**Minimum Size Subarray Sum**

Given an array of **n** positive integers and a positive integer **s**, find the minimal length of a **contiguous** subarray of which the sum ≥ **s**. If there isn't one, return 0 instead.

**Example:**

**Input:** s = 7, nums = [2,3,1,2,4,3]

**Output:** 2

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

**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*).

**Solution**:

// Sliding Window Problem

class Solution {

public int minSubArrayLen(int s, int[] nums) {

if(nums == null || nums.length == 0){

return 0;

}

int res = nums.length+1;

int left = 0;

int sum = 0;

for( int right = 0; right < nums.length; right++){

sum +=nums[right];

while(sum >= s && left <= right ){

res = Math.min(res, right-left +1);

sum -= nums[left];

left++;

}

}

return res == nums.length+1 ? 0 : res;

}

}