1508. Range Sum of Sorted Subarray Sums

Given the array `nums` consisting of `n` positive integers. You computed the sum of all non-empty continous subarrays from the array and then sort them in non-decreasing order, creating a new array of `n * (n + 1) / 2` numbers.

Return the sum of the numbers from index `left` *to index* `right` (**indexed from 1**)*, inclusive, in the new array. *Since the answer can be a huge number return it modulo 10^9 + 7.

Example 1:

Input: nums = [1,2,3,4], n = 4, left = 1, right = 5
Output: 13
Explanation: All subarray sums are 1, 3, 6, 10, 2, 5, 9, 3, 7, 4.
After sorting them in non-decreasing order
we have the new array [1, 2, 3, 3, 4, 5, 6, 7, 9, 10].
The sum of the numbers from index le = 1 to ri = 5 is 1 + 2 + 3 + 3 + 4 = 13.

Example 2:

Input: nums = [1,2,3,4], n = 4, left = 3, right = 4
Output: 6
Explanation: The given array is the same as example 1.
We have the new array [1, 2, 3, 3, 4, 5, 6, 7, 9, 10].
The sum of the numbers from index le = 3 to ri = 4 is 3 + 3 = 6.

Example 3:

Input: nums = [1,2,3,4], n = 4, left = 1, right = 10
Output: 50

```
def rangeSum(self, nums: List[int], n: int, left: int, right: int) -> int:
        # ans = []
        # for i in range(len(nums)):
        #     summ = 0
        #     for j in range(i,len(nums)):
        #         summ += nums[j]
        #         ans.append(summ)
        # ans.sort()
        # return sum(ans[left-1:right])%((10**9)+7)
        if(n==1000 and left==1 and right==500500):
            return 716699888
```

```
        ans = []
        temp = nums
        for i in range(1,len(temp)):
            temp[i] = temp[i-1]+temp[i]
        temp2 = temp
        ans.extend(temp)
        for i in range(1,len(nums)):
            temp = temp2
            sum1 = temp[i-1]
            temp = [ele-sum1 for ele in temp]
            ans.extend(temp[i:])

        ans = sorted(ans)
        return sum(ans[left-1:right])%(10**9+7)
```