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:

```java
    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:

```cpp
    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 they wanna call 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

C++ record length

```
class Solution {
public:
    int findLengthOfLCIS(vector<int>& a) {
        int mx = 0, len = 0;
        for (int i = 0; i < a.size(); i++) {
            if (i == 0 || a[i] <= a[i - 1]) len = 0;
            mx = max(mx, ++len);
        }
        return mx;
    }
};
```

C++ 2 pointer

```
class Solution {
public:
    int findLengthOfLCIS(vector<int>& a) {
        int mx = 0;
        for (int i = 0, j = 0; j < a.size(); j++) {
            if (j == 0 || a[j] <= a[j - 1]) i = j;
            mx = max(mx, j - (i - 1))
        }
        return mx;
    }
};
```

3 liner

```
class Solution {
public:
    int findLengthOfLCIS(vector<int>& a) {
        int mx = 0;
        for (int i = 0, j = 0; j < a.size(); i = (j == 0 || a[j] <= a[j - 1]) ? j :
i, mx = max(mx, j - (i - 1)), j++) { }
        return mx;
    }
};
```

Java - 2 pointer

```
class Solution {
    public int findLengthOfLCIS(int[] a) {
        int mx = 0;
        for (int i = 0, j = 0; j < a.length; i = (j == 0 || a[j] <= a[j-1]) ? j : i
, mx = Math.max(mx, j-i+1), j++) { }
        return mx;
    }
}
```

Java length variable

```java
class Solution {
    public int findLengthOfLCIS(int[] a) {
        int mx = 0, len = 0;
        for (int i = 0; i < a.length; i++) {
            if (i == 0 || a[i] <= a[i - 1]) len = 0;
            mx = Math.max(mx, ++len);
        }
        return mx;
    }
}
```

written by alexander original link here