Nearest Smaller Element

Largest Rectangle in Histogram

(Max - Min) in Subarray

Holidays → 25th Dec - 1st Jan

Class 3rd Jan (Friday)

Nearest Smaller Element

Given an integer array A, find index of nearest smaller element on left for all i index in A[].
For all i, find j such that j<i and A[j] < A[i] and j is maximum.

	0	1	2	3	4	5	6	7
A[] →	8	2	4	9	7	6	3	10
ans →	-1	-1	1	2	2	2	1	6

	0	1	2	3	4	5	6	7
A[] →	4	6	10	11	7	8	3	5
ans →	-1	0	1	2	1	4	-1	6

	0	1	2	3	4	5
A[] →	4	5	2	10	8	2
ans →	-1	0	-1	2	2	-1

Brute Force:

For every i, calculate NSL

↓

we will travel from i-1 to 0

↓

return first element smaller than A[i]

```
int ans[N] = {-1}

for (i=0 ; i<N ; i++) {
    for (j= i-1 ; j ≥0 ; j-- ){
        if (a[j] < a[i]) {
            ans[i] = j
            break
        }
    }
}
```

TC : $O(N^2)$

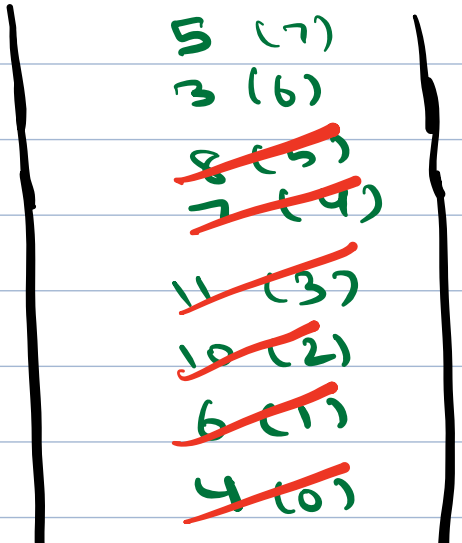
SC : $O(1)$

Optimized Approach :

[8 x x x (5) x x x x x x]

Anything ahead of 5 will never have 8 as nearest smaller element

	0	1	2	3	4	5	6	7
A[] →	4	6	10	11	7	8	3	5
ans :	-1	0	1	2	1	4	-1	6



→ candidates who can be ans

For each element i <

Keep deleting elements from top until they are ≥ A[i]

if (st.empty()) ans = -1

else ans = st.top()

st.push (curr dc → i)

>

```
int[] NSL (int a[], int n) <
```

TC : O(N)

SC : O(N)

```
int ans[n] = <-1>
stack <int> st
for (i=0 ; i<n ; i++) <
  while(!st.isEmpty() && A[st.top()]≥A[i]) <
      st.pop()
  >
  if (! st.isEmpty())
        ans[i] = st.top()
  st.push(i)    // push idx of cur dc
>
return ans
>
```

Q5. Find nearest smaller element on right

Reverse for loop (i = n-1 ; i ≥ 0 ; i--)

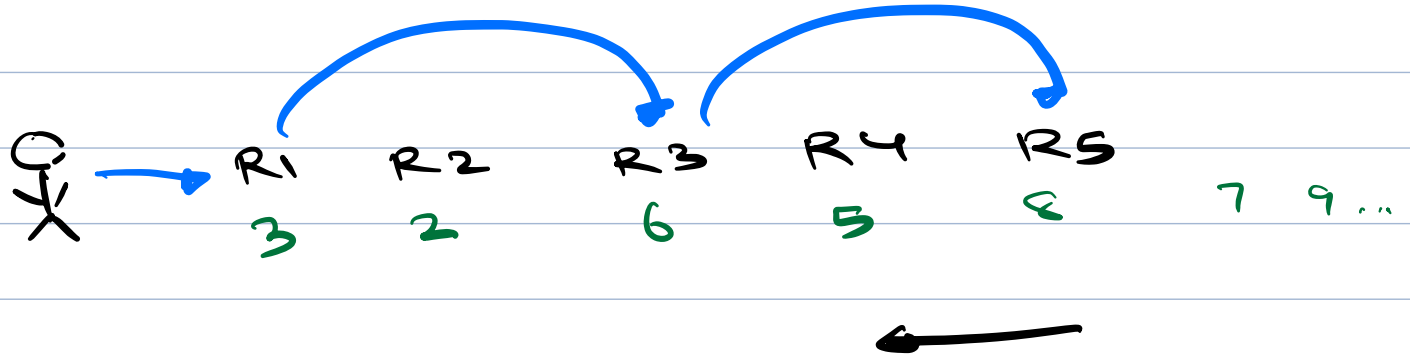
Q3. Find nearest greater element on left

A[st.top()] <= A[i])

Q7. Find nearest greater element on right

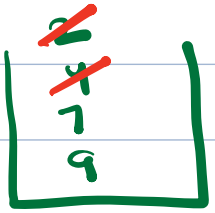
① Right to left

② A[st.top()] <= A[i])



0	1	2	3	4	5	6
3	2	6	5	8	7	9
2	2	4	4	6	6	-1

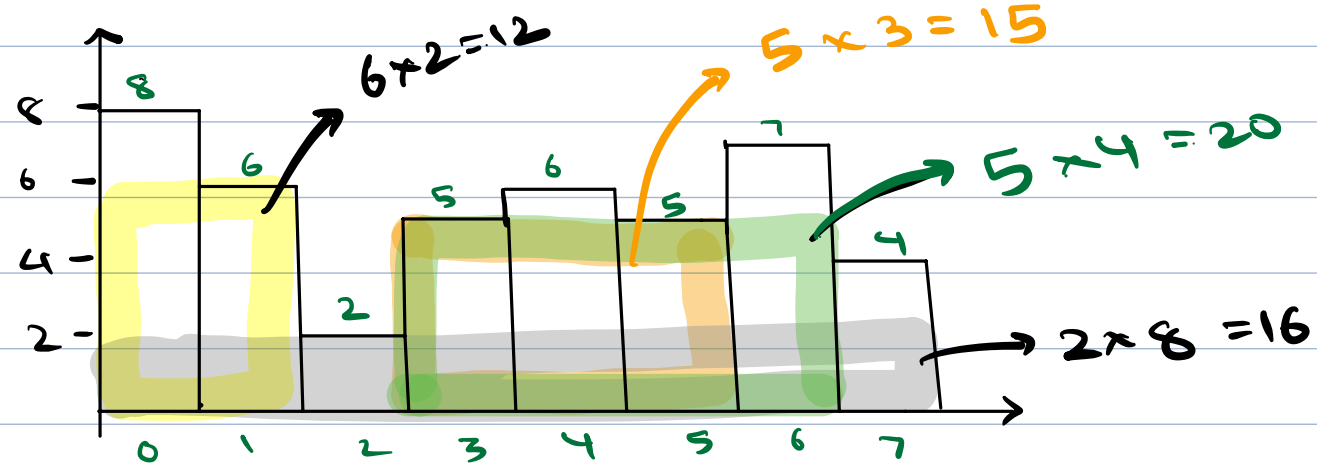
5 2 4 7 9

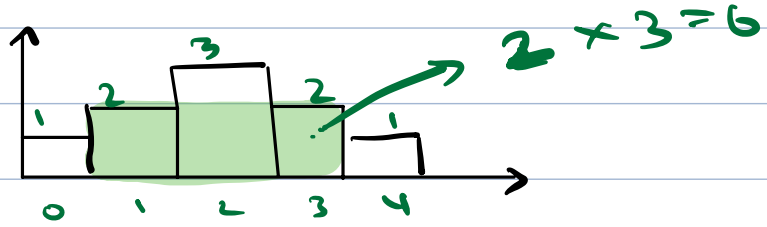


10:11

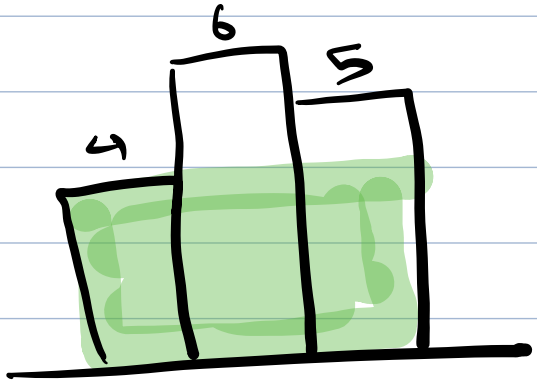
Q. Given an integer array A where
A[i] = height of i^{th} bar
width of each bar = 1
Find area of largest rectangle formed by continuous bars.

A = [8 6 2 5 6 5 7 4] ans = 20





ans = 6



ht = min of selected bars

width = no of bars

ht = min(4, 5, 6) = 4 width = 3

area = 4 x 3 = 12

BF: Try to form as many rectangles as you can

Rectangle → choose a subarray

Take all pairs of s and e

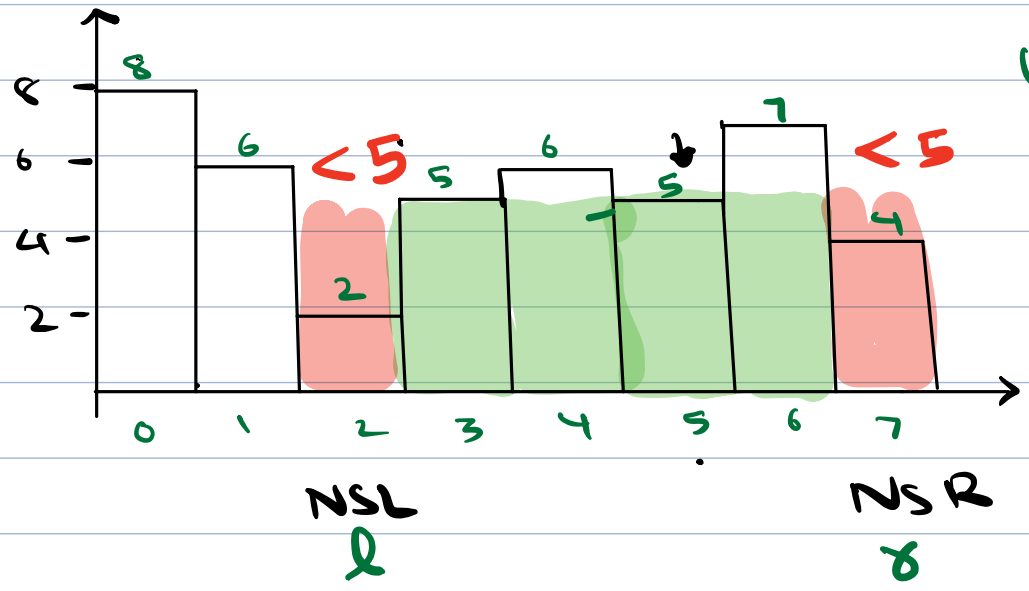
```
int ans = 0
for (s = 0 ; s < N ; s++) {
    int minEle = A[s]
    for (e = s ; e < N ; e++) {
        minEle = min(minEle, A[e])
        area = minEle * (e-s+1)
        ans = max(ans, area)
    }
}
```

TC: $O(N^2)$

SC: $O(1)$

Ans → Max area rectangle → ht of one of bars

Optimized Approach: If I choose a particular bar A[i], what can be max width of rectangle I can get with A[i] as height

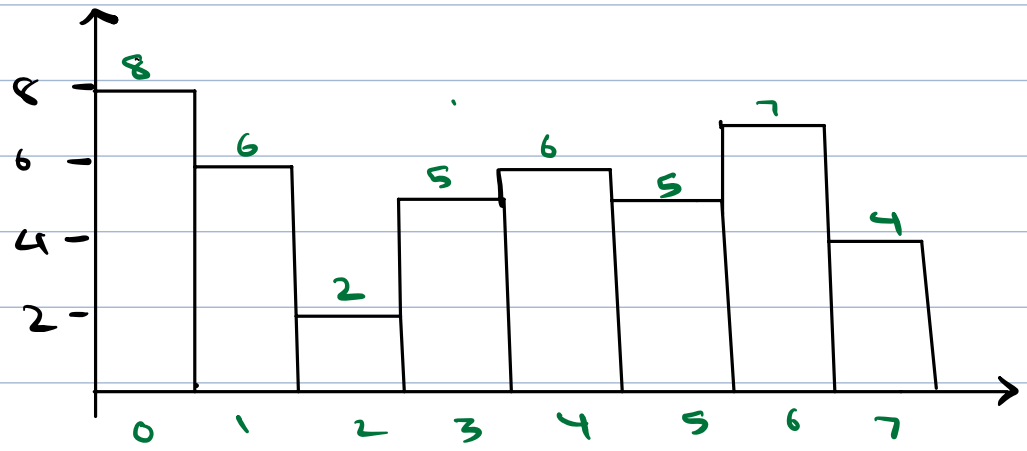


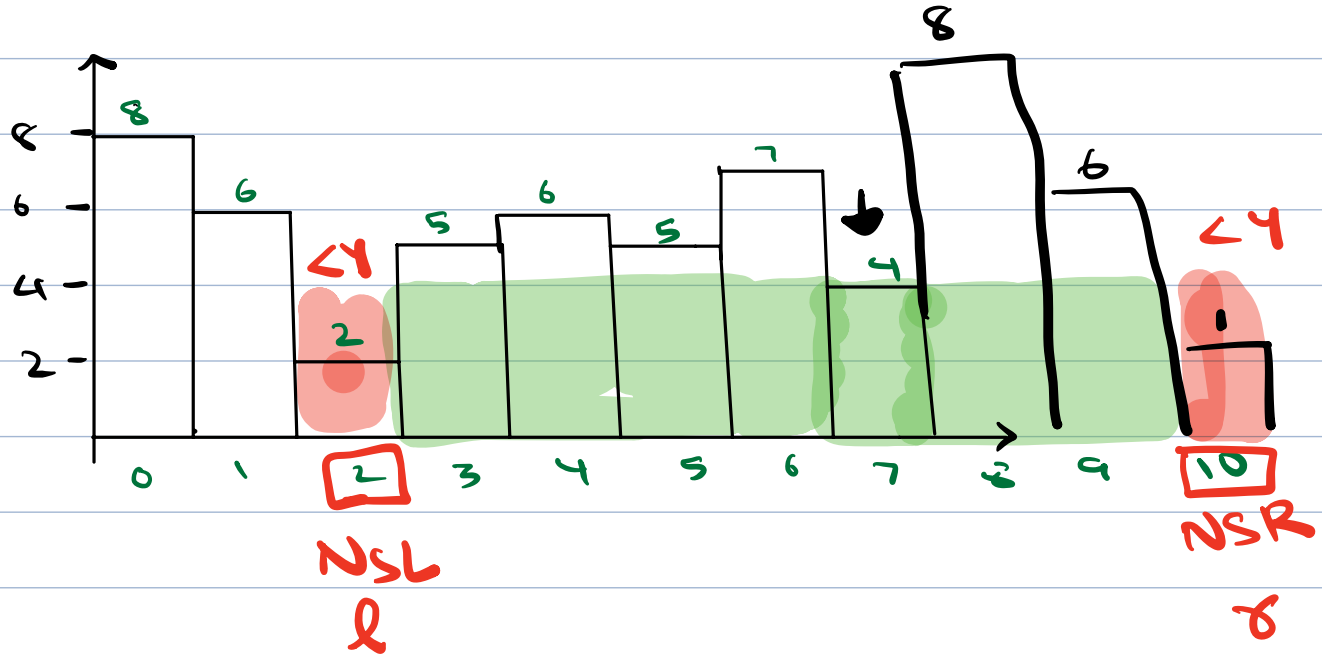
width
$= (r - l + 1) - 2$
$= r - l - 1$

ht = 5

width = 7-2-1
= 4

area = 20





ht = 4 width = 10-2-1 = 7

area = 4 × 7 = 28

For each i {

// ht → a[i]

what is the largest rectangle I can have with a[i] height?

↓

what can be max width

l = nearest smaller on left

r = nearest smaller on right

width = r - l - 1

ht = A[i]

area = ht × width

ans = max(ans - area, area)

}

// Find NSL[N]

int NSL[n] = {-1}

stack<int> st

for (i=0 ; i<n ; i++) {

while(!st.isEmpty() && A[st.top()] ≥ A[i]) {

st.pop()

}

if (!st.isEmpty())

```
        ans[i] = st.top()
    st.push(i)      // push idx of cur de
}
```

// Find NSR[N]

```
int NSR[n] = <N>
stack <int> st
for (i=n-1; i≥0; i--) {
    while(!st.isEmpty() && A[st.top()]≥A[i]) {
        st.pop()
    }
    if (!st.isEmpty())
        ans[i] = st.top()
    st.push(i)      // push idx of cur de
}
```

③
```
int ans=0
for (i=0; i<n; i++) {
    ht = A[i]
    width = NSR[i] - NSL[i] - 1
    cur_area = ht * width
    ans = max(ans, cur_area)
}
return ans
```

TC: O(3N) = O(N)

SC: O(3N) = O(N)

NSL[], NSR[], stack

N = 8

	0	1	2	3	4	5	6	7
A = [	8	6	2	5	6	5	7	4]
(l) NSL	-1	-1	-1	2	3	2	5	2
(r) NSR	1	2	8	7	5	7	7	8
width	1-(-1)-1=1	2-(-1)-1=2	8-(-1)-1=8	7-2-1=4				8-2-1=8
cur_area	8	12	16	20	6	20	7	20

ans = 20

Q. Given an integer array with distinct integers, find sum of (max - min) for all subarrays.

A = [2 (0) 5 3 (2)]

s	e	Max	Min	Max - Min
0	0	2	2	0 +
0	1	5	2	3 +
0	2	5	2	3 +
1	1	5	5	0 +
1	2	5	3	2 +
2	2	3	3	0

ans = 8

$$A = [1 \quad 2 \quad 3]$$ (0, 1, 2)

s	e	Max	Min	Max − Min
0	0	1	1	0
0	1	2	1	1
0	2	3	1	2
1	1	2	2	0
1	2	3	2	1
2	2	3	3	0
		14	10	ans = 4

$14 - 10 = 4$

$1 \times 1 + 2 \times 2 + 3 \times 3 = 14$

BF: Go to every subarray, find max and min; add max - min to ans.

```
int ans = 0
for (s = 0 ; s < N ; s++) {
    int minEle = A[s], maxEle = A[s]
    for (e = s ; e < N ; e++) {
        minEle = min(minEle, A[e])
        maxEle = max(maxEle, A[e])
```

ans += (maxEle - minEle)

return ans

TC: $O(N^2)$

SC: $O(1)$

* We're finding max and min using carry forward

Optimized: Find contribution of every ele and add it to ans

$$ans = \sum_{\text{for all subarrays}} (max - min)$$

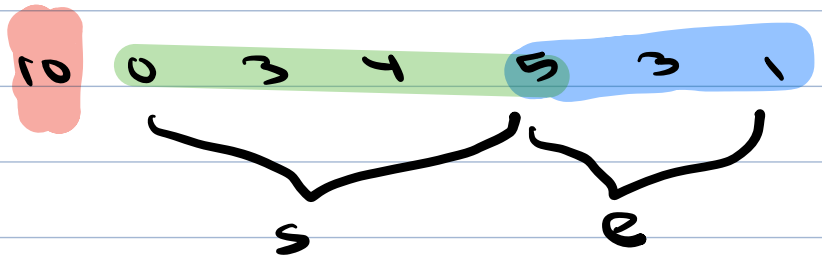
$$= \sum_{\text{for all subarrays}} max - \sum_{\text{for all subarrays}} min$$

$$\sum_{\text{for all subarrays}} max = a[0] \times \text{No. of subarrays in which } a[0] \text{ is max} + a[1] \times \text{No. of subarrays in which } a[1] \text{ is max} +$$

$$a[i] \times \text{No. of subarrays in which } a[i] \text{ is max}$$

How many subarrays A[i] is max?

A = [5 10 0 3 4 5 3 1 6 15 9 0]



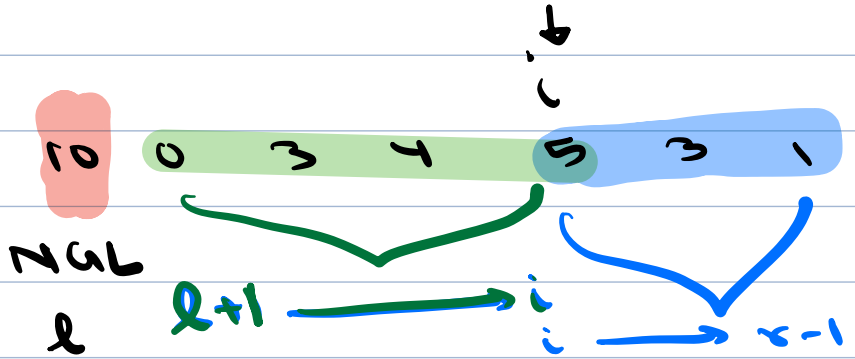
Starting x ending

4 x 3

= 12

Start	End
0	5
3	3
4	1
5	

A = [5 10 0 3 4 5 3 1 6 15 9 0]



No. of starting pts = $i-(l+1)+1$

$= i-l$

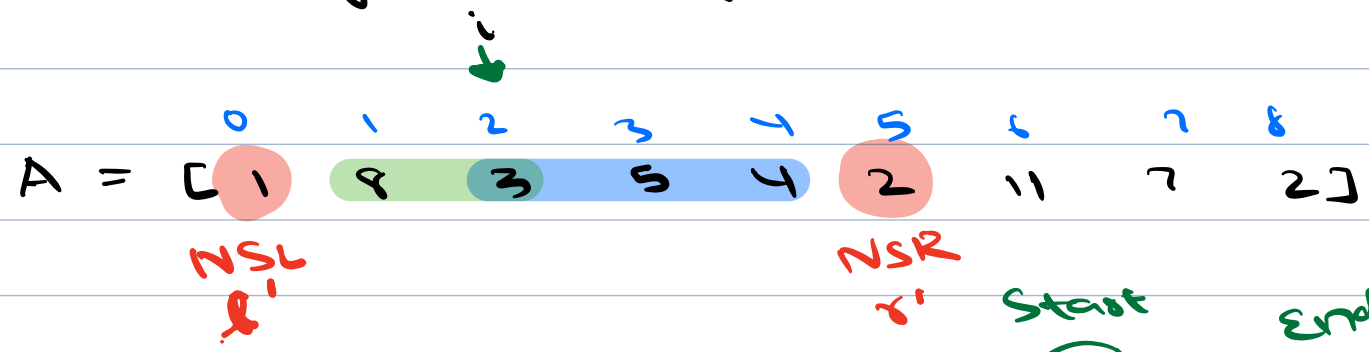
No. of ending pts = $r-1-i+1$

$= r-i$

No. of subarrays = $(i-l) \times (r-i)$

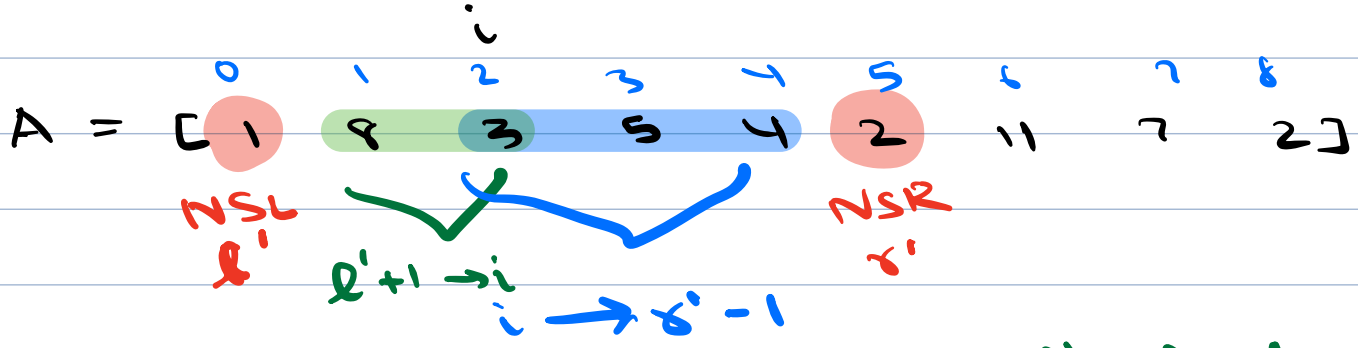
$= (i - NGL[i]) \times (NGR[i])$

How many subarrays A[i] is min?



st = 2, end = 3

No. of subarrays in which 3 is min is 6



No. of starting pts = $i - (l'+1) + 1$

$= i - l' - 1 + 1$

$= i - l'$

No. of ending pts = $r' - 1 - i + 1$

$= r' - i$

No. of subarrays = $(i - l') \times (r' - i)$

$= (i - NSL[i]) \times (NSR[i] - 1)$

$\sum$ min for all subarrays = a[0] x No. of subarrays in which a[0] is min + a[1] x No. of subarrays in which a[1] is min +

a[i] x No. of subarrays in which a[i] is m

① // NSL, NSR, NGL, NGR

NSL[N] = <-1> | NGR = <N>

NGL[N] = <-1> | NGR = <N>

int summax = 0

② for(i=0; i<N; i++) {

// A[i] is max in how many subarrays?

cnt = (i - NGL[i])(NGR[i] - i)

contri = A[i] x cnt

summax = summax + contri

}

(3)

```
int summin = 0
for (i=0; i<N; i++) {
    // A[i] is min in how many subarrays?
    cnt = (i - NSL[i]) (NSR[i] - i)
    contri = A[i] * cnt
    summin = summin + contri
}
return summax - summin
```

TC: O(6N) = O(N)

SC: O(5N) = O(N)

NGL, NGR, NSL, NSR, Stack