/\*

Given an array of non-negative integers, you are initially positioned at the first index of the array.

Each element in the array represents your maximum jump length at that position.

Your goal is to reach the last index in the minimum number of jumps.

For example:

Given array A = [2,3,1,1,4]

The minimum number of jumps to reach the last index is 2. (Jump 1 step from index 0 to 1, then 3 steps to the last index.)

Note:

You can assume that you can always reach the last index.

way-1:用vector来存到达i的步数

way-2:(copy)

ret:目前为止的jump数

curRch:从A[0]进行ret次jump之后达到的最大范围

curMax:从0~i这i+1个A元素中能达到的最大范围

当curRch < i，说明ret次jump已经不足以覆盖当前第i个元素，因此需要增加一次jump，使之达到

记录的curMax。

\*/

class Solution {

public:

int jump(vector<int>& nums)

{

//way-1

/\*

if(nums.size()<2)

return 0;

vector<int> steps(nums.size(),0);

int maxsize=0;

for(int i=0;i<nums.size()-1;i++)

{

if(i+nums[i]<=maxsize)

continue;

else if(i+nums[i]>=nums.size()-1)

return steps[i]+1;

else

for(int j=maxsize+1;j<=i+nums[i];j++)

steps[j]=steps[i]+1;

maxsize=i+nums[i];

}

return steps[nums.size()-1];

\*/

int ret = 0;

int curMax = 0;

int curRch = 0;

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

{

if(curRch < i)

{

ret ++;

curRch = curMax;

}

curMax = max(curMax, nums[i]+i);

}

return ret;

}

};