1. Jump Game

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.

Determine if you are able to reach the last index.

**Example 1:**

Input: [2,3,1,1,4]  
Output: true  
Explanation: Jump 1 step from index 0 to 1, then 3 steps to the last index.

**Example 2:**

Input: [3,2,1,0,4]  
Output: false  
Explanation: You will always arrive at index 3 no matter what. Its maximum  
 jump length is 0, which makes it impossible to reach the last index.

思路：有点像走台阶。自己写的代码太慢了，而且占用的空间大

倒着走，如果要从位置i调到位置j，必须满足条件nums[i] >= j - i，另外用一个数组havePath做标记，havePath[i]表示可以从位置0到位置i，是为了防止在0不能到达i时，重复从其他位置到i然后继续在0到i之间搜索

时间复杂度：

空间复杂度：

class Solution {  
public:  
 bool canJump(vector<int>& nums) {  
 vector<bool> havePath(nums.size(), true);  
 bool ans = false;  
 int curPos = nums.size() - 1;  
 search(nums, havePath, ans, curPos);  
 return ans;  
 }  
 void search(vector<int> &nums, vector<bool> &havePath, bool &ans, int curPos){  
 if(ans)return;  
 if(curPos == 0){  
 ans = true;  
 return;  
 }  
 bool canVisit = false;  
 for(int i = curPos - 1; i >= 0; --i){  
 if(nums[i] >= curPos - i && havePath[i]){  
 search(nums, havePath, ans, i);  
 canVisit = true;  
 }  
 }  
 if(!canVisit)havePath[curPos] = false;  
 }  
};