最长连续递增序列

给定一个未经排序的整数数组，找到最长且 **连续递增的子序列**，并返回该序列的长度。

连续递增的子序列 可以由两个下标 `l` 和 `r`（`l < r`）确定，如果对于每个 `l <= i < r`，都有 `nums[i] < nums[i + 1]` ，那么子序列 `[nums[l], nums[l + 1], ..., nums[r - 1], nums[r]]` 就是连续递增子序列。

示例 1：

输入： `nums = [1,3,5,4,7]`

输出： `3`

解释： 最长连续递增序列是 `[1,3,5]`, 长度为 `3`。

尽管 `[1,3,5,7]` 也是升序的子序列，但它不是连续的，因为 `5` 和 `7` 在原数组里被 `4` 隔开。

示例 2：

输入： `nums = [2,2,2,2,2]`

输出： `1`

解释： 最长连续递增序列是 `[2]`, 长度为 `1`。

```
int max(int a,int b)
{
    return a>b?a:b;
}
int findLengthOfLCIS(int* nums, int numsSize){
    int cout=1;//记录所有遍历到递增滑窗的长度,滑窗长度最少为 1
    int maxlen=0;//记录最长的滑窗长度
    //特殊情况剔除
    if(numsSize==0)
    {
        return 0;
    }
    for(int i=0;i<numsSize-1;i++)
    {
        if(nums[i]<nums[i+1])//递增计数
        {
            cout++;
            continue;
        }
```

```
            //遇到递减情况
            //更新此时的 maxlen  同时将 cout 复位为 1
            maxlen=max(cout,maxlen);
            cout=1;
        }
        //如果最后一个滑窗为最长 还需要将 cout 进行更新 maxlen
        maxlen=max(cout,maxlen);
        return maxlen;
    }
```