贪心：

1. 跳跃游戏

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。

对于数组中的任意一个位置 y，我们如何判断它是否可以到达？

根据题目的描述，只要存在一个位置 x，它本身可以到达，并且它跳跃的最大长度为 x+nums[x]，这个值大于等于 y，即 x+nums[x]≥y，那么位置 y 也可以到达。

换句话说，对于每一个可以到达的位置 x，它使得x+1,x+2,⋯,x+nums[x] 这些连续的位置都可以到达。

这样以来，我们依次遍历数组中的每一个位置，并实时维护 最远可以到达的位置。对于当前遍历到的位置 x，如果它在 最远可以到达的位置 的范围内，那么我们就可以从起点通过若干次跳跃到达该位置，因此我们可以用x+nums[x] 更新 最远可以到达的位置。

在遍历的过程中，如果 最远可以到达的位置 大于等于数组中的最后一个位置，那就说明最后一个位置可达，我们就可以直接返回 True 作为答案。反之，如果在遍历结束后，最后一个位置仍然不可达，我们就返回 False 作为答案

·bool canJump(vector<int>& nums) {

        int n=nums.size();

        int rightmost=0;

        for(int i=0;i<n;i++){

            if(i<=rightmost){

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

            }

            if(rightmost>=n-1)

                return true;

        }

        return false;

    };

1. 跳跃游戏2

你一个非负整数数组 nums ，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。

你的目标是使用最少的跳跃次数到达数组的最后一个位置。

假设你总是可以到达数组的最后一个位置。

如果我们「贪心」地进行正向查找，每次找到可到达的最远位置，就可以在线性时间内得到最少的跳跃次数。

例如，对于数组 [2,3,1,2,4,2,3]，初始位置是下标 0，从下标 0 出发，最远可到达下标 2。下标 0 可到达的位置中，下标 1 的值是 3，从下标 1 出发可以达到更远的位置，因此第一步到达下标 1。

从下标 1 出发，最远可到达下标 4。下标 1 可到达的位置中，下标 4 的值是 4 ，从下标 4 出发可以达到更远的位置，因此第二步到达下标 4。

每一步都选择可以走最远下标的点

·int jump(vector<int>& nums) {

        int count=0;//跳跃次数

        int n=nums.size();

        int end=0;

        int rightmost=0;//初始跳跃的最大长度

        for(int i=0;i<n-1;i++){

            if(i<=rightmost)

            {

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

                if(i==end){

                    count++;

                    end=rightmost;

                 }

            }

        }

        return count;

    }