209. Minimum Size Subarray Sum

Given an array of positive integers `nums` and a positive integer `target`, return the minimal length of a **contiguous subarray** `[nums<sub>l</sub>, nums<sub>l+1</sub>, ..., nums<sub>r-1</sub>, nums<sub>r</sub>]` of which the 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 <= 10<sup>9</sup>`
- `1 <= nums.length <= 10<sup>5</sup>`
- `1 <= nums[i] <= 10<sup>5</sup>`

-
  ```
  def minSubArrayLen(self, target: int, nums: List[int]) -> int:
          if sum(nums)<target:
              return 0
          if sum(nums)==target:
              return len(nums)
          ans = 0
          temp = 0
          i = -1
          j = -1
          while True:
  ```

```
            f1 = False
            f2 = False
            while i<len(nums)-1 and temp<target:
                i = i+1
                temp = temp+nums[i]
                f1 = True

            while j<i and temp>=target:
                pAns = i-j
                ans = min(pAns,ans) if ans!=0 else pAns
                j = j+1
                temp = temp-nums[j]
                f2 = True

            if f1 is False and f2 is False:
                break
        return ans
```