Union-Find (disjoint sets)
- **Make-Set(x)**: Create new set \{x\}
- **Union(x, y)**: Replace set containing \(x\) (S) and set containing \(y\) (T) with SUT
- **Find(x)**: Return representative of set containing \(x\). One rep. per set

\[ \text{m operations total} \]
\[ n \text{ Make-Sets} \quad \# \text{elts} = n \]

**Lists**: Linked list for each set

Representative is head

Each element stored as triple:

\[ x | \text{head} | \text{next} \]
Make-Set(x):

\[ x \rightarrow \text{head} = x \]
\[ x \rightarrow \text{next} = \text{NULL} \]

Find(x):

\[ \text{return } x \rightarrow \text{head} \]

Union(x,y):

Time \( O(\log(n)) \)
S: \[\text{nice snack list}\]

T: \[\text{smaller list}\]

\[
\text{solve small list}\text{ after x}
\]

Time: \(O(\text{ITL}) = O(\min(5, ITL))\)

smaller set
**Thm:** Total time $\leq O(m + n \log n)$

(equivalent: amortized $O(\log n)$ for Make-Set, $O(1)$ for Union, Find)

**Pr:**

**Make-Set(x):**

\[ \text{Add} \ (-\log n \text{ } \text{to } \text{link to } x) \]

\[ \text{amortized cost} = \chi_{\text{Add}} \text{ } O(\log n) \]

\[ = (\log n + c) \]

**Find(x):** return x's head

\[ \text{amortized cost} = c + O(\log n) \]

**Union(x,y):** Take 1 $\$$ from each element in $\text{amortin}(x,y)$'s bank

\[ \text{amortized cost} = \text{free cost} + \Delta \$ \]

\[ < (T1 + c) - (T1 = O(1)) \]

Claim: Bank is always nonnegative

Proof: Every time lose $\$$ set doubles

\[ \exists \text{free} \leq \log n \text{ } \text{bank nonnegative} \]
Trees: $O(mn \log n)$ not great if $mn$ want to do better

"Lazy Unions":

Get rid of next pointers: trees:

Pros: Union fast (O(1))
Cons: Find slow (O(depth))
Two ideas:

1) Path Compression.
   When walking up tree in a Find, then update parent pointers.

2) Union by rank.
   Use "ranks" to decide which tree should be new root.

Thus, with union by rank and path compression,
the total cost is $O(m \log^* n) < (n^2)$

**Def:** $\log^* n = \#$ times need to take $\log_2$ until get $1$

**Ex:** $\log^* (2^{65536}) = 1 + \log^* (65536)$
   $= 1 + 1 + \log^* (16) = 1 + 1 + \log^* (4)$
   $= 4 + \log^* (2) = 5$
Make-Set\((x)\) : set \(x\rightarrow\text{rank} = 0, \ x\rightarrow\text{parent} = x\)

Find\((x)\) : follow parent pointers until reach root. For all nodes in path, set parent pointers to root.

\(\text{Row} \ (\text{time} : \Omega(\text{depth}))\)

Link\((v_1, v_2)\)

\(i f \ \text{rank}(v_1) > \text{rank}(v_2), \ \text{set}\)
\(\quad v_2\rightarrow\text{parent} = v_1\)

\(i f \ \text{rank}(v_2) > \text{rank}(v_1), \ \text{set}\)
\(\quad v_1\rightarrow\text{parent} = v_2\)

\(i f \ \text{rank}(v_2) = \text{rank}(v_1)\)
\(\quad v_2\rightarrow\text{parent} = v_1\)
\(\quad \text{rank}(v_1)++\)

\(\text{Union}(x, y) : \ \text{link}(\text{Find}(x), \text{Find}(y))\)
Find \( x \): 

\[
\begin{align*}
&\text{if } \text{rank}(z) = \text{rank}(w), \\
&\quad \text{\quad \quad \quad \quad \quad \quad \quad rank(z) \leftarrow \text{rank}(z) + 1
\end{align*}
\]

Union \( x, y \): 

\[
\begin{align*}
&\text{if } \text{rank}(z) = \text{rank}(w), \\
&\quad \text{\quad \quad \quad \quad \quad \quad \quad rank(z) \leftarrow \text{rank}(z) + 1
\end{align*}
\]
Properties of ranks:

1) If \( x \) not root, then \( \text{rank}(x) < \text{rank}(\text{parent}(x)) \)
   
2) When doing path compression, if parent of \( x \) changes, then new parent has rank \( \geq \) old parent

3) \( \text{rank}(x) \) can only change if \( x \) a root, and once \( x \) not a root, never becomes a root again

4) When \( x \) first reaches rank \( r \), must have \( \geq 2^r \) nodes in its tree

   \( \begin{align*}
   \text{i induction on } r \\
   r=0 \therefore \text{nothing to be done} \\
   \text{union of two trees of rank } r \\
   \Rightarrow 2^r \text{ elements in each} \\
   \geq 2^r \text{ elements in union}
   \end{align*} \)
Lemma: At most \( \frac{n}{2} \) nodes of rank \( \geq r \).

Rf: From property 4 plus (plug complication)
The total running time \( \leq O(m \log^* n) \)

- Make-Set: \( O(1) \) cost \( \Rightarrow O(m) \) total

- Union: \( \text{cost } O(1) + 2 \cdot \text{cost}(\text{Find}) \Rightarrow O(m) + 2 \cdot \text{cost}(\text{in Find}) \)

- Find: cost of a Find op

- Cost of all Finds \( \Rightarrow \) parent pointers inspected

- # of parent pointers to root \( \leq 2m \)

Case where: # parent pointers enqueued to non-roots

Let \( 2^{\uparrow i} \leq \exists \)

\( B(i) = \log n / \text{rank in} \)

\( (2^{\uparrow(i-1)}, 2^{\uparrow i}) \)
\[ \sum \log \pi_k \leq \log \pi_k \]

\[ \sum \pi_k \geq \pi_k \]

\[ \sum \pi_k \rho \]

\[ \rho \]