1508. Range Sum of Sorted Subarray Sums

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 `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
```

Constraints:

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

Range Sum of Sorted Subarray Sums

```
/*
    Brute force (preprocessing)
    Time complexity:O(n+n+n^2+nlogn+n)=O(n^2+nlogn+3n)=O(n^2)
    Space complexity: O(n+n+logn)=O(2n+logn)=O(n)
*/
class Solution {
public:
    int rangeSum(std::vector<int>& nums, int n, int left, int right) {
        int MOD=1'000'000'007;
        std::vector<int> prefix_sum(n+1,0);
        for(int i=1;i<=n;++i) prefix_sum[i]=prefix_sum[i-1]+nums[i-1];

        std::vector<int> sums;
        for(int i=0;i<=n;++i){
            for(int j=i+1;j<=n;++j){
                sums.push_back(prefix_sum[j]-prefix_sum[i]);
            }
        }

        std::sort(sums.begin(),sums.end());

        int s=0;
        left--;
        right--;
        for(int i=left;i<=right;++i) s=(s%MOD+sums[i]);
        return s;
    }
};
```

Range Sum of Sorted Subarray Sums

```
/*
    Brute force
    Time complexity:O(n^2+nlogn+n)=O(n^2+nlogn+n)=O(n^2)
    Space complexity: O(n+logn)=O(n+logn)=O(n)
*/
class Solution {
public:
    int rangeSum(std::vector<int>& nums, int n, int left, int right) {
        int MOD=1'000'000'007;
        std::vector<int> sums;
        for(int i=0;i<n;++i){
            int s=0;
            for(int j=i;j<n;++j){
                s+=nums[j];
                sums.push_back(s);
            }
        }

        std::sort(sums.begin(),sums.end());

        int s=0;
        left--;
        right--;
        for(int i=left;i<=right;++i) s=s%MOD+sums[i];

        return s;
    }
};
```

Range Sum of Sorted Subarray Sums

```
/*
    min heap
    Time complexity:O(nlogn)
    Space complexity: O(n)
*/
typedef std::pair<int,int> ii;
typedef std::vector<ii> vii;

class Solution {
    public:
        int rangeSum(std::vector<int>& nums, int n, int left, int right) {
            int MOD=1'000'000'007;

            std::priority_queue<ii,vii,std::greater<ii>> min_heap;

            for(int i=0;i<n;++i) min_heap.push({nums[i],i});

            int s=0;
            for(int i=0;i<right;++i){
                auto [e,index]=min_heap.top();
                min_heap.pop();
                if(i>=left-1) s=(s+e)%MOD;
                if(index+1<n) min_heap.push({e+nums[index+1],index+1});
            }
            return s;
        }
};
```