128 Longest Consecutive Sequence

128. Longest Consecutive Sequence

1. Question

Given an unsorted array of integers, find the length of the longest consecutive elements sequence.

For example,
Given `[100, 4, 200, 1, 3, 2]`,
The longest consecutive elements sequence is `[1, 2, 3, 4]`. Return its length: `4`.

Your algorithm should run in O(n) complexity.

2. Implementation

(1) Hash Table + Union Find

```
class Solution {
    public int longestConsecutive(int[] nums) {
        Map<Integer, Integer> map = new HashMap<>();
        int n = nums.length;

        UnionFind uf = new UnionFind(n);

        for (int i = 0; i < nums.length; i++) {
            if (map.containsKey(nums[i])) {
                continue;
            }

            map.put(nums[i], i);

            if (map.containsKey(nums[i] - 1)) {
                uf.union(i, map.get(nums[i] - 1));
            }

            if (map.containsKey(nums[i] + 1)) {
                uf.union(i, map.get(nums[i] + 1));
            }
        }
        return uf.maxSize();
    }

    class UnionFind {
        int[] sets;
        int[] size;
        int count;

        public UnionFind(int n) {
            sets = new int[n];
            size = new int[n];
            count = n;

            for (int i = 0; i < n; i++) {
                sets[i] = i;
                size[i] = 1;
            }
        }

        public int find(int node) {
            while (node != sets[node]) {
                node = sets[node];
            }
            return node;
        }

        public void union(int i, int j) {
            int node1 = find(i);
            int node2 = find(j);

            if (node1 == node2) {
                return;
            }

            if (size[node1] < size[node2]) {
                sets[node1] = node2;
                size[node2] += size[node1];
            }
            else {
                sets[node2] = node1;
                size[node1] += size[node2];
            }
            --count;
        }

        public int maxSize() {
            int res = 0;
            for (int i = 0; i < size.length; i++) {
                res = Math.max(res, size[i]);
            }
            return res;
        }
    }
}
```

3. Time & Space Complexity

时间复杂度O(n), 空间复杂度O(n)

Previous
Union Find

Next
130 Surrounded Regions