# Problem 1508: Range Sum of Sorted Subarray Sums

## Problem Information

**Difficulty:** Medium
**Acceptance Rate:** 63.09%
**Paid Only:** No
**Tags:** Array, Two Pointers, Binary Search, Sorting, Prefix Sum

## Problem Description

You are given the array `nums` consisting of `n` positive integers. You computed the sum of all non-empty continuous subarrays from the array and then sorted 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 `109 + 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

**Constraints:**

* `n == nums.length` * `1 <= nums.length <= 1000` * `1 <= nums[i] <= 100` * `1 <= left <= right <= n * (n + 1) / 2`

## Code Snippets

**C++:**

```
class Solution {
public:
int rangeSum(vector<int>& nums, int n, int left, int right) {

}
};
```

**Java:**

```
class Solution {
public int rangeSum(int[] nums, int n, int left, int right) {

}
}
```

**Python3:**

```
class Solution:
def rangeSum(self, nums: List[int], n: int, left: int, right: int) -> int:
```