55-跳跃游戏

题述

55. 跳跃游戏

难度 中等 1630

给定一个非负整数数组 `nums` ，你最初位于数组的 **第一个下标** 。

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下标。

示例 1：

```
输入：nums = [2,3,1,1,4]
输出：true
解释：可以先跳 1 步，从下标 0 到达下标 1，然后再从下标 1 跳 3 步到达最后一个下标。
```

示例 2：

```
输入：nums = [3,2,1,0,4]
输出：false
解释：无论怎样，总会到达下标为 3 的位置。但该下标的最大跳跃长度是 0 ， 所以永远不可能到达最后一个下标。
```

提示：

- $1 <= nums.length <= 3 * 10^4$
- $0 <= nums[i] <= 10^5$

通过次数 400,485 | 提交次数 920,843

思路

刚看到本题一开始可能想：当前位置元素如果是3，我究竟是跳一步呢，还是两步呢，还是三步呢，究竟跳几步才是最优呢？

其实跳几步无所谓，关键在于可跳的覆盖范围！

不一定非要明确一次究竟跳几步，每次取最大的跳跃步数，这个就是可以跳跃的覆盖范围。

这个范围内，别管是怎么跳的，反正一定可以跳过来。

那么这个问题就转化为跳跃覆盖范围究竟可不可以覆盖到终点！

每次移动取最大跳跃步数（得到最大的覆盖范围），每移动一个单位，就更新最大覆盖范围。

贪心算法局部最优解：每次取最大跳跃步数（取最大覆盖范围）

整体最优解：最后得到整体最大覆盖范围，看是否能到终点。

局部最优推出全局最优！

题解

C++：

```
class Solution {
public:
    bool canJump(vector<int>& nums)
    {
        //贪心
        //注意：数组中的元素对应的是最大条约步数  比如 为3 可以跳跃1、2、3步
        //局部最优：每次取最大跳跃步数 （取最大覆盖范围）
        //整体最优 最后得到整体最大覆盖范围 看能否覆盖到终点
        int cover = 0;
        if(nums.size()==1) return true; //只有一个元素的话肯定能跳跃到
        for(int i = 0; i <= cover; i++)
        {
            //注意  是小于等于cover
            cover = max(i+nums[i],cover);
            if(cover >= nums.size()-1) return true; //可以覆盖到终点
        }
        return false;
    }
};
```

Python

```python
class Solution:
    def canJump(self, nums: List[int]) -> bool:
        cover = 0
        if len(nums) == 1:
            return True
        i = 0
        # python不支持动态修改for循环中变量,使用while循环代替
        while i <= cover:
            cover = max(i + nums[i], cover)
            if cover >= len(nums) - 1:
                return True
            i += 1
        return False
```

提交结果	执行用时	内存消耗	语言	提交时间	备注
通过	116 ms	16 MB	Python3	2022/02/09 09:42	
通过	48 ms	47.2 MB	C++	2022/02/09 09:39	

思考

这道题目关键点在于：不用拘泥于每次究竟跳跳几步，而是看覆盖范围，覆盖范围内一定是可以跳过来的，不用管是怎么跳的。