674 Longest Continuous Increasing Subsequence

2018年4月23日 16:38

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

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.

来自 <https://leetcode.com/problems/longest-continuous-increasing-subsequence/description/>

给定一个未经排序的整数数组，找到最长且**连续**的的递增序列。

注意：数组长度不会超过10000。

Solution for Python3:

```python
class Solution:
    def findLengthOfLCIS(self, nums):
        """
        :type nums: List[int]
        :rtype: int
        """
        maxL = i = 0
        while i < len(nums):
            cnt = 1
            while i + 1 < len(nums) and nums[i] < nums[i+1]:
                cnt += 1
                i += 1
            maxL = max(maxL, cnt)
            i += 1
        return maxL
```

Solution for C++:

```cpp
class Solution1 {
public:
```

```
    int findLengthOfLCIS(vector<int>& nums) {
        int maxL = 0, cnt = 1, i = 0;
        while (i < nums.size()) {
            while (i + 1 < nums.size() && nums[i+1] >
nums[i]) {
                cnt++;
                i++;
            }
            maxL = max(maxL, cnt);
            cnt = 1;
            i++;
        }
        return maxL;
    }
};

class Solution2 {
public:
    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;
    }
};
```