# 题目

给定一个未排序的整数数组 nums ，找出数字连续的最长序列（不要求序列元素在原数组中连续）的长度。

请你设计并实现时间复杂度为 O(n) 的算法解决此问题。

示例 1：

输入：nums = [100,4,200,1,3,2]

输出：4

解释：最长数字连续序列是 [1, 2, 3, 4]。它的长度为 4。

示例 2：

输入：nums = [0,3,7,2,5,8,4,6,0,1]

输出：9

提示：

0 <= nums.length <= 10^5

-10^9 <= nums[i] <= 10^9

# 分析

## 方法一：哈希表

思路：

为了实现 O(n) 的时间复杂度，可以利用哈希表来存储每个数字的信息。具体步骤如下：

1、首先将数组中的所有数字存入一个哈希表中，以便快速查找某个数字是否存在。

2、然后遍历数组中的每个数字，对于每个数字，判断其是否为一个连续序列的起点，即判断其前一个数字（num-1）是否在哈希表中存在。如果不存在，则该数字为一个连续序列的起点。

3、对于每个连续序列的起点，向后查找连续的数字，直到不连续为止，统计连续序列的长度。

4、在统计过程中，更新最长连续序列的长度。

代码：

class Solution {

public:

int longestConsecutive(vector<int>& nums) {

unordered\_set<int> num\_set(nums.begin(), nums.end());

int longest\_streak = 0;

for (int num : num\_set) {

// 判断当前数字是否为连续序列的起点

if (num\_set.find(num - 1) == num\_set.end()) {

int current\_num = num;

int current\_streak = 1;

// 统计连续序列的长度

while (num\_set.find(current\_num + 1) != num\_set.end()) {

current\_num++;

current\_streak++;

}

// 更新最长连续序列的长度

longest\_streak = max(longest\_streak, current\_streak);

}

}

return longest\_streak;

}

};