08 Sum of Subarray Ranges

21 February 2025 23:03

2104. Sum of Subarray Ranges

Medium | Topics | Companies | Hint

You are given an integer array `nums`. The **range** of a subarray of `nums` is the difference between the largest and smallest element in the subarray.

Return *the **sum of all** subarray ranges of* `nums`.

A subarray is a contiguous **non-empty** sequence of elements within an array.

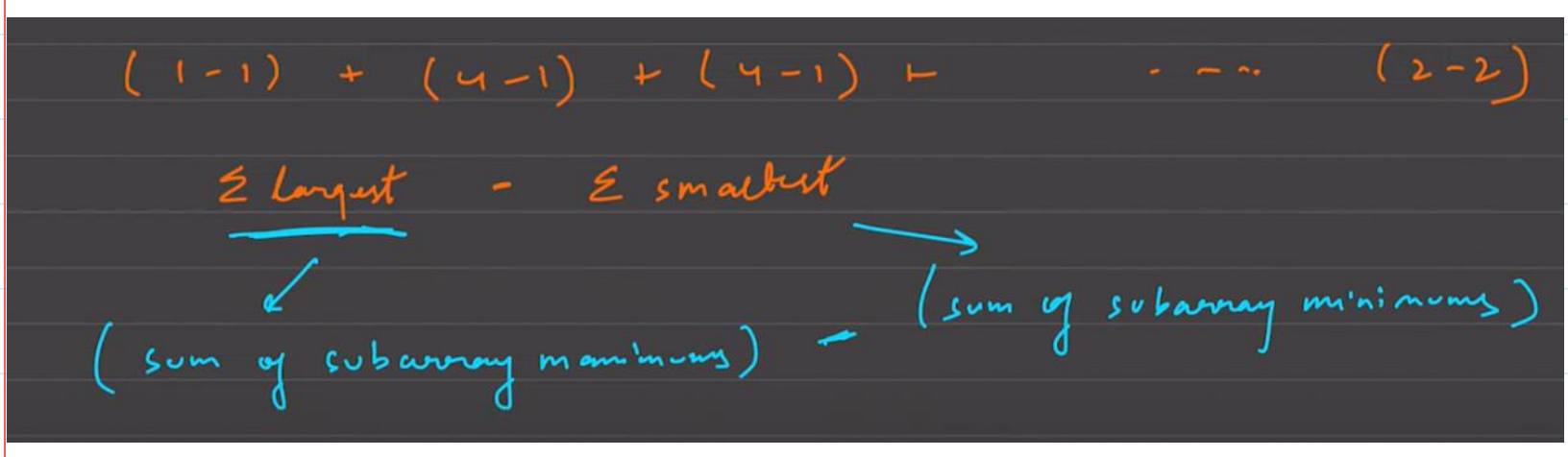
Example 1:

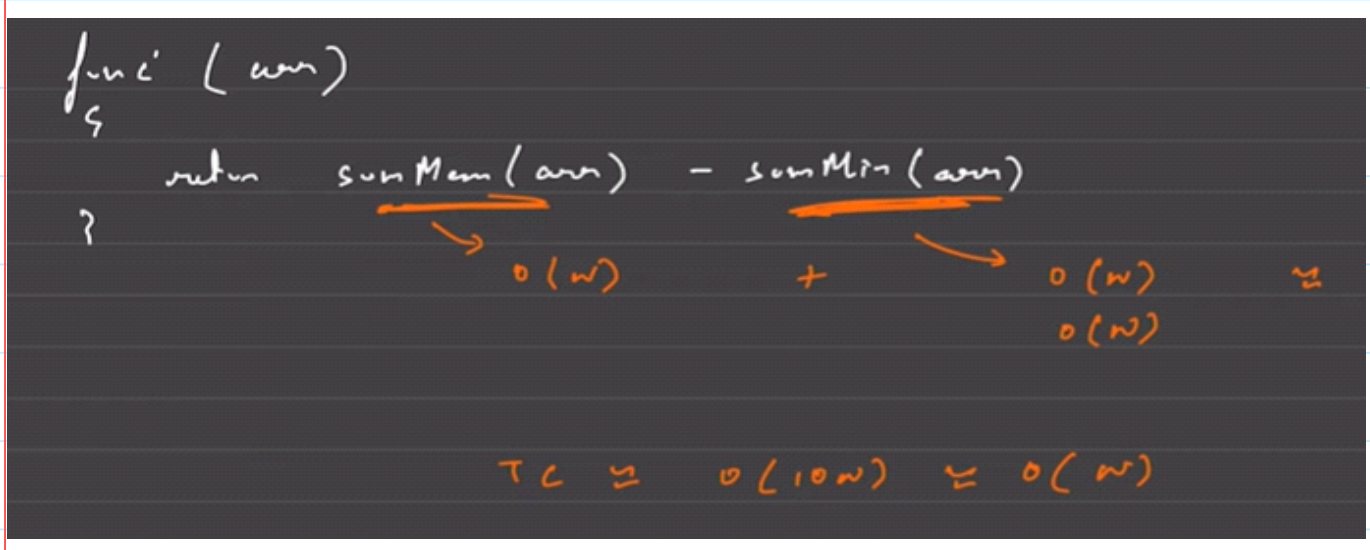
```
Input: nums = [1,2,3]
Output: 4
Explanation: The 6 subarrays of nums are the following:
[1], range = largest - smallest = 1 - 1 = 0
[2], range = 2 - 2 = 0
[3], range = 3 - 3 = 0
[1,2], range = 2 - 1 = 1
[2,3], range = 3 - 2 = 1
[1,2,3], range = 3 - 1 = 2
So the sum of all ranges is 0 + 0 + 0 + 1 + 1 + 2 = 4.
```


```
public long subArrayRanges(int[] nums)
{
    long ans =0 , n=nums.length;
    for(int i=0;i<n;i++){
      int min=nums[i] , max =nums[i];
      for(int j=i;j<n;j++){
        min=Math.min(min,nums[j]);
        max=Math.max(max,nums[j]);
         ans+=(max-min);
      }
    }
    return ans;
  }
```

Time complexity :- O(N^2)

Second Approach ... Method-2 : find the sum of largest element inside the each subarray, find the sum of smallest element inside the sub array, Then, return The return it's difference...





code

22 February 2025 09:46

```
class Solution {
    public long subArrayRanges(int[] nums) {
        return (sumOfLargest(nums)-sumOfSmallest(nums));
    }
_____
_____

    public static long sumOfLargest(int[] nums){
        int[] nge=nge(nums);
        int[] pnge=pnge(nums);
        long ans=0;

        for(int i=0;i<nums.length;i++){
            long a=nge[i]-i;
            long b=i-pnge[i];
            long c=a*b;
            ans+=(c*nums[i]);
        }
        return ans;
    }
_____
_____

    public static long sumOfSmallest(int[] nums){
        int[] nse=nse(nums);
        int[] pnse=pnse(nums);
        long ans=0;
        for(int i=0;i<nums.length;i++){
            long a=nse[i]-i;
            long b=i-pnse[i];
            long c=a*b;
            ans+=(c*nums[i]);
        }
        return ans;
    }
_____
_____

    public static int[] nge(int[] arr){
        int[] ans=new int[arr.length];
        Stack<Integer> stack=new Stack<>();
        int n=arr.length;
        for(int i=n-1;i>=0;i--){
            while(!stack.isEmpty() && arr[i]>=arr[stack.peek()])
            stack.pop();
            ans[i]=(stack.isEmpty())?n:stack.peek();
            stack.push(i);
        }
        return ans;
    }
_____
_____

    public static int[] pnge(int[] arr){
        int[] ans =new int[arr.length];
        Stack<Integer> stack=new Stack<>();
        int n=arr.length;
        for(int i=0;i<n;i++){
            while(!stack.isEmpty() && arr[i]>arr[stack.peek()])
            stack.pop();
            ans[i]=(stack.isEmpty())?-1:stack.peek();
            stack.push(i);
        }
        return ans;
    }
_____
_____

    public static int[] nse(int[] arr){
        int[] ans=new int[arr.length];
        Stack<Integer> stack=new Stack<>();
        int n=arr.length;
        for(int i=n-1;i>=0;i--){
            while(!stack.isEmpty() && arr[i]<=arr[stack.peek()])
            stack.pop();

            ans[i]=stack.isEmpty()?n:stack.peek();
            stack.push(i);
        }
        return ans;
    }
_____
_____

    public static int[] pnse(int[] arr){
        int[] ans=new int[arr.length];
        Stack<Integer> stack=new Stack<>();
        int n=arr.length;
        for(int i=0;i<n;i++){
            while(!stack.isEmpty() && arr[i]<arr[stack.peek()])
            stack.pop();

            ans[i]=stack.isEmpty()?-1:stack.peek();
            stack.push(i);
        }
        return ans;
    }
}
```