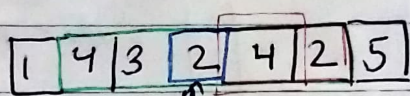
907: Sum of Subarray Minimums

Sum the minimum elements of all possible sub arrays.

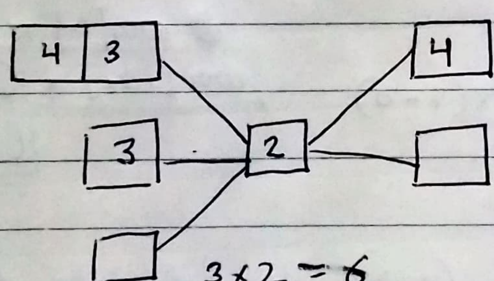
$O(N^2)$ solution is straightforward.

Algorithm

Instead of finding subarrays and then finding minimum. Find the count/number of subarrays in which an element can be minimum.



Number of subarrays wher 2 will be minimum.



4, 3, 2, 4
4, 3, 2
3, 2, 4
3, 2
2, 4
2

left_index = index at which element is >= current element. (left)

right_index = index at which element is >= current element (right)

total sum from current index (idx).

$$\boxed{arr[idx] * (idx - left\_idx) * (right\_idx - idx)}$$

$$\boxed{2 \times 3 \times 2 = 12}$$

We have to do it ~~fo~~ for all the elements.

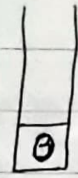
but how it can be done in $O(n)$

Monotonically Increasingly stack

The top element would always be greater if the current top element is greater then first pop the top element & then push our element.

Simple example

[3, 1, 2, 4]
↑



← index of 3

[3, 1, 2, 4]
↑

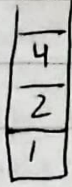
idx = pop 0 (3 - index)
left-idx = -1 (stack empty)
right_idx = 1

~~arr[0] x~~

$\boxed{arr[idx] \times (idx - left\_idx) \times (right\_idx - idx)}$

$3 \times (0-(-1)) \times (1-0)$
$= 3$

We will use stack (pop) to calculate the contribution to sum from that element.



Push the elements to the stack, as each element is greater than previous one.

* When iteration is done (finished) some elements on the stack would be left.

pop them one by one and use
right_idx = len(arr) for calculation

$4 \rightarrow 4 \times (3-2) \times (4-3) = 4$
$2 \rightarrow 2 \times (2-1) \times (4-2) = 4$ $\rightarrow 3+4+4+6 = 17$
$1 \rightarrow 1 \times (1-(-1)) \times (4-1) = 6$

↑ stack empty