Array Representation with **Quick Find** Invariants

Before connect(2, 3) operation:

{0, 1, 2, 4}, {3, 5}, {6}

After connect(2, 3) operation:

{0, 1, 2, 4, 3, 5}, {6}

**Quick Find Analysis**

If we have $V$ vertices...

- $E$ isConnected calls, each $O(1)$.
- $V$ connect calls, each $O(V)$.

**Simple graph:** $E < V^2$.

Kruskal’s: $O(E \log V + E + V^2) = O(E \log V + V^2)$

Both operations need to be $O(\log V)$!

**Improving the connect Operation**

Quick Union invariant. For each $v$, $\text{parent}[v]$ is the parent of $v$.

Show the result after calling connect(5, 0).

Worst-Case Height Trees

Spindly tree: repeatedly connect the first item’s tree below the second item’s tree.

- connect(4, 3)
- connect(3, 2)
- connect(2, 1)
- connect(1, 0)

Worst-case runtime for **both** connect and isConnected is $O(N)$.

```java
private int[] id;
boolean isConnected(int p, int q) {
    return id[p] == id[q];
}
void connect(int p, int q) {
    int setP = id[p];
    int setQ = id[q];
    for (int i=0; i<id.length; i++) {
        if (id[i] == setP)
            id[i] = setQ;
    }
}
```
Naive Quick Union Analysis

If we have $V$ vertices...

- $E$ isConnected calls, each $O(V)$.
- $V$ connect calls, each $O(V)$.

Kruskal's: $O(E \log V + EV + V^2) = O(E \log V + EV + V^2) = O(EV + V^2)$

Worst case is slower than Quick Find!

Weighted Quick Union by Height

Quick Union invariant. For each $v$, parent[$v$] is the parent of $v$.
The result of connect(5, 0) and connect(0, 5) should be the same!

Describe how to construct a worst-case height tree given a weighted quick union by height.

Private int find(int p) {
    while (p != parent[p])
        p = parent[p];
    return p;
}

boolean isConnected(int p, int q) {
    return find(p) == find(q);
}

void connect(int p, int q) {
    int i = find(p);
    int j = find(q);
    parent[i] = j;
}

Naive Quick Union Analysis

Hypothesis (from B-Trees). Unbalanced growth leads to worst-case height trees.

Identify (different due to parent pointers). When connecting, the second item's tree always becomes the new root.

Plan. Choose the new root based on a metric such as tree height.

Join by Web

1. Go to PollEv.com
2. Enter KEVINL
3. Respond to activity
**WQUByHeight: Worst-Case Height Tree**

- Size = 4, Height = 2
- Size = 8, Height = 3
- Size = 16, Height = 4

**WQUByHeight Analysis**

Keep track of heights with an extra array.

Worst-case height is \( \log(V) \!

- \( E \) isConnected calls, each \( O(\log V) \).
- \( V \) connect calls, each \( O(\log V) \).

Kruskal’s: \( O(E \log V + E \log V + V \log V) = O(E \log V + V \log V) = O(E \log V) \text{ if } E > V \)

**Weighted Quick Union with Path Compression**

Tie all visited nodes to the root.

Same asymptotic runtime.

Draw result of isConnected(14, 13).
**WQUBySize**: Worst-Case Height Tree

Worst-case analysis still works when we track subtree size, rather than subtree height!

```
void connect(int p, int q) {
    int i = find(p);
    int j = find(q);
    if (i == j) return;
    if (size[i] < size[j]) {
        parent[i] = j;
        size[j] += size[i];
    } else {
        parent[j] = i;
        size[i] += size[j];
    }
}
```

**WQUPathCompression**

WQUBySize with Path Compression.

Worst-case height is \( \log^* V \), where \( \log^* \) is the **iterated logarithm**—nearly constant.

- \( E \) isConnected calls, each \( O(\log^* V) \).
- \( V \) connect calls, each \( O(\log^* V) \).

\[ \log^*(2^{65536}) = 5. \]

Analysis is out of scope.

Kruskal’s: \( O(E \log V + E \log^* V + V \log V) \) = \( O(E \log V + V \log V) \) if \( E > V \)

**Summary**

Disjoint Sets ADT is used to track connected components in Kruskal’s algorithm.

- *Quick Find*: Array representation with no tree structure. Fast isConnected, slow connect.
- *Quick Union*: Array representation with tree structure. Worst-case linear-height trees.
- *Weighted Quick Union*: Choose the new root strategically based on a metric.

- **WQUByHeight**: Use subtree height as a metric. Results in \( \log V \) height.
- **WQUBySize**: Use subtree size as a metric. Results in \( \log V \) height.
- **WQUPathCompression**: Use subtree size as a metric. Results in \( \log^* V \) height—nearly constant.