Union/Find with “up trees”

Each set is implemented as a tree

Every node in the tree, other than the root, has a pointer “up” to its parent

The representative of a set is the root of its tree

Find: follow pointers to the root

Union: Merge one root into the other root
Example of Union:

Assume $n$ items and $m$ operations

Worst case: $mn$ — trees may degenerate into lists

Two simple ideas: size balancing and path compression
Size balancing

**Size balancing rule**

In a Union operation, always merge the smaller tree into the larger tree.

**Lemma 1**

If $T$ is a tree created by balanced merges, and $T$ has size $n$ and height $h$, then $n \geq 2^h$.
Proof: induction on $n$.

Assume $T$ was obtained by merging $T_1$ into $T_2$

where $n_1 := \text{Size}(T_1) \leq n_2 := \text{Size}(T_2)$

Let $h_i := \text{Height}(T_i)$ for $i = 1, 2$

By induction, $n_1 \geq 2^{h_1}$ and $n_2 \geq 2^{h_2}$

If $h_1 \geq h_2$, then $h = h_1 + 1$ and

$$n = n_1 + n_2 \geq 2n_1 \geq 2 \cdot 2^{h_1} = 2^h$$

If $h_1 < h_2$, then $h = h_2$ and

$$n = n_1 + n_2 \geq n_2 \geq 2^{h_2} = 2^h$$

QED
Path compression

Path compression rule

After each Find operation, make all nodes visited point to the root of the tree
Running time analysis

For $g \geq 0$, define

$$F(g) := 2^{2^2 \ldots 2} \quad \text{g 2's}$$

Formally, $F(0) := 1$ and $F(g + 1) := 2^{F(g)}$

Define $\log^* r := \text{least } g \text{ such that } F(g) \geq r$

**Theorem**

With size balancing and path compression, any sequence of $m$ union/find operations on $n$ items takes time $O((m + n) \log^* n)$
\[
R_g := (\log^*)^{-1}(g) = \{ r : \log^* r = g \}
\]
\[
\log^* r \leq g \iff r \leq F(g)
\]
\[
R_0 = \{0, 1\}, \ R_g = \{F(g - 1) + 1, \ldots, F(g)\} \text{ for } g > 0
\]
Let $Op_1, \ldots, Op_m$ be a sequence of union/find operations.

Consider the forest of trees $\mathcal{F}$ that results after executing $Op_1, \ldots, Op_m$ with size balancing, but no path compression.

Define the rank of a node $v$ to be its height in $\mathcal{F}$.

Rank is a static quantity – it does not change over time.
Lemma 2

For every $r \geq 0$, there are at most $n/2^r$ nodes of rank $r$

Proof:

- By Lemma 1, any node of rank $r$ is the root of a subtree in $\mathcal{F}$ of size $\geq 2^r$
- Any two distinct nodes of rank $r$ are roots of disjoint subtrees in $\mathcal{F}$
- Therefore, there can be at most $n/2^r$ nodes of rank $r$
- QED
Lemma 3

Suppose that at some time during the execution of $Op_1, \ldots, Op_m$ with compression, $\nu$ is a (strict) descendent of $\omega$. Then $Rank(\nu) < Rank(\omega)$

Proof:

• Key observations:
  - path compression only eliminates descendency relations – it never creates any new ones
  - with no path compression, union/find operations never destroy descendency relations
Prove by induction on $i$:

- for all $v, w$, if $v$ is a descendent of $w$ after executing $Op_1, \ldots, Op_i$ with compression, then $v$ is a descendent of $w$ after executing $Op_1, \ldots, Op_i$ without compression.

Thus, if $v$ is a descendent of $w$ at some point in time during the execution of $Op_1, \ldots, Op_m$ with compression, then $v$ is a descendent of $w$ in $F$, and hence $Rank(v) < Rank(w)$ – QED

**Definition**

For a node $v$, we define its **group** as $G(v) := \log^* Rank(v)$

Clearly, $G(v) \leq \log^* n$
Proof of Theorem

Union operations take $O(1)$, so we can focus on find operations.

Let $\mathcal{I}$ be the set of indices $i$ such that $Op_i$ is a find operation.

Consider a fixed $i \in \mathcal{I}$, with $Op_i = \text{“Find}(\nu)\text{”}$

Consider the path from $\nu$ to the root:

$\nu = \nu_1, \nu_2, \ldots, \nu_{k-2}, \nu_{k-1}, \nu_k = \text{root}$

By Lemma 3, we have

$Rank(\nu_1) < Rank(\nu_2) < \cdots < Rank(\nu_k)$

$G(\nu_1) \leq G(\nu_2) \leq \cdots \leq G(\nu_k)$
Let $X_i = \{v_1, \ldots, v_k\}$

$C := \sum_{i \in I} |X_i|$ is the cost of all the find operations

Let’s split $X_i$ into 3 sets:

- $Y_i := \{v_j : j < k - 1 \text{ and } G(v_j) = G(v_{j+1})\}$
- $Z_i := \{v_j : j < k - 1 \text{ and } G(v_j) < G(v_{j+1})\}$
- $W_i := \{v_j : j \geq k - 1\}$

We have

- $|Z_i| \leq G(v_k) \leq \log^* n$
- $|W_i| \leq 2$
So we have

\[ C = \sum_{i \in I} (|Y_i| + |Z_i| + |W_i|) \leq \sum_{i \in I} |Y_i| + m \log^* n + 2m \]

**Claim:** \( C' := \sum_i |Y_i| \leq n \log^* n \)

**Idea:** Consider a fixed node \( \nu \)

- Each time \( \nu \) moves during a path compression, \( \nu \)'s new parent has a higher rank than \( \nu \)'s old parent
- If \( G(\nu) = g \), then after \( |R_g| - 1 \) moves, \( \nu \) must acquire a parent whose group is \( > g \)
For $g \geq 0$, let $V_g := \{ \nu : G(\nu) = g \}$

We have

$$C' \leq \sum_{g=0}^{\log^* n} |V_g| \cdot (|R_g| - 1)$$

$$\leq n + \sum_{g=2}^{\log^* n} |V_g||R_g|$$

To prove the claim, it will suffice to show that

$$|V_g||R_g| \leq n$$

for $g > 0$ (we may assume $n > 1$ and so $\log^* n > 0$)
For $g > 0$, we have

$$R_g = \{ F(g - 1) + 1, \ldots, F(g) \}$$

$$|V_g| \leq \sum_{r \in R_g} n/2^r \quad (\text{by Lemma 2})$$

$$= \frac{n}{2^{F(g-1)+1}} \sum_{j=0}^{F(g)-F(g-1)-1} 1/2^j$$

$$\leq \frac{n}{2^{F(g-1)}} = \frac{n}{F(g)}$$

Therefore,

$$|V_g||R_g| \leq \frac{n}{F(g)} \cdot |R_g| \leq \frac{n}{F(g)} \cdot F(g) = n.$$