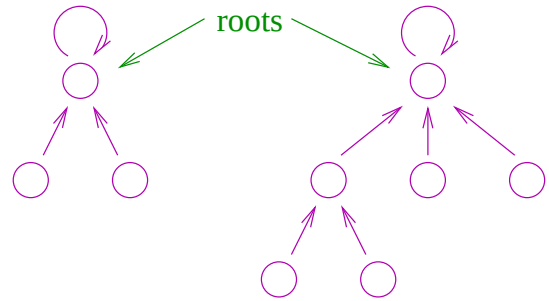
Advanced Union-Find

Algorithms: Design and Analysis, Part II

Union by Rank

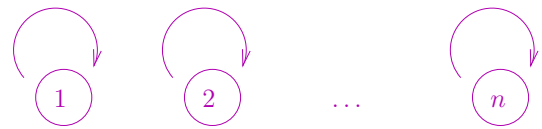
The Lazy Union Implementation

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 $\iff$ parent[x]=x)

Initially: For all x, parent[x]=x



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

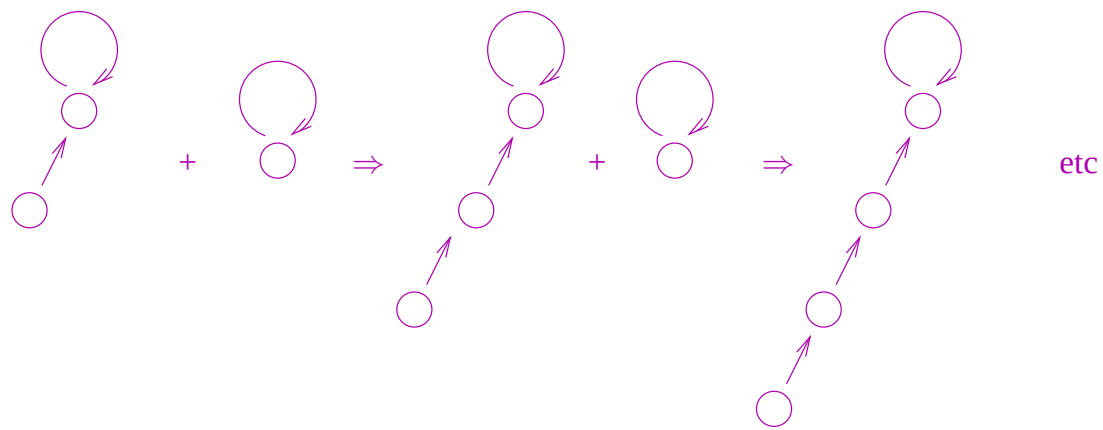
UNION(x, y): s_1 =FIND(x); s_2 =FIND(y); Reset parent of one of s_1, s_2 to be the other.

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) $\Theta(n), \Theta(n)$

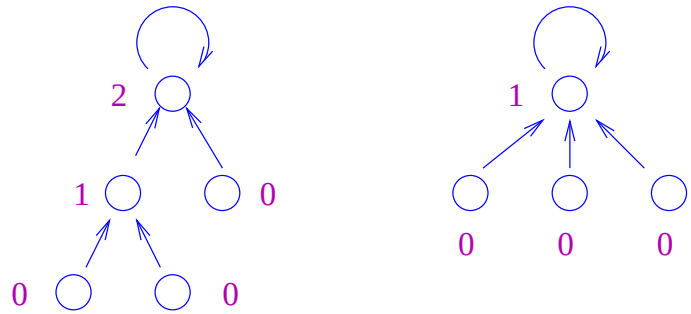
Issue: Scraggly trees:



Union by Rank

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

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



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 $x \in X$]

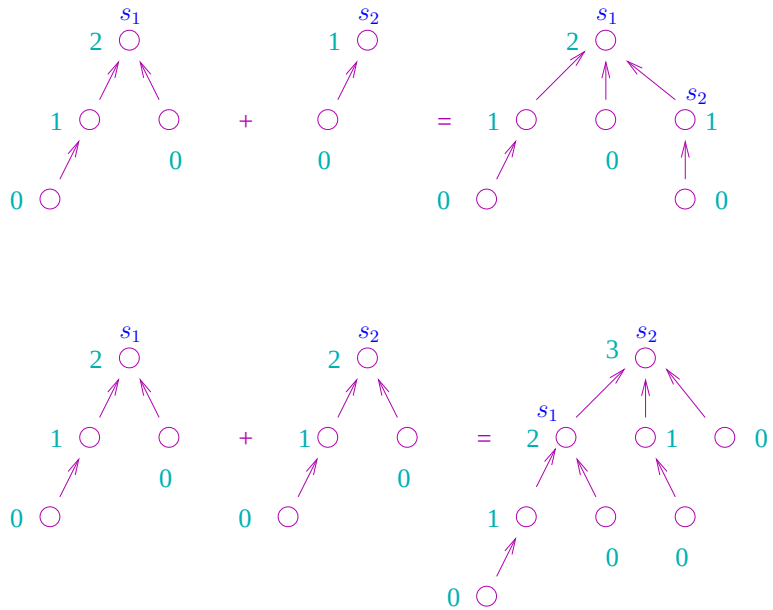
To avoid scraggly trees ("Union by Rank"): Given x & y:

- s_1=FIND(x), s_2=FIND(y)
- If rank[s_1]>rank[s_2] then set parent[s_2] to s_1 else set parent[s_1] to s_2.

To-do: Update ranks to restore Invariant.

Quiz on Rank Updates

Question: Recall s_1=FIND(x), s_2=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