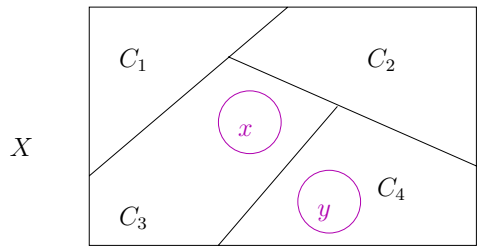
Advanced Union-Find

Algorithms: Design and Analysis, Part II

Lazy Unions

The Union-Find Data Structure

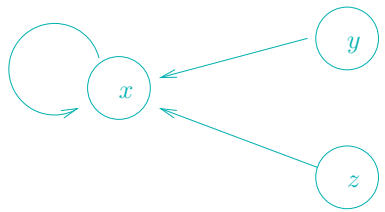
Raison d'être: Maintain a partition of a set X.



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.



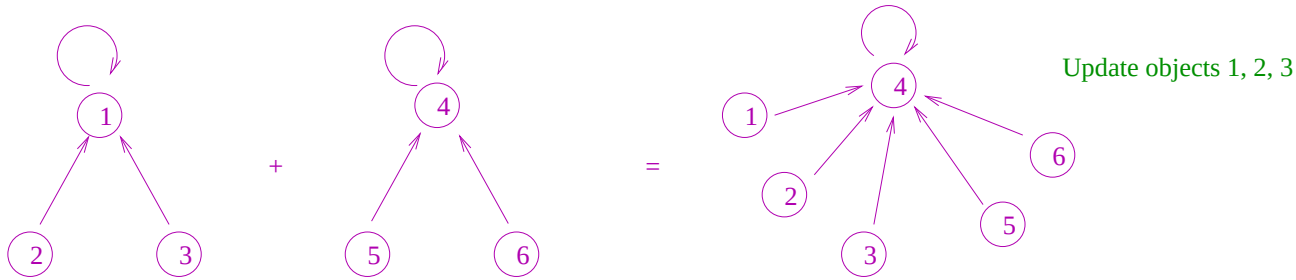
- $O(1)$ FIND [just return x's leader]
- $O(n \log n)$ total work for n UNIONS [when 2 groups merge, smaller group inherits leader of larger one]

Lazy Unions

New idea: Update only one pointer each merge!

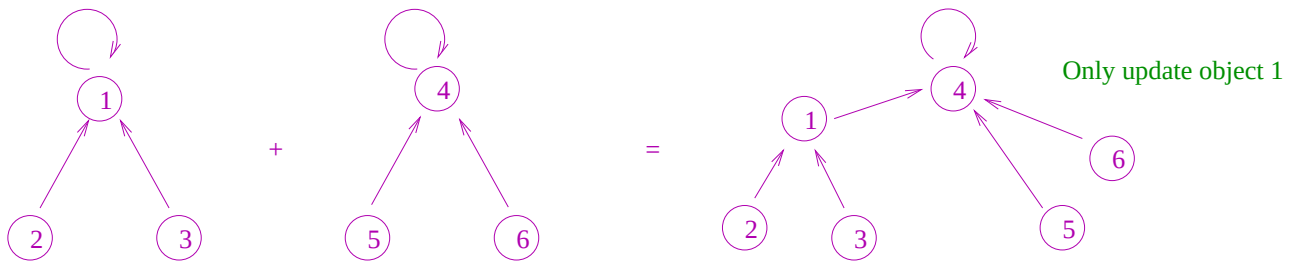
How?

Old solution:



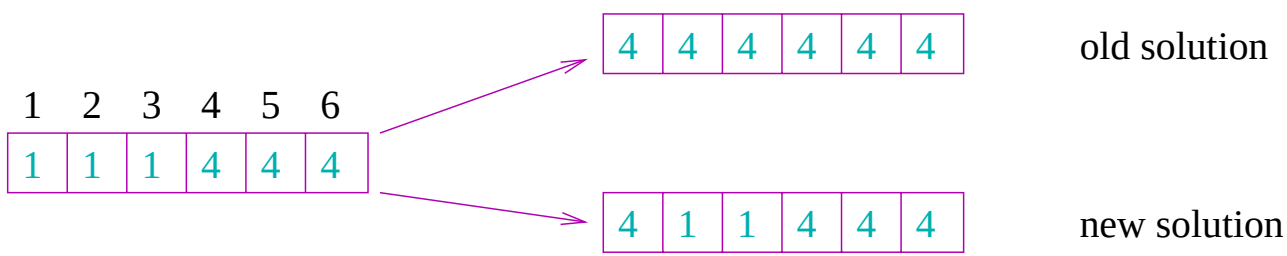
Update objects 1, 2, 3

New solution:



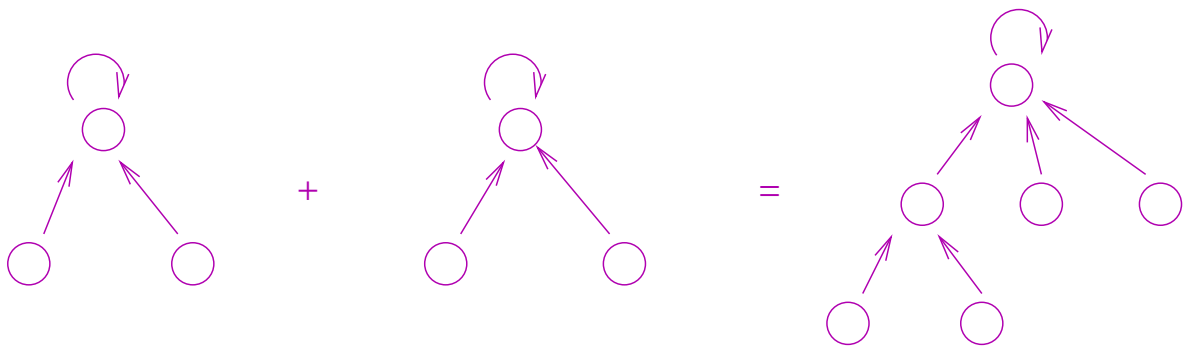
Only update object 1

In array representation: (Where $A[i] \leftrightarrow$ name of i's parent)



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!]
$\Rightarrow$ Not clear if FIND still takes $O(1)$ time.