Time $O(n^2)$

Space $O(1)$

C++

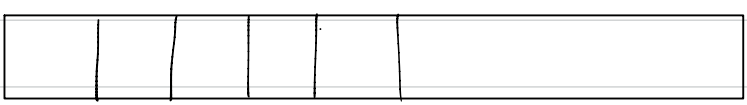
```
long long subArrayRanges(vector<int>& A) {
    long long res = 0;
    for (int i = 0; i < A.size(); i++) {
        int ma = A[i], mi = A[i];
        for (int j = i; j < A.size(); j++) {
            ma = max(ma, A[j]);
            mi = min(mi, A[j]);
            res += ma - mi;
        }
    }
    return res;
}
```

get subarrays by using increasingly larger starting index

⇒ find max value in subarray

⇒ find min value in subarray

all the rest numbers do no matter



Eg: [4,-2,-3,4,1]

subrange

i=0 ma=4 mi=4
- j=0 res+=0 — 4
- j=1 ma=4 mi=-2 res+=6 =6 — 4,-2
- j=2 ma=4 mi=-3 res+=7 =13 — 4,-2,-3
- j=3 ma=4 mi=-3 res+=7 =20 — 4,-2,-3,4
- j=4 ma=4 mi=-3 res+=7 =27 — 4,-2,-3,4,1

i=1 ma=mi=-2
- j=1 res+=0 — -2
- j=2 ma=-2 mi=-3 res+=1 28 — -2,-3
- j=3 ma=4 mi=-3 res+=7 35 — -2,-3,4
- j=4 ma=4 mi=-3 res+=7 42 — -2,-3,4,1

i=2 ma=mi=-3
- j=2 res+=0 — -3
- j=3 ma=4 mi=-3 res+=7 49 — -3,4
- j=4 ma=4 mi=-3 res+=7 56 — -3,4,1

i=3 ma=mi=4
- j=3 res+=0 — 4,1
- j=4 ma=4 mi=1 res+=3 59

Time $O(n)$ ⇒ push/pop each element twice, at most

Space $O(n)$

C++

```cpp
long long subArrayRanges(vector<int>& A) {
    long res = 0, n = A.size(), j, k;
    stack<int> s;
    for (int i = 0; i <= n; ++i) {
        while (!s.empty() && A[s.top()] > (i == n ? -2e9 : A[i])) {
            j = s.top(), s.pop();
            k = s.empty() ? -1 : s.top();
            res -= (long)A[j] * (i - j) * (j - k);
        }
        s.push(i);
    }
    s = stack<int>();
    for (int i = 0; i <= n; ++i) {
        while (!s.empty() && A[s.top()] < (i == n ? 2e9 : A[i])) {
            j = s.top(), s.pop();
            k = s.empty() ? -1 : s.top();
            res += (long)A[j] * (i - j) * (j - k);
        }
        s.push(i);
    }
    return res;
}
```

increasing stack, find previous/next less element

needed to pop out last elements in stack

no PLE/NLE

decreasing stack, find previous/next greater element

Intuition:

Sum up all (max − min) in all subarrays.

So we just need to find sum of max in all subarrays, and subtract min in all subarrays.

So how to find sum of max in all subarrays?

→ say we have array $\{x_1, x_2, x_3 \ldots x_n\}$

need to find # of subarrays a_i with x_i being the max, i.e. $\sum_1^n a_i x_i$

→ in coding, use monotomic stack to find previous/next greater/less element

eg. $\{9, 2, 10, 8, 5, 7, 4, 1\}$

find array with 5 as smallest element, sum up minimums

$\{9, 2, \{10, 8, 5, 7\}, 4, 1\}$

look for PLE/NLE, 3 positions for left { , 2 positions for right }

∴ Sum = $(3 \times 2) \times 5 = 30$