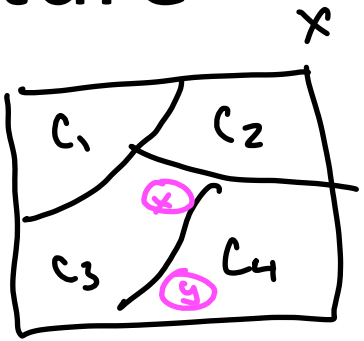
Advanced Union-Find

Lazy Unions

Algorithms: Design and Analysis, Part II

The Union-Find Data Structure



Raison d'être: maintain a partition of a set X.

FIND: given $x \in X$, return name of x's group.

UNION: given x & y, merge groups containing them.

Previous Solution (for Kruskal's MST algorithm)

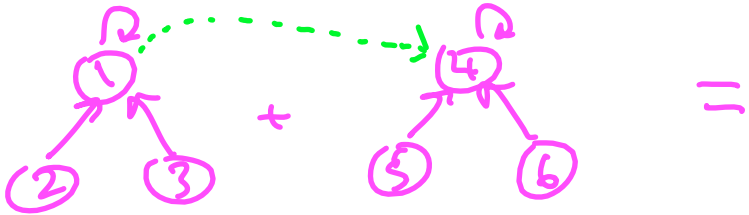
- each $x \in X$ points directly to the "leader" of its group
- $O(1)$ FIND [just return x's leader]
- $O(n \log n)$ total work for n UNIONs (when 2 groups merge, smaller group inherits leader of larger one)

Lazy Unions

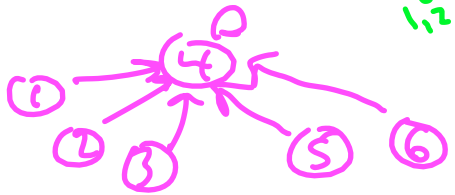
New idea: update only one pointer each merge!

How?:

old solution:

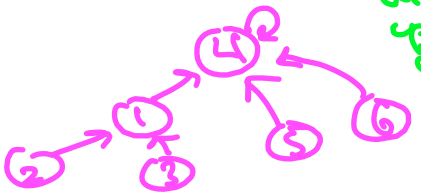


update objects 1, 2, 3

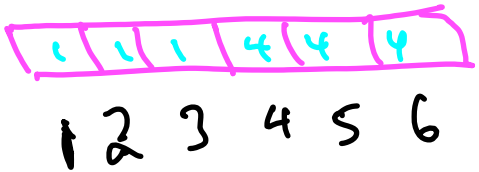


new solution:

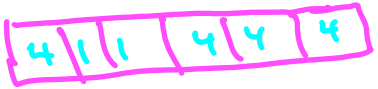
only update object 1



In array representation: (where A[i] → name of i's parent)



old solution



new solution

How to Merge?

In general: when two groups merge in a UNION, make one group's leader (i.e., root of the tree) a child of the other one.

Pro: UNION reduces to 2 FINDs [r_1 = FIND(x), r_2 = FIND(y)] and O(1) extra work [link r_1, r_2 together].

Con: To recover leader of an object, need to follow a path of parent pointers [not just one!]

⇒ not clear if FIND still takes O(1) time