

## Longest Continuous Increasing Subsequence

Given an unsorted array of integers, find the length of longest continuous increasing subsequence.

### Example 1:

**Input:** [1,3,5,4,7]

**Output:** 3

**Explanation:** The longest continuous increasing subsequence is [1,3,5], its length is 3.

Even though [1,3,5,7] is also an increasing subsequence, it's not a continuous one where 5 and 7 are separated by 4.

### Example 2:

**Input:** [2,2,2,2,2]

**Output:** 1

**Explanation:** The longest continuous increasing subsequence is [2], its length is 1.

**Note:** Length of the array will not exceed 10,000.

## Solution 1

The idea is to use `cnt` to record the length of the current continuous increasing subsequence which ends with `nums[i]`, and use `res` to record the maximum `cnt`.

Java version:

```
public int findLengthOfLCIS(int[] nums) {  
    int res = 0, cnt = 0;  
    for(int i = 0; i < nums.length; i++){  
        if(i == 0 || nums[i-1] < nums[i]) res = Math.max(res, ++cnt);  
        else cnt = 1;  
    }  
    return res;  
}
```

C++ version:

```
int findLengthOfLCIS(vector<int>& nums) {  
    int res = 0, cnt = 0;  
    for(int i = 0; i < nums.size(); i++){  
        if(i == 0 || nums[i-1] < nums[i]) res = max(res, ++cnt);  
        else cnt = 1;  
    }  
    return res;  
}
```

written by [Vincent Cai](#) original link [here](#)

## Solution 2

A continuous subsequence is essentially a subarray. Hence this question is asking for the longest increasing subarray and I have no idea why the question calls it continuous subsequence to confuse the readers.

Anyway, we can make one pass of the array and keep track of the current streak of increasing elements, reset it when it does not increase.

*By Yang Shun*

```
class Solution(object):
    def findLengthOfLCIS(self, nums):
        """
        :type nums: List[int]
        :rtype: int
        """
        # Time: O(n)
        # Space: O(1)
        max_len = i = 0
        while i < len(nums):
            curr = 1
            while i + 1 < len(nums) and nums[i] < nums[i + 1]:
                curr, i = curr + 1, i + 1
            max_len = max(max_len, curr)
            i += 1
        return max_len
```

written by [yangshun](#) original link [here](#)

## Solution 3

```
class Solution {
    public int findLengthOfLCIS(int[] nums) {
        if (nums == null || nums.length == 0) return 0;
        int n = nums.length;
        int[] dp = new int[n];

        int max = 1;
        dp[0] = 1;
        for (int i = 1; i < n; i++) {
            if (nums[i] > nums[i - 1]) {
                dp[i] = dp[i - 1] + 1;
            }
            else {
                dp[i] = 1;
            }
            max = Math.max(max, dp[i]);
        }

        return max;
    }
}
```

written by [shawngao](#) original link [here](#)

From [LeetCoder](#).