Sets with Union/Find

Given a set of elements (singleton sets).

Operations:

- *Union:* of 2 sets.
- *Find:* return the name of the set that contains the given element.
- *Make-set(x):* creates $\{x\}$.

One approach, with pointers linking the same set names in the following:

Elements:	A	B	C	D	E	F	G	H
Set names:	1	2	1	3	4	1	3	3

Thus the complexity is:

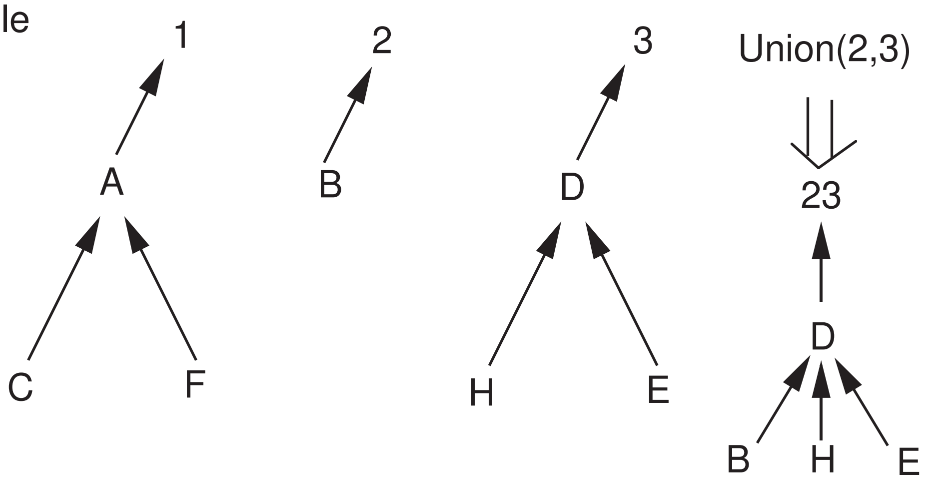
• *Find* –constant time (given element)
• *Union*–size of one of the sets (to change names)

{ Amortized worst case: for n operations, $\Theta(n \log n)$ if we make sure the union cost is the smaller set size. }

Another approach

- make "Find" harder, "Union" trivial.
- A set is represented by a <u>tree</u> with edges oriented toward root. (root points to name of set).
- Union - root of smaller tree becomes child of root of the larger.
- Find - follow path to root to name of set.

Example



1 = {A,C,F} 2 = {B} 3 = {D,E,H}

Claim. Height of any tree of size d is at most $\log d$.

Proof: By induction. Suppose the claim is true for all $d < n$. i.e. all trees whose size $d < n$, has height at most $\log d$. (This is true for $d = 1$.)

Consider first time we obtain a tree of size n from 2 trees of sizes $d_1, d_2 < n$. These 2 trees have heights at most $\log d_1, \log d_2$.

If the new tree does not grow in height, then the claim is true. We only need to consider the case when the resulting tree grows in height (by 1).

Because of the "rules" for union—we hook smaller tree to bigger tree, the resulting tree height grows (by 1) only when both trees have the same height h. But by inductive hypothesis, $h \leq \log d_1, \log d_2$. Hence

$$h + 1 \leq \log(d_1 + d_2) = \log n.$$

Thus the resulting tree of size n has height at most $\log n$. $\square$

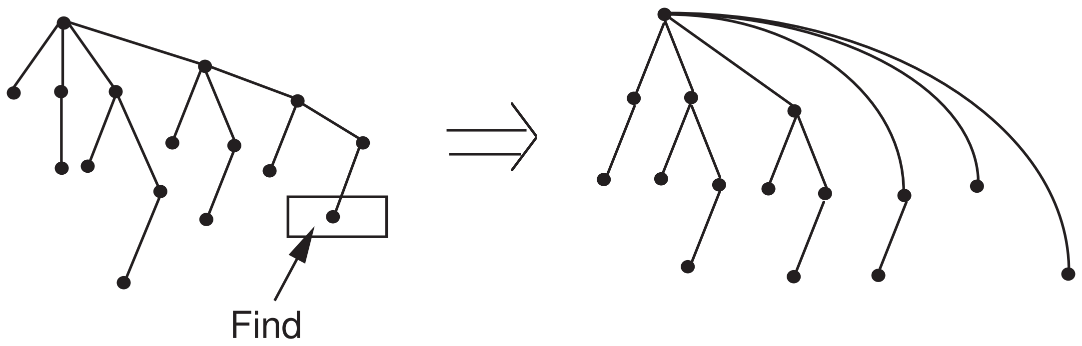
Thus with above approach, we have

- Union: $O(1)$
- Find: $O(\log n)$ (tree height).

We can still do better by **Path Compression**:

When we do a find, we make second pass and change all pointers to refer directly to root.

Example.



Union/Find with path compression:

Claim. A sequence of n union/find operations cost $O(n\alpha(n))$, where $\alpha(n) = \#$ of times you take logarithm of n to get down to 1. $\alpha(n) < 5$ for all practical purposes. (Proof omitted)