给定一个未排序的整数数组，找出最长连续序列的长度。

要求算法的时间复杂度为 *O(n)*。

示例:

输入: [100, 4, 200, 1, 3, 2]
输出: 4
解释: 最长连续序列是 [1, 2, 3, 4]。它的长度为 4。

思路
题目会给出一个数组，数组里面包含了一些数字，现在需要用O（n）的时间找出这些数字里面，能够成连续区间的最长长度是多少。

这道题比较难的地方在于，要求在常数时间解题。

解题思想呢也是用HashMap，HashMap保存某个数（在作为边界的情况下）及其对应的最长连续长度：
1、假设我们新过来一个数，我们首先保证他没有被处理过，这个后面的步骤需要保证，即每个数只处理一次。
2、对于没处理过的数，我们在HashMap中寻找其左边，其右边是否存在，存在的话返回其长度，不存在则置0
3、当前数的位置的长度，就是步骤2中找到的两个长度的和，加一
4、同时需要找到当前位置，最左边的边界和最右边的边界，根据步骤2的长度就可以，然后也更新为3中计算得到的当前长度的位置。
5、至于最左边最右边和当前位置之外的，是不是感觉中间的那些长度没有被更新？恩其实，如果不作为边界，那么他们是不会再被访问到的（1中保证了），所以这下就懂了吧

作者：MebiuW
来源：CSDN
原文：https://blog.csdn.net/MebiuW/article/details/53886129?utm_source=copy

代码

```
class Solution {
public:
    int longestConsecutive(vector<int>& nums) {
         unordered_map<int,int> len;
         int max=0;
         for(auto i:nums)
         {
             if(len[i]==0)//避免重复元素
            {
                 int l=len[i-1],r=len[i+1];
                 len[i]=l+r+1;
                 len[i+r]=l+r+1;
                 len[i-l]=l+r+1;
```

```
                max=max>len[i]?max:len[i];
            }
        }
        return max;
    }
};
```