99. Minimum size subarray sum:

I/P: target = 7, nums = [2,3,1,2,4,3]

O/P: 2

Explanation: The subarray [4,3] has the minimal length under the problem constraint.

Brute force → slow way $O(n^2)$:

Optimized:

use two pointers: start and end

Expand end to increase the sum

shrink start when sum >= target.

```
class Solution {
    public int minSubArrayLen(int target, int[] nums) {
        int left = 0;
        int sum = 0;
        int minLength = Integer.MAX_VALUE;
    for(int right = 0; right < nums.length; right++){
        sum = sum + nums[right];
    while(sum >= target) {
        minLength = Math.min(minLength, right - left + 1);
        sum = sum - nums[left];
        left++;
    }
    }
    return minLength == Integer.MAX_VALUE ? 0 :
                                    minLength;
    }
}
```

Time → $O(n)$ space → $O(1)$.