1. 题目
2. 题干

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。

1. 示例

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。

1. 题解
2. 思路
3. DFS

这种问题，拿到手就想到使用DFS递归+回溯的方法解决，首先，思考递归的结束条件是到达数组的最后一个位置，就需要设置flag为true，并且return。其次，思考进入下一个递归的条件，当前索引所在数组位置值不为0时，就可以继续往后走，即进入下一个递归，因此只要当前位置不为0就可以进入下一个递归。最后一个问题是如何回溯，若当前位置的数为0，则不能进入下一个递归，直接return到上一个递归的状态。

1. 贪心算法

还有一种非常简单的算法，贪心算法。首先，题目要求的是找到可以到达的最远位置，那么我们只需要维护可以到达的最远位置即可，如何获得可以到达的最远位置呢？每到一个位置，如果该位置不大于当前可以到达的最远位置，说明当前位置是可以到达的，因此可以继续更新最远位置。将当前的索引加上该位置的数就是该索引可以到达的最远位置，将该位置与之前获得的最远位置进行比较，如果更远，就更新最远位置。这样，一直循环下去，就能够找到最远的位置。若最远位置已经到达了数组的末尾，那么返回true。

1. 代码实现
2. DFS

JAVA：

class Solution {

    public boolean flag;

    public boolean canJump(int[] nums) {

        flag = false;

        int len = nums.length;

        if(len <= 1)        return true;

        dfs(nums, 0);

        return flag;

    }

    private void dfs(int[] nums, int start) {

        if(start >= nums.length - 1)

        {

            flag = true;

            return;

        }

        if(nums[start] == 0)        return;

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

        {

            int nextStart = start + i;

            dfs(nums, nextStart);

        }

    }

}

1. 贪心算法

Java：

class Solution {

    public boolean canJump(int[] nums) {

        int n = nums.length;

        int rightMax = 0;

        for(int i = 0; i < n; ++i){

            if(i <= rightMax){

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

                if(rightMax >= n-1){

                    return true;

                }

            }

        }

        return false;

    }

}