Problem 209: Minimum Size Subarray Sum

Problem Information

Difficulty: Medium
Acceptance Rate: 50.34%
Paid Only: No
Tags: Array, Binary Search, Sliding Window, Prefix Sum

Problem Description

Given an array of positive integers `nums` and a positive integer `target`, return _the**minimal length** of a __subarray_ _whose sum is greater than or equal to_ `target`. If there is no such subarray, return `0` instead.

Example 1:

Input: target = 7, nums = [2,3,1,2,4,3] **Output:** 2 **Explanation:** The subarray [4,3] has the minimal length under the problem constraint.

Example 2:

Input: target = 4, nums = [1,4,4] **Output:** 1

Example 3:

Input: target = 11, nums = [1,1,1,1,1,1,1,1] **Output:** 0

Constraints:

* `1 <= target <= 109` * `1 <= nums.length <= 105` * `1 <= nums[i] <= 104`

Follow up: If you have figured out the `O(n)` solution, try coding another solution of which the time complexity is `O(n log(n))`.

Code Snippets

C++:

```
class Solution {
public:
int minSubArrayLen(int target, vector<int>& nums) {

}
};
```

Java:

```
class Solution {
public int minSubArrayLen(int target, int[] nums) {

}
}
```

Python3:

```
class Solution:
def minSubArrayLen(self, target: int, nums: List[int]) -> int:
```