Leetcode 128. 最长连续序列

宋星霖

2024 年 2 月 20 日

1 题目描述

1.1 题目描述

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。

1.2 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`。

1.3 示例 2

输入：nums = [0,3,7,2,5,8,4,6,0,1]

输出：9

1.4 提示

- $0 \leqslant nums.length \leqslant 10^5$
- $-10^9 \leqslant nums\,[i] \leqslant 10^9$

2 算法分析

对于这个题目，可以这样做

2.1 第 1 步：构建一个并查集

并查集需要一个 fa 数组，用于记录每一个节点的父节点。在此笔者与读者约定，如果一个节点的父节点是它本身，说明这个节点是根节点。

对于这道题目，还需要一个 size 数组，用于记录每一个节点拥有的节点数量。

并查集还需要两个方法：查找和合并。在查找时我们可以使用路径压缩。我么可以将自己的父节点编号变为递归下来的父节点编号。

并查集的关键代码如下

Listing 1: 并查集

```
class UnionSet {
    public:
    UnionSet(int n):fa(n),size(n) {
        for(int i = 0;i < n;i++) {
            fa[i] = i;
            size[i] = 1;
        }
    }
    int find(int x) {
        return fa[x] = (fa[x] == x ? x : find(fa[x]));
    }
    void merge(int a,int b) {
        int aa = find(a),bb = find(b);
        if(aa == bb) return;
        fa[aa] = bb;
        size[bb] += size[aa];
    }
    vector<int> fa,size;
};
```

2.2 第 2 步：使用一个哈希表

哈希表的作用有两个：

- 将每一个数字映射到下标，作用是方便查找和存储。
- 查找已经处理过的数据

2.3 第 3 步：查找、合并

遍历数组，给当前遍历的数字分配下标。如果能与当前遍历的数字相连的数字已经处理过了，则合并两个数字。如果当前处理的数字已经处理过了，跳过这一位。

2.4 第 4 步：找到集合最大值

‘在所有集合中找的最大大小的集合，返回他的大小。

3 代码演示

，

```
class UnionSet {
    public:
    UnionSet(int n):fa(n),size(n) {
        for(int i = 0;i < n;i++) {
            fa[i] = i;
            size[i] = 1;
        }
    }
    int find(int x) {
        return fa[x] = (fa[x] == x ? x : find(fa[x]));
    }
    void merge(int a,int b) {
        int aa = find(a),bb = find(b);
        if(aa == bb) return;
        fa[aa] = bb;
        size[bb] += size[aa];
    }
    vector<int> fa,size;
};

class Solution {
    public:
    int longestConsecutive(vector<int>& nums) {
        int size = nums.size(),cnt = 0;
        UnionSet u(size);
        unordered_map<int,int> m;
        for(int i = 0;i < size;i++) {
            int x = nums[i];
            if(m.find(x) != m.end()) continue;
            m[x] = cnt++;
            if(m.find(x - 1) != m.end()) {
                u.merge(m[x],m[x - 1]);
            }
            if(m.find(x + 1) != m.end()) {
                u.merge(m[x],m[x + 1]);
            }
        }
        int ans = 0;
        for(int i:u.size) {
            ans = max(ans,i);
        }
        return ans;
    }
};
```