Jump Game 总结

Jump Game 作为一个系列题，对理解贪婪算法、DFS 都有较好的体现，现总结与此。

1. Jump Game

Category	Difficulty	Likes	Dislikes
algorithms	Medium (32.18%)	2061	207

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.
```

解题思路1（O(n^2) 时间复杂度，O(1) 空间复杂度）：

代码思路比较清晰，因为能影响到跳动不能到达终点的只有出现在值为 0 的点，因此，直接针对这些点进行分情况讨论即可。

即只对数组中值为 0 的数进行判定即可，因为非 0 值代表始终可以有一个非零增量；

对值 0 进行判定的方法是：判定值为 0 的前面所有索引所能跳到的最远值是否可以跨越 0 值对应索引，若能，则证明这个 0 值不会造成跳动终止；若不能，则证明跳动截止到该 0 值；

```
class Solution {
public:
    bool canJump(vector<int>& nums) {
        int temp = 0;
        bool mark = true;
        for (int i = 0; i < nums.size(); ++ i) {
            if (nums[i] == 0) {
                for (int j = i - 1; j >= 0; j --) {
                    if (nums[j] + j > i) {
                        mark = false;
                        break;
                    }
                }
```

```
                if (mark) {
                    return false;
                }
            }
        }
        return true;

    }
};
```

解题思路2（O(n) 时间复杂度，O(1) 空间复杂度）：

上面那种方式的思路容易理解，但是时间复杂度太高，事实上，我们可以通过一遍遍历完成判定：

拿到遍历位置所能达到的最远位置，然后判定该位置是否可以到达终点，若能，则正能能达到终点，return true；同时，判定所能达到的最远位置对应跳数是否为 0，为 0 证明永远无法到达终点，return false；

```
class Solution {
public:
    bool canJump(vector<int>& nums) {
        // if (nums.size() <= 1) return true;
        int temp = 0;
        // bool mark = true;
        for (int i = 0; i < nums.size(); ++ i) {
            temp = max(temp, nums[i] + i);
            if (temp == nums.size() - 1) return true;
            else if (temp <= i && nums[i] == 0) {
                return false;
            }
        }
        return true;

    }
};
```

2. Jump Game II

Category	Difficulty	Likes	Dislikes
algorithms	Hard (27.92%)	1104	56

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.

Example:

```
Input: [2,3,1,1,4]
Output: 2
Explanation: 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.

解题思路：

这个题和 55. Jump Game很像，都是求怎么能跳到最后的位置，只不过这个题加了一个条件，就是我们需要的最小的步数，使用的策略是贪心。

我们每次贪心的找在自己当前能到达的几个位置里面，跳到哪个位置的时候，在下一步能跳的最远。然后，我们当前步就跳到这个位置上去，所以我们在这一步的跳跃时，给下一步留下了最好的结果。（因为题中已经说明，能到达的最远步只表示一个可跳跃范围，并非是固定的跳动步）

所以，使用一个 cur 表示当前步能到达的最远位置，使用 pre 表示上一次能到达的最远位置。所以，我们应该遍历在上一步的覆盖范围内，当前能跳的最远位置来更新 cur。

一个节省计算资源的方式是，保存以前已经检查了的位置为 index，这样每次检查的范围是 index~pre。

```
class Solution {
public:
    int jump(vector<int>& nums) {
        int cur = 0;
        int pre = 0;
        int index = 0;
        int count = 0;
        int length = nums.size();
        while (cur < length - 1) {
            pre = cur;
            count ++;
            while (index <= pre) {
                cur = max(cur, index + nums[index]);
                index ++;
            }

        }
        return count;
    }
};
```

参考文献

[1] https://blog.csdn.net/fuxuemingzhu/article/details/84578893

[2] https://www.cnblogs.com/grandyang/p/4371526.html

[3] https://www.cnblogs.com/grandyang/p/4373533.html