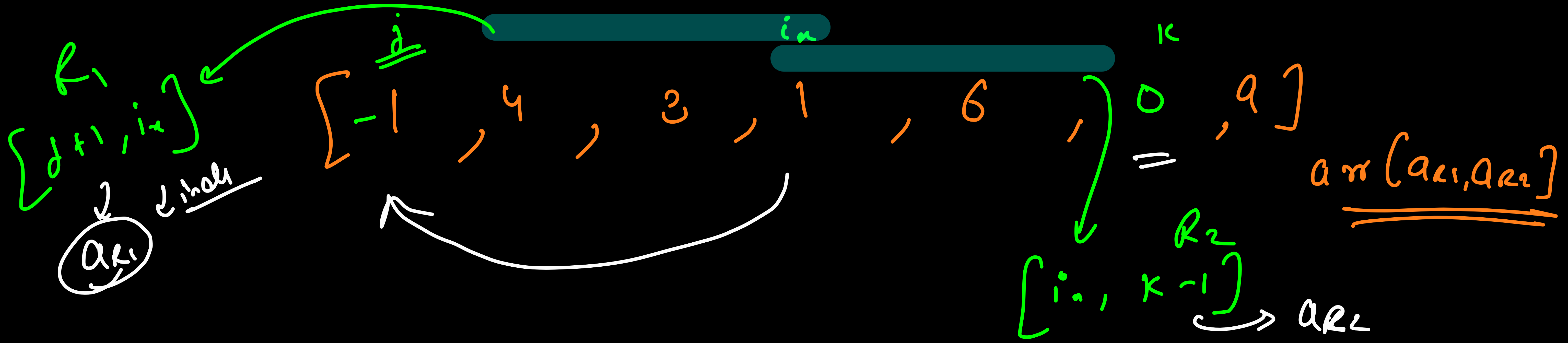
$[1, 4, 3, 1, 6, 2, 9]_n$

Brute force → to calc all possible subarrays

$O(n^2)$

one single element can be minimum of multiple subarrays

for how many subarrays 1 will be the min



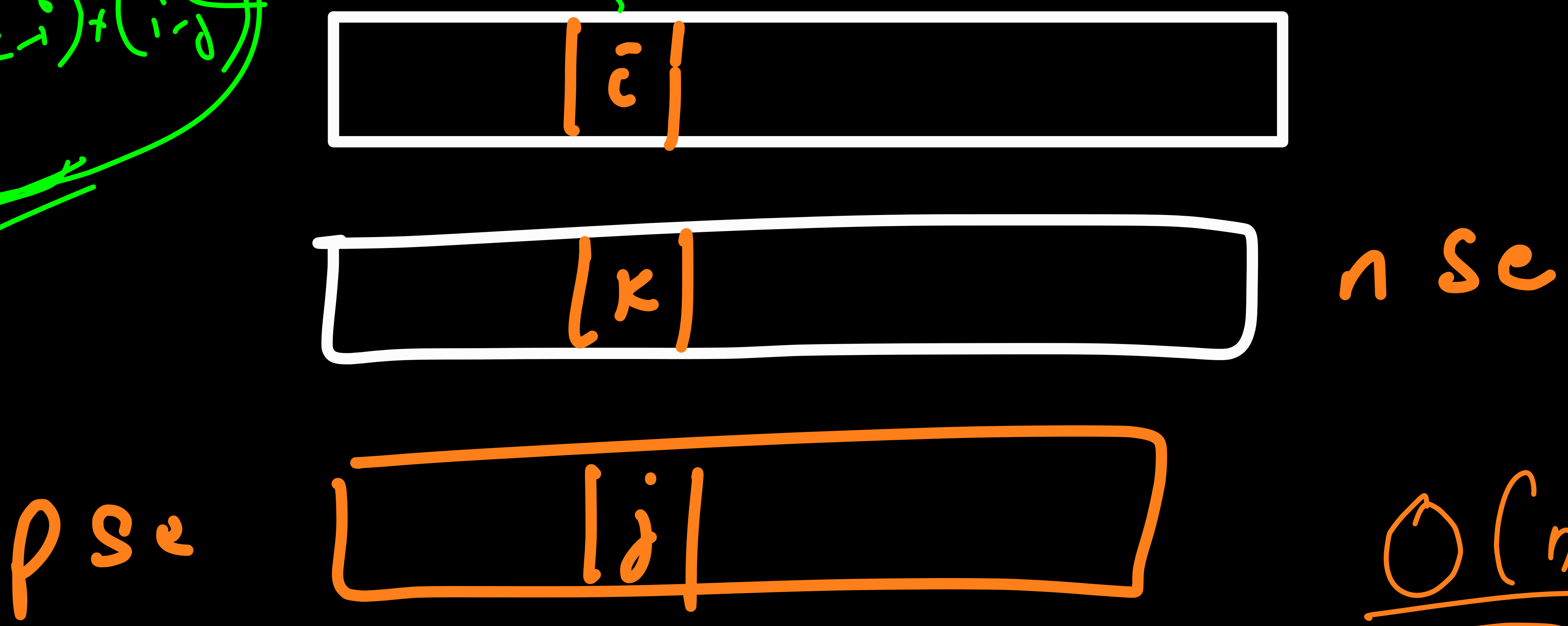
for any element x to be the min, we need to make sure all other elmts are greater.

first elmt index less than x ⟵R1⟵ x (i_x) ⟶R2⟶ first elmt index less than

$$\left(arr[i_x] \times \overset{\#\text{of st}}{(i_x - j)} \times \overset{\#\text{of en}}{(k - i_x)} \right) \longrightarrow$$

# of subarr have arr[ix] as min.

$arr[i] \times (k-i) \times (i-j)$



$O(n)$

$O(n)$