查阅更多的题解，请点击

Problem

45. Jump Game II(Hard)

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.

Solution

设输入大小为n

Corner Cases：

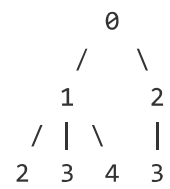
- n=0 或者 n=1，此时结果为0

O(n) time, O(1) space

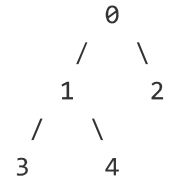
尝试reduce原问题，但是很难直接从子问题出发求解该问题，那么考虑最朴素的思想：

- 从起点出发，经一跳能到达的元素有哪些
- 从一跳能到达的元素出发，再一跳能达到的元素有哪些
- ……

上述过程在到达数组最后一个元素时终止，这样的搜索过程很像一棵树，对于例子[2,3,1,1,4]，可以给出以下的搜索过程(结点标号为元素下标，一个结点的子节点为从该结点出发，经一跳能到达的结点下标)：



从上述搜索树中可以看出经两步能到达数组最后一个元素，注意在上述搜索过程中，1的子节点已经在上一层结点中出现，再次搜索没有任何意义，所以可以简化如下



上述搜索过程是一个BFS的过程：**设某层搜索的元素下标区间为$[start, end]$，该层元素能到达的最远元素下标为$maxIndex$，则下一层需要搜索元素下标区间为$[end+1, maxIndex]$**

```
class Solution
{
public:
    int jump(vector<int> &nums)
    {
        if (nums.empty() || nums.size() == 1)
            return 0;
        int lastIndex = nums.size() - 1;
        int end = 0, start = 0, step = 0, maxIndex = 0;
        while (end < lastIndex)
        {
            step++;
            for (int i = start; i <= end; ++i)
            {
                if (i + nums[i] >= lastIndex)
                    return step;
                maxIndex = max(i + nums[i], maxIndex);
            }
            start = end + 1;
            end = maxIndex;
        }
        return step;
    }
};
```

上述过程也可以利用Greedy的思想来分析：Concise O(n) one loop JAVA solution based on Greedy，尽管这样的分析方式会很难让人接受。

GitHub传送门