Disjoint Sets

- Set of items - X .
- Maintain disjoint sets $S_1, \ldots, S_k$; i.e. $S_i \cap S_j = \emptyset \;\; \forall i \neq j$
- Operations:
  - MakeSet(x) - create a one-element set with x
  - Find-Set(x) - return the "name" of the set containing x
  - Union(x , y) - merge the set containing x and the set containing y into one set.

Representation

- Represent set as a rooted tree, with name being root
- Time per operation is proportional to height of tree.
- Two good heuristics
  - Union by Rank - make shallow tree a child of root of big tree
  - Path Compression - every time you touch a node, make it a child of root
- Union by Rank gives $\log V$ time per operation
- Union by Rank and path compression give better performance.

Disjoint Set Code

Make-Set(x)

```
p[x] ← x
rank[x] ← 0
```

Union(x, y)

```
LINK(FIND-SET(x), FIND-SET(y))
```

Link(x, y)

```
if rank[x] > rank[y]
    then p[y] ← x
    else  p[x] ← y
          if rank[x] = rank[y]
             then rank[y] ← rank[y] + 1
```

Find-Set(x)

```
if x ≠ p[x]
    then p[x] ← FIND-SET(p[x])
return p[x]
```

Ackerman's Function

$$A_k(j) = \begin{cases} j+1 & \textbf{if } k = 0 \\ A_{k-1}^{(j+1)}(j) & \textbf{if } k \geq 1 \end{cases}$$

$$\alpha(n) = \min\{k : A_k(1) \geq n\}$$

$$\begin{aligned}
A_0(j) &= j+1 \\
A_1(j) &= A_0^{(j+1)}(j) \\
&= 2j+1 \\
A_2(j) &= A_1^{(j+1)}(j) \\
&= 2(2(\cdots(2j+1)\cdots)+1)+1 \\
&= 2^{j+1}(j+1)-1
\end{aligned}$$

Ackerman

$$
\begin{aligned}
A_3(1) &= A_2^{(2)}(1) \\
&= A_2(A_2(1)) \\
&= A_2(7) \\
&= 2^8 \cdot 8 - 1 \\
&= 2^{11} - 1 \\
&= 2047
\end{aligned}
$$

$$
\begin{aligned}
A_4(1) &= A_3^{(2)}(1) \\
&= A_3(A_3(1)) \\
&= A_3(2047) \\
&= A_2^{(2048)}(2047) \\
&\gg A_2(2047) \\
&= 2^{2048} \cdot 2048 - 1 \\
&> 2^{2048} \\
&= (2^4)^{512} \\
&= 16^{512} \\
&\gg 10^{80} \ ,
\end{aligned}
$$

Summary

- **Amortized time per operation is** $\alpha(V)$ **.**
- **Can think of it as** $\lg^* V$ **, which is slightly bigger.**