# 题目

给定一个含有 n 个正整数的数组和一个正整数 target 。

找出该数组中满足其和 ≥ target 的长度最小的 连续子数组 [numsl, numsl+1, ..., numsr-1, numsr] ，并返回其长度。如果不存在符合条件的子数组，返回 0 。

示例 1：

输入：target = 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。

示例 2：

输入：target = 4, nums = [1,4,4]

输出：1

示例 3：

输入：target = 11, nums = [1,1,1,1,1,1,1,1]

输出：0

提示：

1 <= target <= 109

1 <= nums.length <= 105

1 <= nums[i] <= 105

进阶：

如果你已经实现O(n)时间复杂度的解法, 请尝试设计一个O(n log(n))时间复杂度的解法。

注意：本题与主站 209 题相同：https://leetcode-cn.com/problems/minimum-size-subarray-sum/

# 分析

## 方法一：暴力法

## 方法二：二分查找

class Solution {

public:

    int minSubArrayLen(int target, 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>=target)

            {

                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。