Problem 45: Jump Game II

Problem Information

Difficulty: Medium
Acceptance Rate: 42.14%
Paid Only: No
Tags: Array, Dynamic Programming, Greedy

Problem Description

You are given a **0-indexed** array of integers `nums` of length `n`. You are initially positioned at index 0.

Each element `nums[i]` represents the maximum length of a forward jump from index `i`. In other words, if you are at index `i`, you can jump to any index `(i + j)` where:

* `0 <= j <= nums[i]` and * `i + j < n`

Return _the minimum number of jumps to reach index_`n - 1`. The test cases are generated such that you can reach index `n - 1`.

Example 1:

Input: nums = [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.

Example 2:

Input: nums = [2,3,0,1,4] **Output:** 2

Constraints:

* `1 <= nums.length <= 104` * `0 <= nums[i] <= 1000` * It's guaranteed that you can reach `nums[n - 1]`.

Code Snippets

C++:

```
class Solution {
public:
int jump(vector<int>& nums) {

}
};
```

Java:

```
class Solution {
public int jump(int[] nums) {

}
}
```

Python3:

```
class Solution:
def jump(self, nums: List[int]) -> int:
```