/\*

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.

For example, given the array [2,3,1,2,4,3] and s = 7,

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

法一：o(n^2)

法二：两个指针，移动

\*/

class Solution {

public:

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

{

//way-1 有一部分重复运算

/\*

int min=nums.size()+1;

int sum=0;

for(int i=0;i<nums.size();i++)

{

for(int j=i;j<nums.size();j++)

{

sum+=nums[j];

if(sum>=s && j-i+1<min)

min=j-i+1;

}

sum=0;

}

if(min==nums.size()+1)

return 0;

return min;

\*/

//way-2

int first=0,second=-1;

int sum=0;

int length=nums.size();

int ret=length+1;

while(first<length && second<length)

{

//second后移，直到sum超过s

while(sum<s && second<length)

{

second++;

sum+=nums[second];

}

if(second==nums.size())

break;

//first后移，直到sum小于s

while(sum>=s)

{

ret=(ret>(second-first+1))?(second-first+1):ret;

sum-=nums[first];

first++;

}

}

if(ret==nums.size()+1)

return 0;

return ret;

}

};