Problem 327: Count of Range Sum

Problem Information

Difficulty: Hard
Acceptance Rate: 37.92%
Paid Only: No
Tags: Array, Binary Search, Divide and Conquer, Binary Indexed Tree, Segment Tree, Merge Sort, Ordered Set

Problem Description

Given an integer array `nums` and two integers `lower` and `upper`, return _the number of range sums that lie in_ `[lower, upper]` _inclusive_.

Range sum `S(i, j)` is defined as the sum of the elements in `nums` between indices `i` and `j` inclusive, where `i <= j`.

Example 1:

Input: nums = [-2,5,-1], lower = -2, upper = 2 **Output:** 3 **Explanation:** The three ranges are: [0,0], [2,2], and [0,2] and their respective sums are: -2, -1, 2.

Example 2:

Input: nums = [0], lower = 0, upper = 0 **Output:** 1

Constraints:

* `1 <= nums.length <= 105` * `-231 <= nums[i] <= 231 - 1` * `-105 <= lower <= upper <= 105` * The answer is **guaranteed** to fit in a **32-bit** integer.

Code Snippets

C++:

```
class Solution {
public:
int countRangeSum(vector<int>& nums, int lower, int upper) {

}
};
```

Java:

```
class Solution {
public int countRangeSum(int[] nums, int lower, int upper) {

}
}
```

Python3:

```
class Solution:
def countRangeSum(self, nums: List[int], lower: int, upper: int) -> int:
```