Sum the minimum elements of all possible sub arrays O(N2) solution is straightforward. Algorithm Trustead of finding subarrays and finding minimum. Find the count/number of subarrays in which an element can be minimum. IT 3 2 4 2 5 Number of subarrays wher 2 will be minimum. Number of subarrays wher 2 will be minimum. H 3 4 4,3,2,4 3,2 3,2 2,4 2,4	907: Sum of Subarray Minimums	
Algorithm Instead of finding subarrays and finding minimum. Find the count/number of subarrays in which an element can be minimum. I Y 3 2 4 2 5 Number of subarrays wher 2 will be minimum. I Y 3 2 4 2 5 Number of subarrays wher 2 will be minimum. I Y 3 2 4 2 5 Number of subarrays wher 2 will be minimum. I Y 3 2 4 2 5 Number of subarrays wher 2 will be minimum. I Y 3 2 4 2 5 I H 3 4 4 3, 2, 4 I H 3 2 4 2 5 I H 3 5 5 Number of subarrays wher 2 will be minimum. I Y 3 2 4 2 5 I H 3 5 7 6 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7		•
Tratead of finding subarrays and finding minimum. Find the count/number of subarrays in which an element can be minimum. IT 3 2 4 2 5 Number of subarrays wher 2 will be minimum. H 3 4 4,3,2,4 H,3,2 3,2,4 3,2,4 3,2,4 2,4 2,4 2,4		
Number of subarrays wher 2 will be minimum. 4 3 4 4,3,2,4 4,3,2 3,2,4 3,2 2,4 2,4 2 2,4 2 left-index = index at which element is >= warrent element; (left) sught-index = index at which element is >= warrent element (right) total sum from current index(idx). [avoCidx] * (idx - left-idax) * [avoCidx] * [avoCidx] * (idx - left-idax) * [avoCidx] * [avoCid	Instead of finding subarrays and finding minimum. Fitthe count/number of subarrays in which an element can	nd
Teff-index = index at which element is >= wrent element (left) sught-index = index at which element is >= current element (right) total sum from current index (idx) avalidx) * (idx - left-idx) * [2 x 3 x 2 = 12] We have to do it fatt for all the elements		•
we have to do it July for all the elements	3 2 3, 2, 4 3, 2 2, 4	
total sum from current index (idx). [avr[cidx] * (idx - left-idx) * [2 x 3 x 2 = 12] We have to do it fall for all the elements		0
we have to do it tothe for all the elements	suight-index = index at which element is > = current element (right)	
We have to do it tothe for all the elements.		
We have to do it tothe for all the elements	(night-idx)	iex)
		•

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 $\begin{bmatrix}
3, 1, 2, 4
\end{bmatrix}$ $\Theta \leftarrow index of 3$

[3,1,2,4] | idx = pop @ 0 (3-index)

left-idx = -1 (stack empty)

sught-idx = 1

 $3 \times (0-(-1)) \times (1-0)$ = (3) $wor[o] \times (1dx - left_idx) \times (1dx - left_idx) \times (xight_idx - dx)$

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 stock would be left.

pop them one by one and use sight-idx = len(avr) 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 > 1 x (1-(-1)) x (4-1) = 6