# 题目

给定一个含有n个正整数的数组和一个正整数s ，找出该数组中满足其和 ≥s的长度最小的 连续 子数组，并返回其长度。如果不存在符合条件的子数组，返回0。

**示例：**

输入：s = 7, nums = [2,3,1,2,4,3]

输出：2

解释：子数组 [4,3] 是该条件下的长度最小的子数组。

**进阶：**

如果你已经完成了 O(n) 时间复杂度的解法, 请尝试 O(nlogn) 时间复杂度的解法。

# 分析

## 方法一：暴力法

## 方法二：二分查找

**思路：**

**代码：**

class Solution {

public:

int minSubArrayLen(int s, vector<int>& nums) {

if(nums.size()==0)return 0;

int left=0,right=0;

int sum=0,ret=INT\_MAX;

while(right<nums.size())

{

sum += nums[right];

while(sum>=s)

{

ret = min(ret,right-left+1);

sum -= nums[left++];

}

right++;

}

return ret==INT\_MAX?0:ret;

}

};

**复杂度分析：**

时间复杂度：O(nlogn)，其中n是数组的长度。需要遍历每个下标作为子数组的开始下标，遍历的时间复杂度是O(n)，对于每个开始下标，需要通过二分查找得到长度最小的子数组，二分查找得时间复杂度是(logn)，因此总时间复杂度是O(nlogn)。

空间复杂度O(n)，其中n是数组的长度。额外创建数组sums存储前缀和。

## 方法三：双指针