Problem 674: Longest Continuous Increasing Subsequence

Problem Information

Difficulty: Easy
Acceptance Rate: 51.58%
Paid Only: No
Tags: Array

Problem Description

Given an unsorted array of integers `nums`, return _the length of the longest**continuous increasing subsequence** (i.e. subarray)_. The subsequence must be **strictly** increasing.

A **continuous increasing subsequence** is defined by two indices `l` and `r` (`l < r`) such that it is `[nums[l], nums[l + 1], ..., nums[r - 1], nums[r]]` and for each `l <= i < r`, `nums[i] < nums[i + 1]`.

Example 1:

Input: nums = [1,3,5,4,7] **Output:** 3 **Explanation:** The longest continuous increasing subsequence is [1,3,5] with length 3. Even though [1,3,5,7] is an increasing subsequence, it is not continuous as elements 5 and 7 are separated by element 4.

Example 2:

Input: nums = [2,2,2,2,2] **Output:** 1 **Explanation:** The longest continuous increasing subsequence is [2] with length 1. Note that it must be strictly increasing.

Constraints:

* `1 <= nums.length <= 104` * `-109 <= nums[i] <= 109`

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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