0128. 最长连续序列

ITCharge 大约 2 分钟

- 标签：并查集、数组、哈希表
- 难度：中等

题目链接

- 0128. 最长连续序列 - 力扣

题目大意

描述：给定一个未排序的整数数组 `nums` 。

要求：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。并且要用时间复杂度为 $O(n)$ 的算法解决此问题。

说明：

- $0 \leq nums.length \leq 10^5$。
- $-10^9 \leq nums[i] \leq 10^9$。

示例：

- 示例 1：

```py
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：

```py
输入：nums = [0,3,7,2,5,8,4,6,0,1]
输出：9
```

解题思路

暴力做法有两种思路。

- 第 1 种思路是先排序再依次判断，这种做法时间复杂度最少是 $O(n \log_2 n)$。
- 第 2 种思路是枚举数组中的每个数 `num`，考虑以其为起点，不断尝试匹配 `num + 1`、`num + 2`、`...` 是否存在，最长匹配次数为 `len(nums)`。这样下来时间复杂度为 $O(n^2)$。

我们可以使用哈希表优化这个过程。

思路 1：哈希表

1. 先将数组存储到集合中进行去重，然后使用 `curr_streak` 维护当前连续序列长度，使用 `ans` 维护最长连续序列长度。
2. 遍历集合中的元素，对每个元素进行判断，如果该元素不是序列的开始（即 `num - 1` 在集合中），则跳过。
3. 如果 `num - 1` 不在集合中，说明 `num` 是序列的开始，判断 `num + 1`、`nums + 2`、`...` 是否在哈希表中，并不断更新当前连续序列长度 `curr_streak`。并在遍历结束之后更新最长序列的长度。
4. 最后输出最长序列长度。

思路 1：代码

```py
class Solution:
    def longestConsecutive(self, nums: List[int]) -> int:
        ans = 0
        nums_set = set(nums)
        for num in nums_set:
            if num - 1 not in nums_set:
                curr_num = num
                curr_streak = 1

                while curr_num + 1 in nums_set:
```

```
                curr_num += 1
                curr_streak += 1
            ans = max(ans, curr_streak)

    return ans
```

思路 1：复杂度分析

- **时间复杂度**：$O(n)$。将数组存储到集合中进行去重的操作的时间复杂度是 $O(n)$。查询每个数是否在集合中的时间复杂度是 $O(1)$，并且跳过了所有不是起点的元素。更新当前连续序列长度 `curr_streak` 的时间复杂度是 $O(n)$，所以最终的时间复杂度是 $O(n)$。
- **空间复杂度**：$O(n)$。

参考资料

- 【题解】128. 最长连续序列 - 力扣（Leetcode）