【DS】Day1(3)

Tags	
Date	@May 19, 2022
Summary	Optimization: Weighting and Path Compression

【Week1】Union-Find

1.4 Quick-Union Improvements

1.4.1 Weighting

- Modify quick-union to avoid tall trees
- Keep track of size of each tree(number of objects)
- Balance by linking root of smaller tree to root of larger tree

Data Structure: Same as quick-union, but maintain extra array `sz[i]` to count number of objects in the tree rooted at i.

Find: Identical to quick-union

Union: Modify quick-union to

- Link root of smaller tree to root of large tree
- Update the `sz[]` array

```
int i = root(p);
int j = root(q);

if (i == j) return;
if (sz[i] < sz[j]) {
  id[i] = j;
  sz[j] += sz[i];
} else {
  id[j] = i;
  sz[i] += sz[j];
}
```

1.4.2 Path Compression

Quick union with path compression: Just after computing the root of p, set the id of each examined node to point to that root.

Simple one-pass variant: Make every other node in path point to its grandparent:

```
private int root(int i) {
  while (i != id[i]) {
    id[i] = id[id[i]];
    i = id[i];
  }
  return i;
}
```