# 题目

给定一个非负整数数组，你最初位于数组的第一个位置。

数组中的每个元素代表你在该位置可以跳跃的最大长度。

判断你是否能够到达最后一个位置。

**示例 1:**

输入: [2,3,1,1,4]

输出: true

解释: 我们可以先跳1步，从位置0到达 位置1, 然后再从位置1跳3步到达最后一个位置。

**示例 2:**

输入: [3,2,1,0,4]

输出: false

解释: 无论怎样，你总会到达索引为3的位置。但该位置的最大跳跃长度是0，所以你永远不可能到达最后一个位置。

# 分析

## 方法一：贪心算法

class Solution {

public:

bool canJump(vector<int>& nums) {

int rightMax = 0;

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

{

if(i<=rightMax)//只要是i小于最右侧的最大值就执行

{

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

if(rightMax>=nums.size()-1)

{

return true;

}

}

}

return false;

}

};

**复杂度分析：**

时间复杂度：O(n)，其中n为数组的大小。只需要访问nums数组一遍，共n个位置。

空间复杂度：O(1)，不需要额外的空间开销。