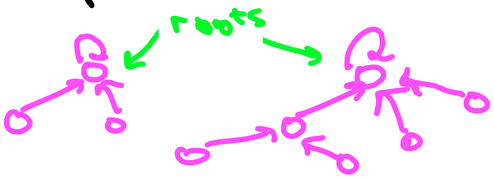
Algorithms: Design and Analysis, Part II

Advanced Union-Find

Union by Rank

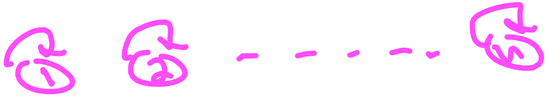
The Lazy Union Implementation



New implementation: each object $x \in X$ has a parent field.

Invariant: parent pointers induce a collection of directed trees on X.

(x is a root $\Longleftrightarrow$ parent[x] = x)



Initially: for all x, parent[x] := x.

FIND(x): traverse parent pointers from x until you hit the root.

UNION(x,y): s_1 = FIND(x); s_2 = FIND(y);
reset parent of one of s_1, s_2 to be the other

Quiz on Lazy Unions

Question: Suppose, in the UNION operation, we choose the new root arbitrarily from the two old ones. What is the worst-case running time of the FIND and UNION operations, respectively?

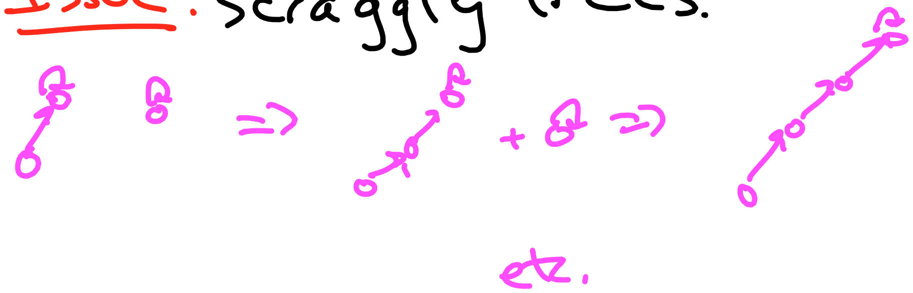
(A) $\Theta(1)$, $\Theta(1)$

(B) $\Theta(\log n)$, $\Theta(1)$

(C) $\Theta(\log n)$, $\Theta(\log n)$

[✓ (D) $\Theta(n)$, $\Theta(n)$]

Issue: scraggly trees.

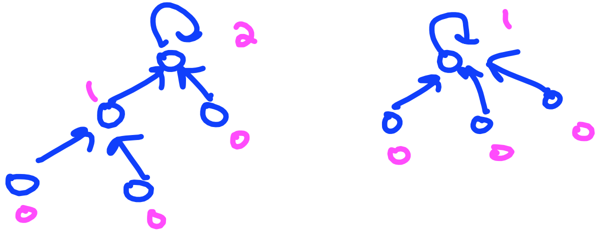


Union by Rank

in general, rank[x] = 1 + (max rank of x's children)

Ranks: For each $x \in X$, maintain field rank[x].

Invariant (for now): for all $x \in X$, rank[x] = maximum number of hops from some leaf to x.



[initially, rank[x] = 0 for all $x \in X$]

To avoid scraggly trees ("Union by Rank") given x & y:

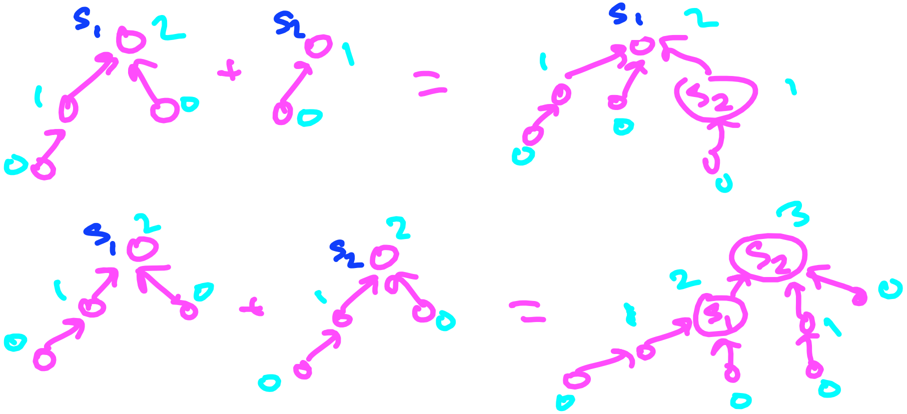
- $s_1 = \text{FIND}(x)$, $s_2 = \text{FIND}(y)$
- if $\text{rank}[s_1] > \text{rank}[s_2]$ then set $\text{parent}[s_2]$ to s_1 else set $\text{parent}[s_1]$ to s_2

To-do: update ranks to restore Invariant

Tim Roughgarden

Quiz on Rank Updates

Question: Recall $s_1 = \text{FIND}(x)$, $s_2 = \text{FIND}(y)$. How do the ranks of s_1 & s_2 change after UNION(x,y)?



(A) unchanged

(B) the one with larger rank goes up by 1

(C) the one with smaller rank goes up by 1

[✓ (D)] no change unless ranks of s_1, s_2 were equal, in which case s_2's rank goes up by 1

Tim Roughgarden