

LeetCode

Explore

Problems

Interview

NewContest

Discuss

Store

Premium

Quick Find - Disjoint Set

Quick Union - Disjoint Set

Union by Rank - Disjoint Set

Path Compression Optimiz...

Optimized "disjoint set" wi...

Summary of the "disjoint s...

Number of Provinces

LeetCode 547 - Number of...

Graph Valid Tree

Number of Connected Co...

The Earliest Moment When...

Previous

Next

Optimized "disjoint set" with Path Compression and Union by Rank

Report Issue

Optimized "disjoint set" with Path Compression and Union by Rank

This implementation of the "disjoint set" is optimized with both "path compression" and "union by rank".

Implementation

C++JavaPython3

CopyRunPlayground

```
1 class UnionFind {
2 public:
3     UnionFind(int sz) : root(sz), rank(sz) {
4         for (int i = 0; i < sz; i++) {
5             root[i] = i;
6             rank[i] = 1;
7         }
8     }
9
10    int find(int x) {
11        if (x == root[x]) {
12            return x;
13        }
14        return root[x] = find(root[x]);
15    }
16
17    void unionSet(int x, int y) {
18        int rootX = find(x);
19        int rootY = find(y);
20        if (rootX != rootY) {
21            if (rank[rootX] > rank[rootY]) {
22                root[rootY] = rootX;
23            } else if (rank[rootX] < rank[rootY]) {
24                root[rootX] = rootY;
25            } else {
26                root[rootY] = rootX;
27                rank[rootX] += 1;
28            }
29        }
30    }
31 }
```

Time Complexity

| | Union-find Constructor | Find | Union | Connected |
|-----------------|------------------------|----------------|----------------|----------------|
| Time Complexity | $O(N)$ | $O(\alpha(N))$ | $O(\alpha(N))$ | $O(\alpha(N))$ |

Note: N is the number of vertices in the graph. α refers to the Inverse Ackermann function. In practice, we assume it's a constant. In other words, $O(\alpha(N))$ is regarded as $O(1)$ on average.

For the union-find constructor, we need to create two arrays of size N each.

When using the combination of union by rank and the path compression optimization, the find operation will take $O(\alpha(N))$ time on average. Since union and connected both make calls to find and all other operations require constant time, union and connected functions will also take $O(\alpha(N))$ time on average.

Space Complexity

We need $O(N)$ space to store the array of size N .