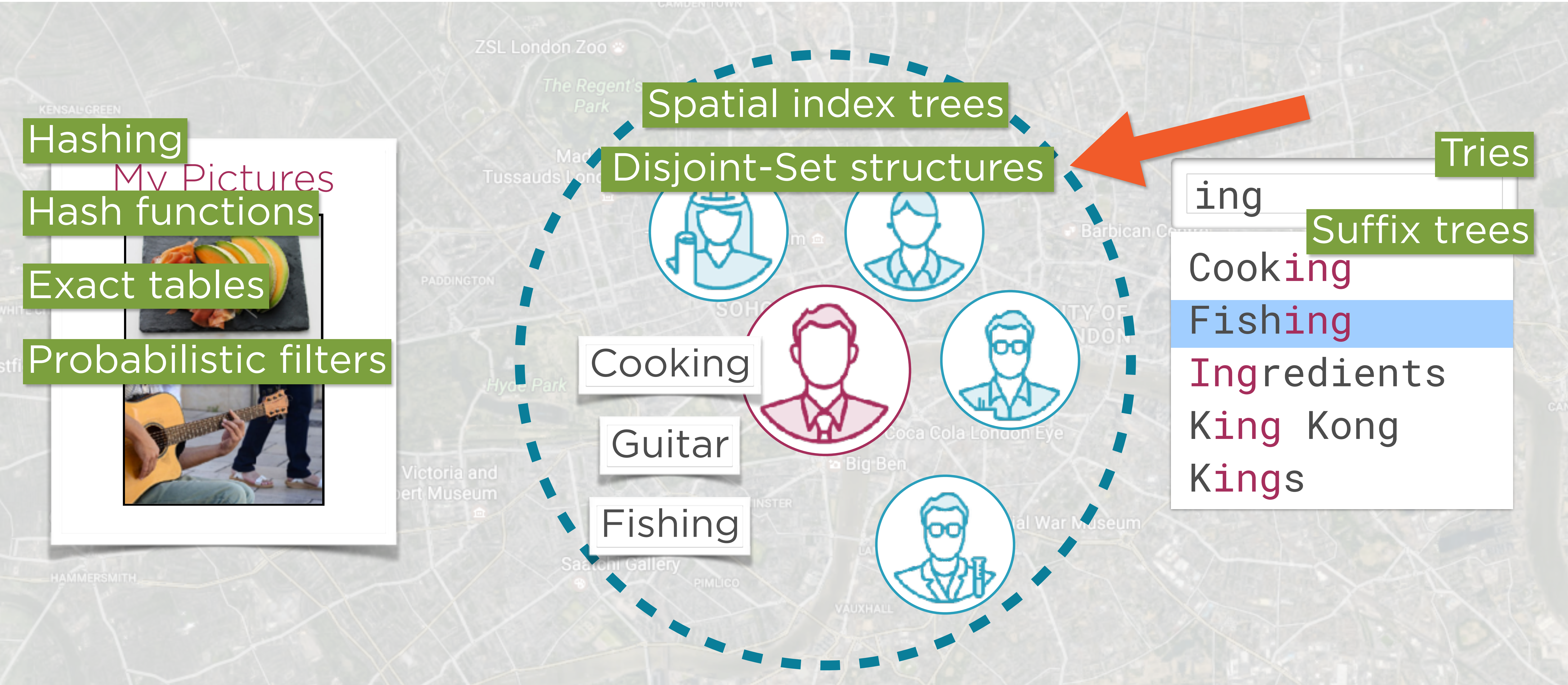
Efficiently Set Operations with Disjoint-Set Structures

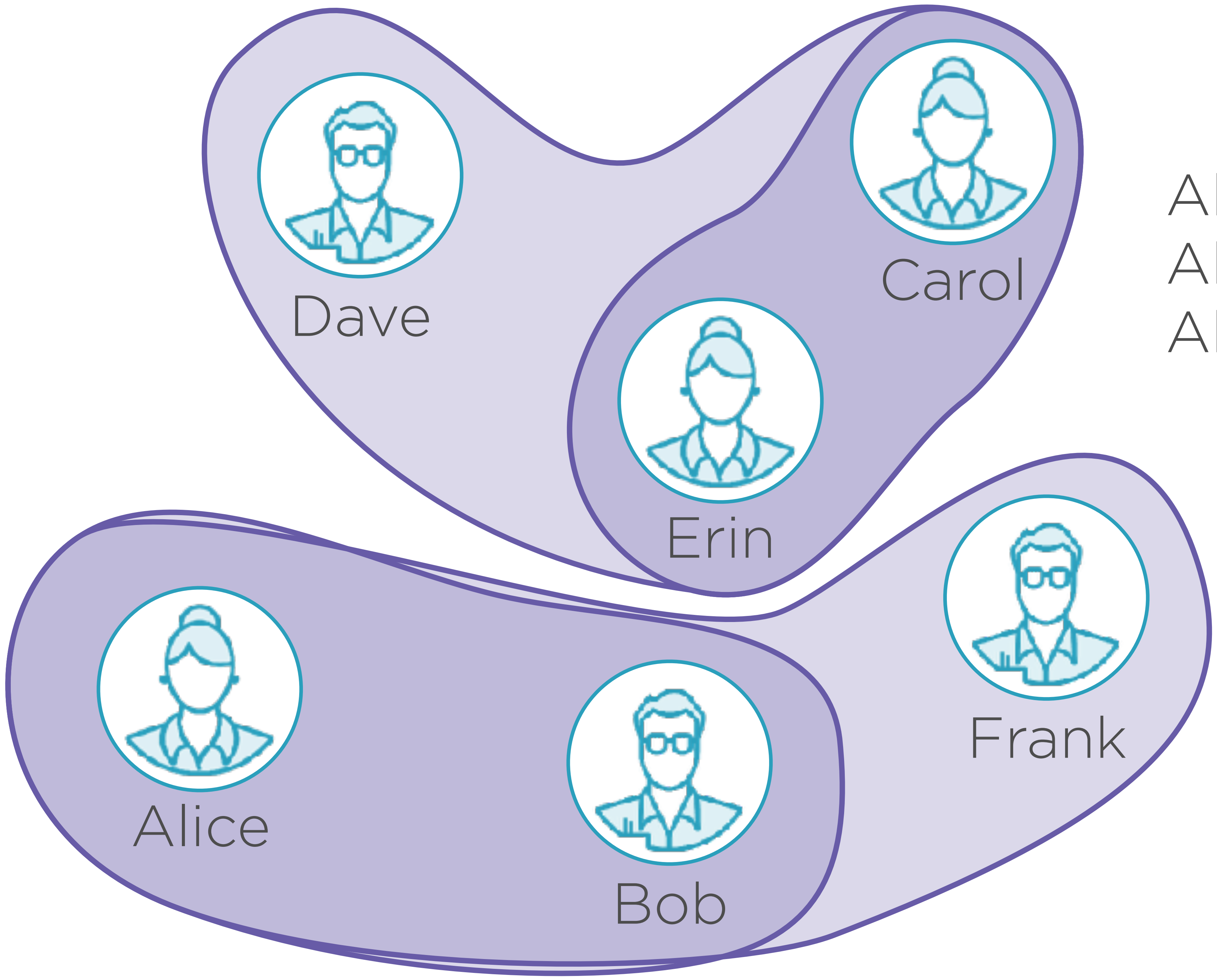
Rasmus Resen Amossen
SOLUTION ARCHITECT

rasmus.resen.org

The Match Finder App



Connecting Users

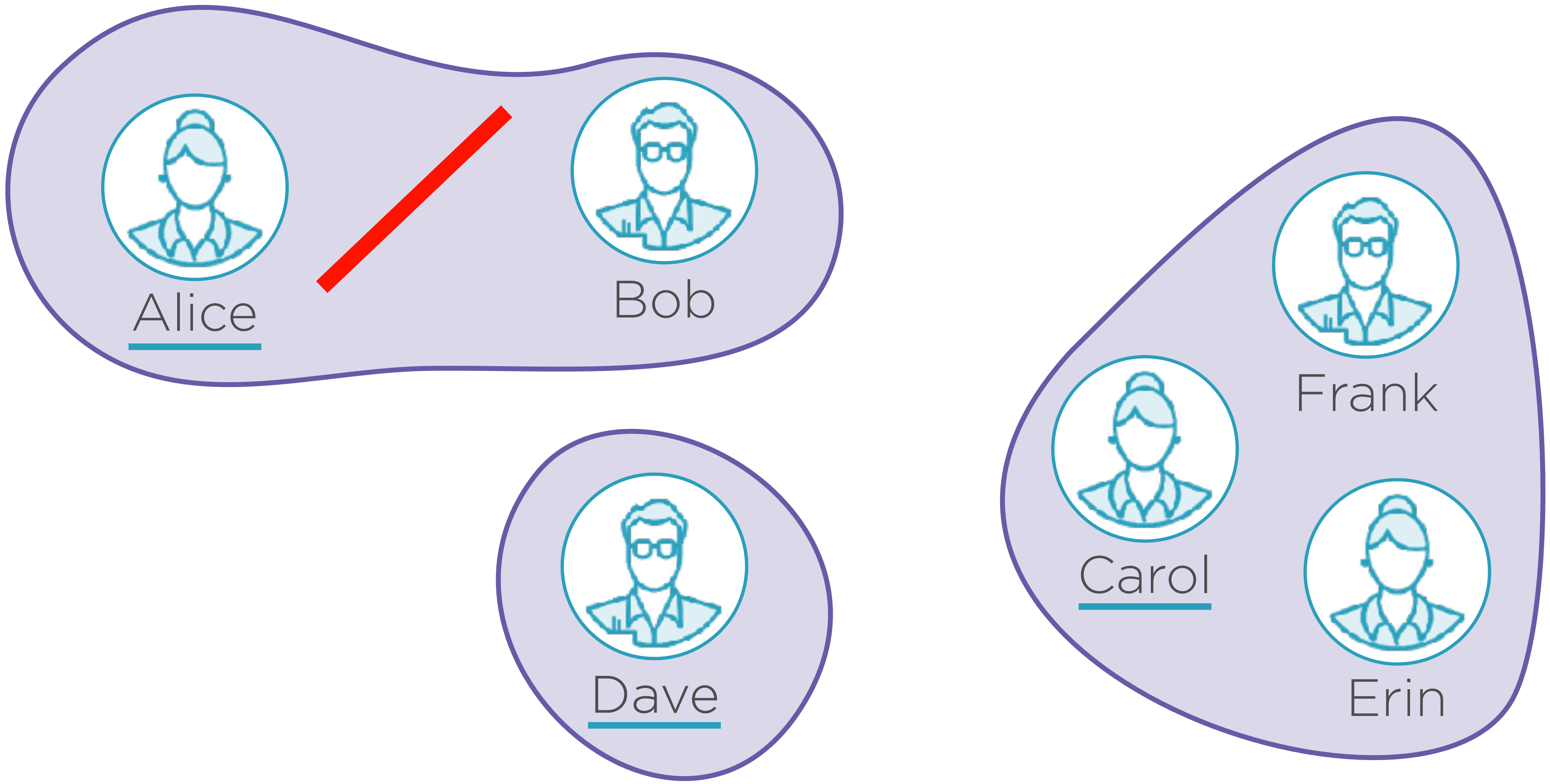


Alice texts Bob
Alice adds Bob to network
Alice watches Bob's profile

Demo

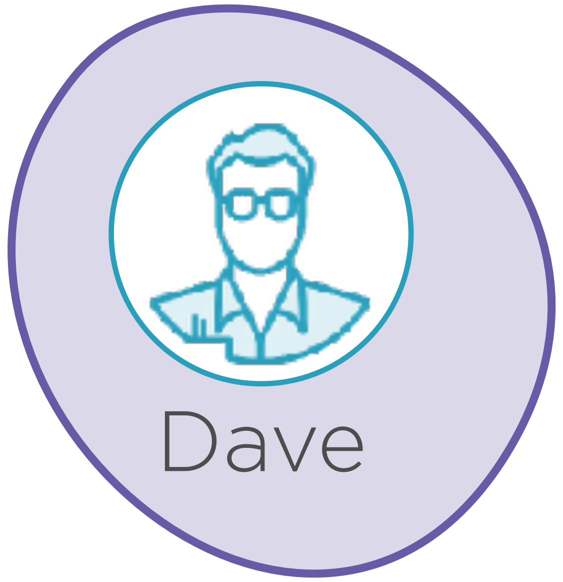
Beginning an Interaction Monitor

Disjoint Sets

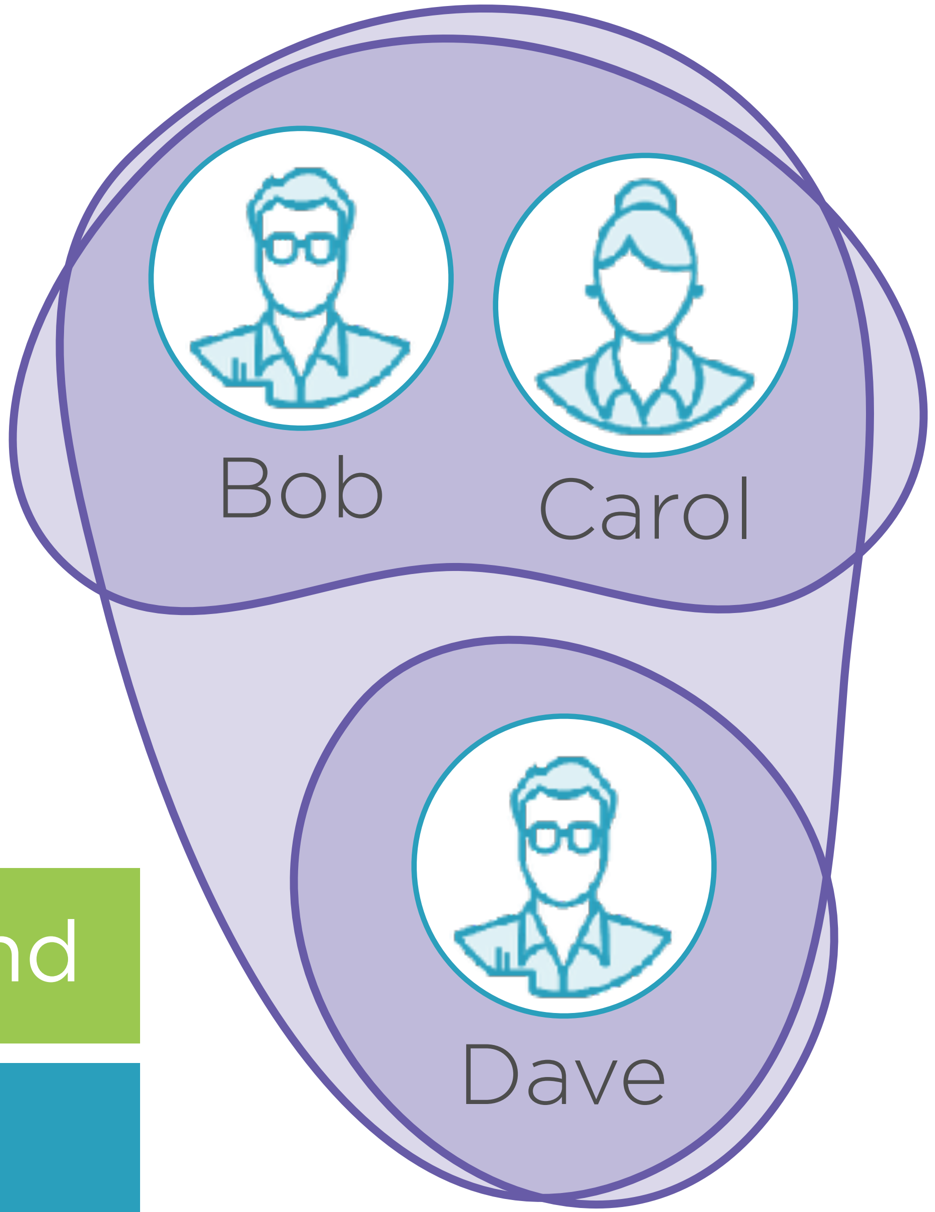


Operations

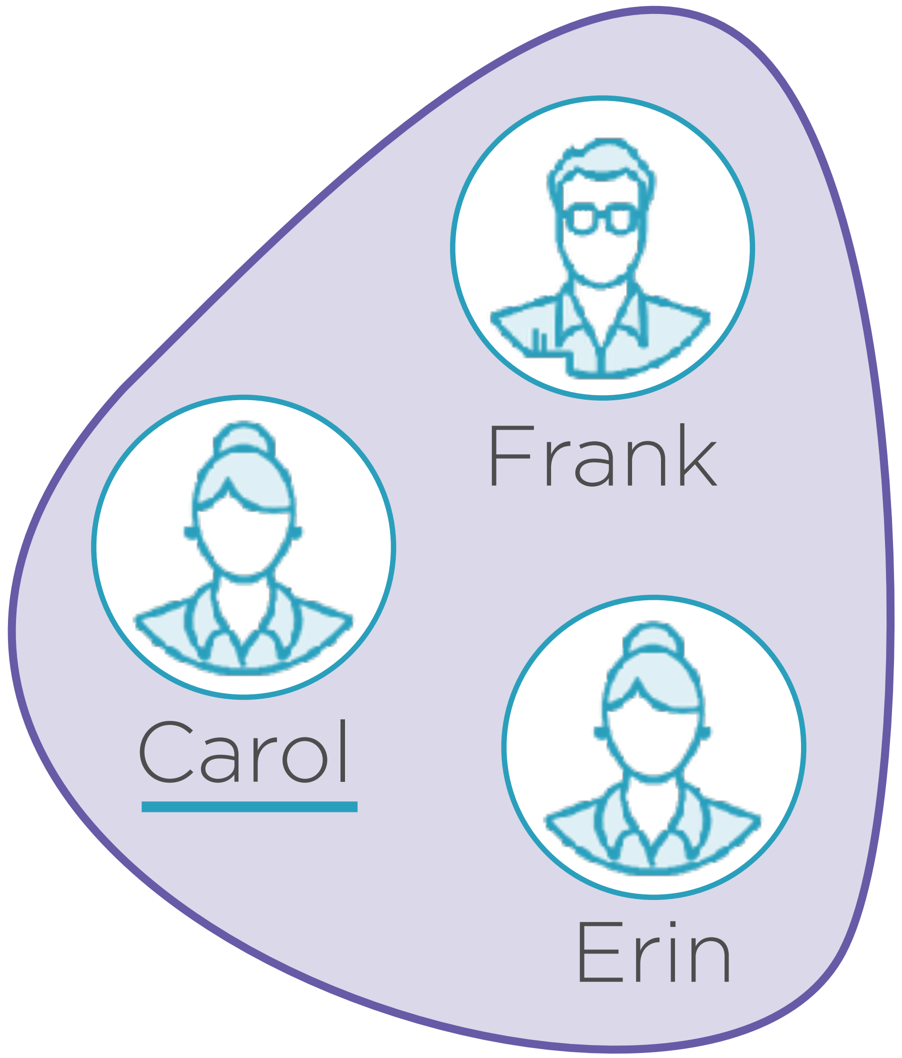
MakeSet(x)



Union(x, y)



FindSet(x)

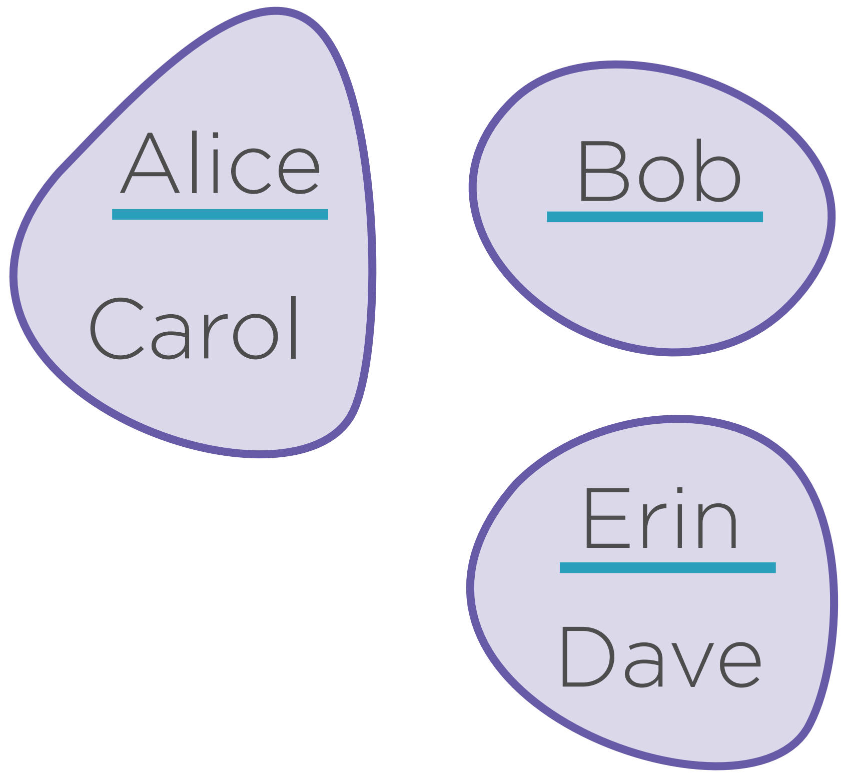


Disjoint-set	Union-Find
Maps	Forest

Basic Maps Structure

Set ID to Item List

Alice	Bob	Erin
$\downarrow$	$\downarrow$	$\downarrow$
Alice Carol	Bob	Erin Dave



Item ID to Set ID

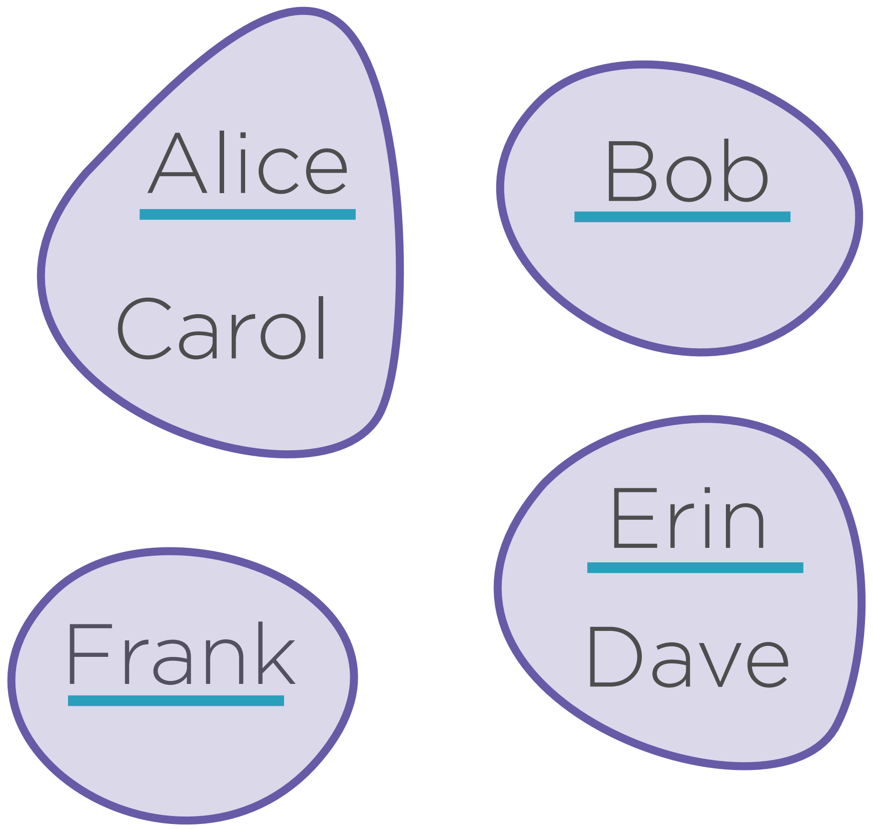
Alice	Bob	Carol	Dave	Erin
$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$
Alice	Bob	Alice	Erin	Erin

MakeSet(*x*) Using Maps

Complexity: $O(1)$

Set ID to item list

Alice	Bob	Erin	Frank
↓	↓	↓	↓
Alice Carol	Bob	Erin Dave	Frank



MakeSet(Frank)

Item ID to set ID

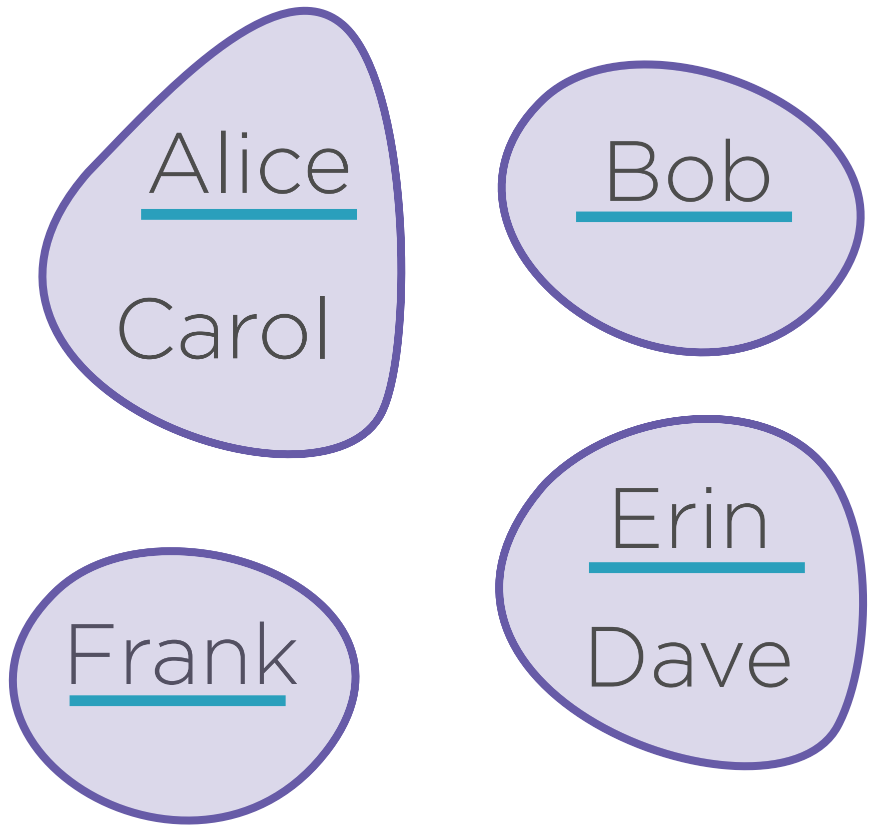
Alice	Bob	Carol	Dave	Erin	Frank
↓	↓	↓	↓	↓	↓
Alice	Bob	Alice	Erin	Erin	Frank

FindSet(x) Using Maps

Complexity: $O(1)$

Set ID to item list

Alice	Bob	Erin	Frank
↓	↓	↓	↓
Alice Carol	Bob	Erin Dave	Frank



FindSet(Carol) = Alice

Item ID to set ID

Alice	Bob	Carol	Dave	Erin	Frank
↓	↓	↓	↓	↓	↓
Alice	Bob	Alice	Erin	Erin	Frank

Union(*x*, *y*) Using Maps

Complexity: *O*(*N*) for *N* items

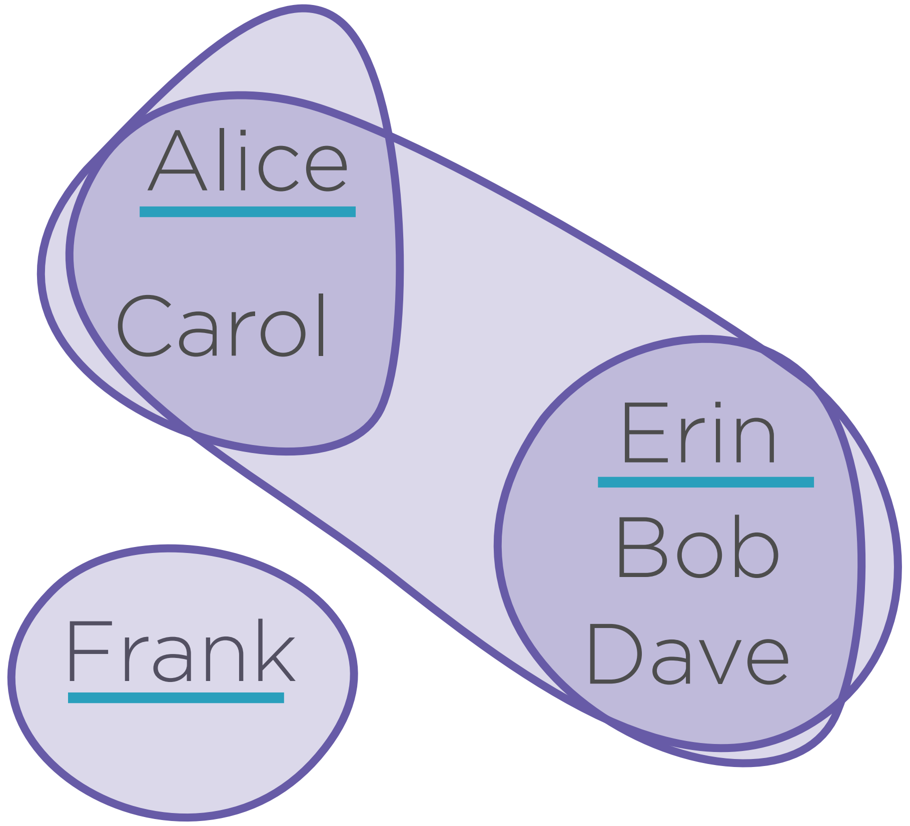
Set ID to item list

Alice	Erin	Frank
Alice Carol	Erin Dave Bob	Frank

Union(Carol, Erin)

Append smaller to larger

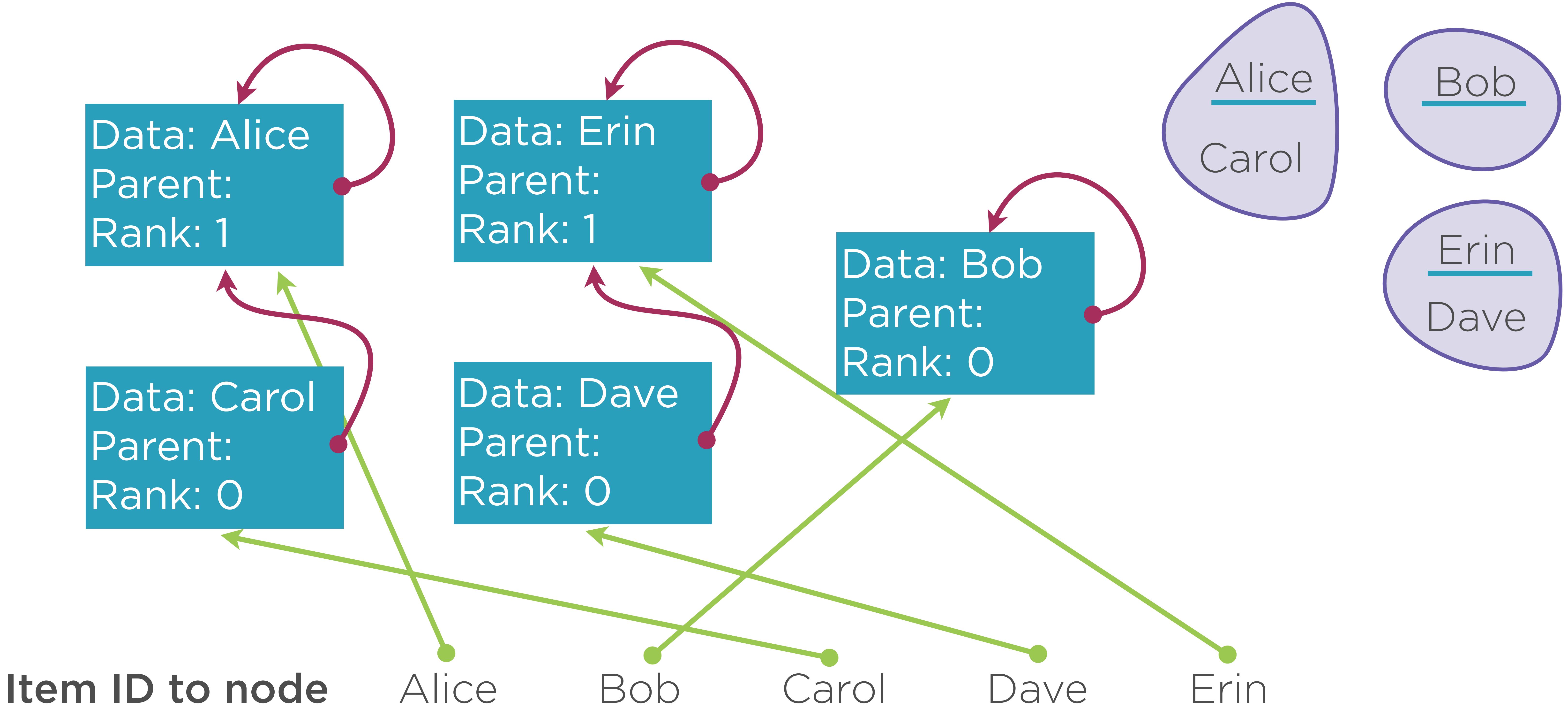
Change set ID



Item ID to set ID

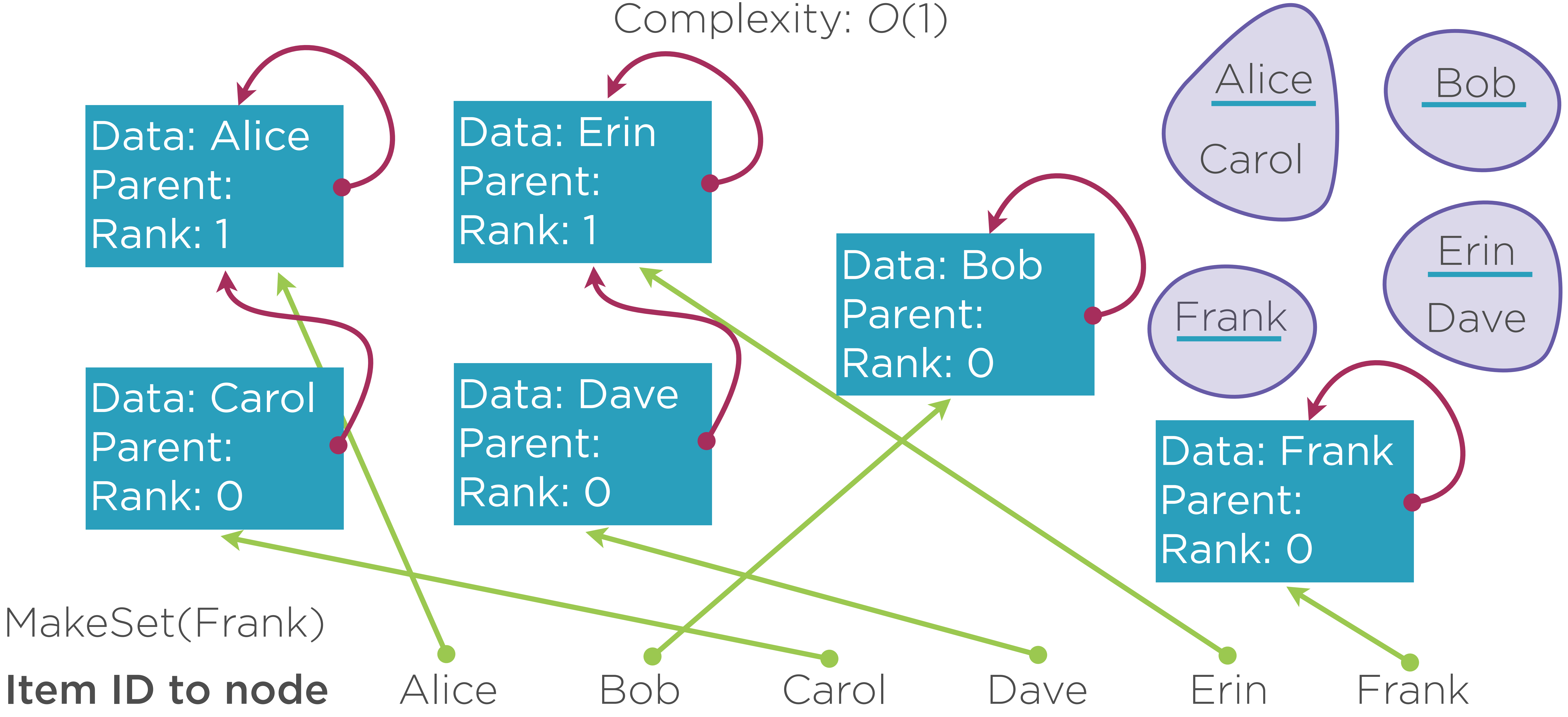
Alice	Bob	Carol	Dave	Erin	Frank
Alice Erin	Bob	Alice Erin	Erin	Erin	Frank

Basic Forest Structure



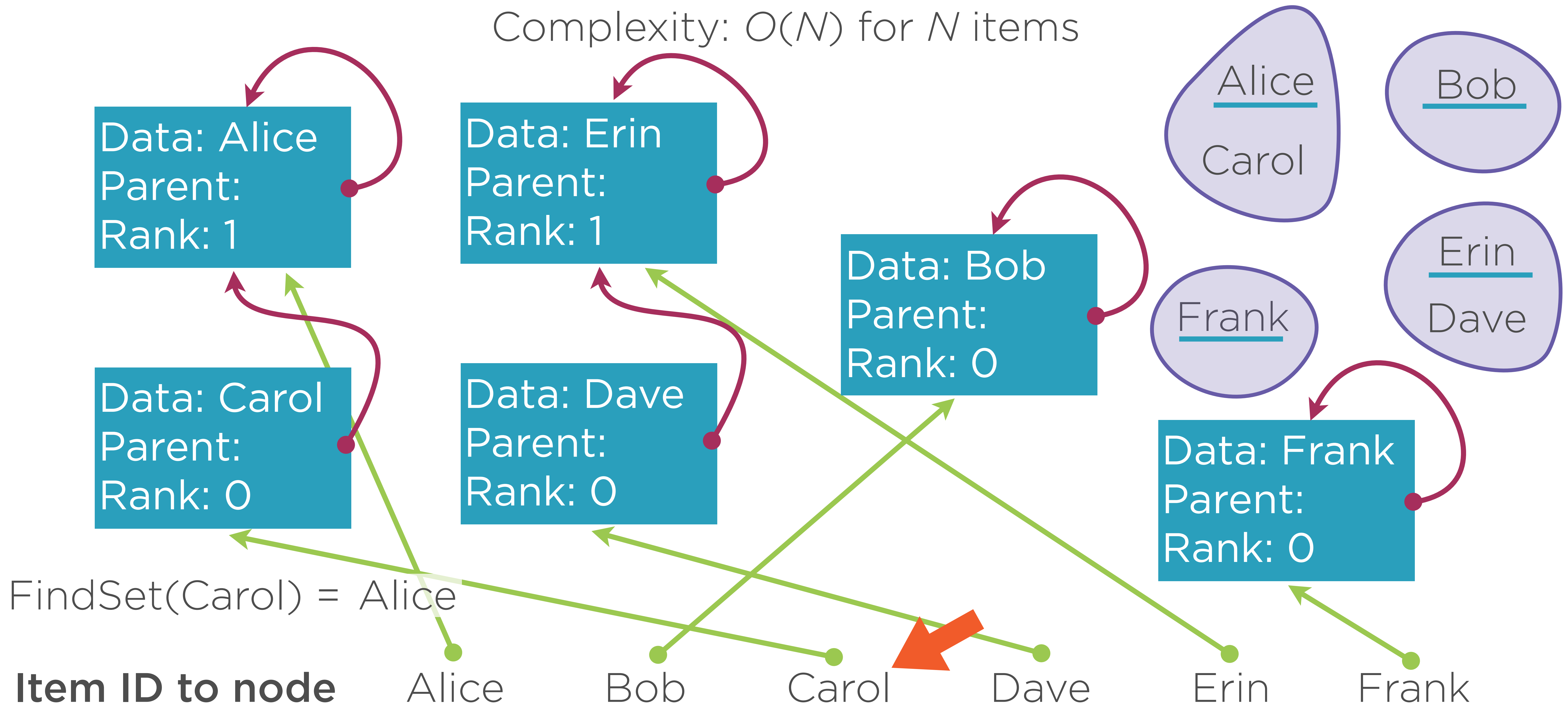
MakeSet(x) Using Forest

Complexity: $O(1)$



FindSet(*x*) Using Forest

Complexity: *O*(*N*) for *N* items



With path compression

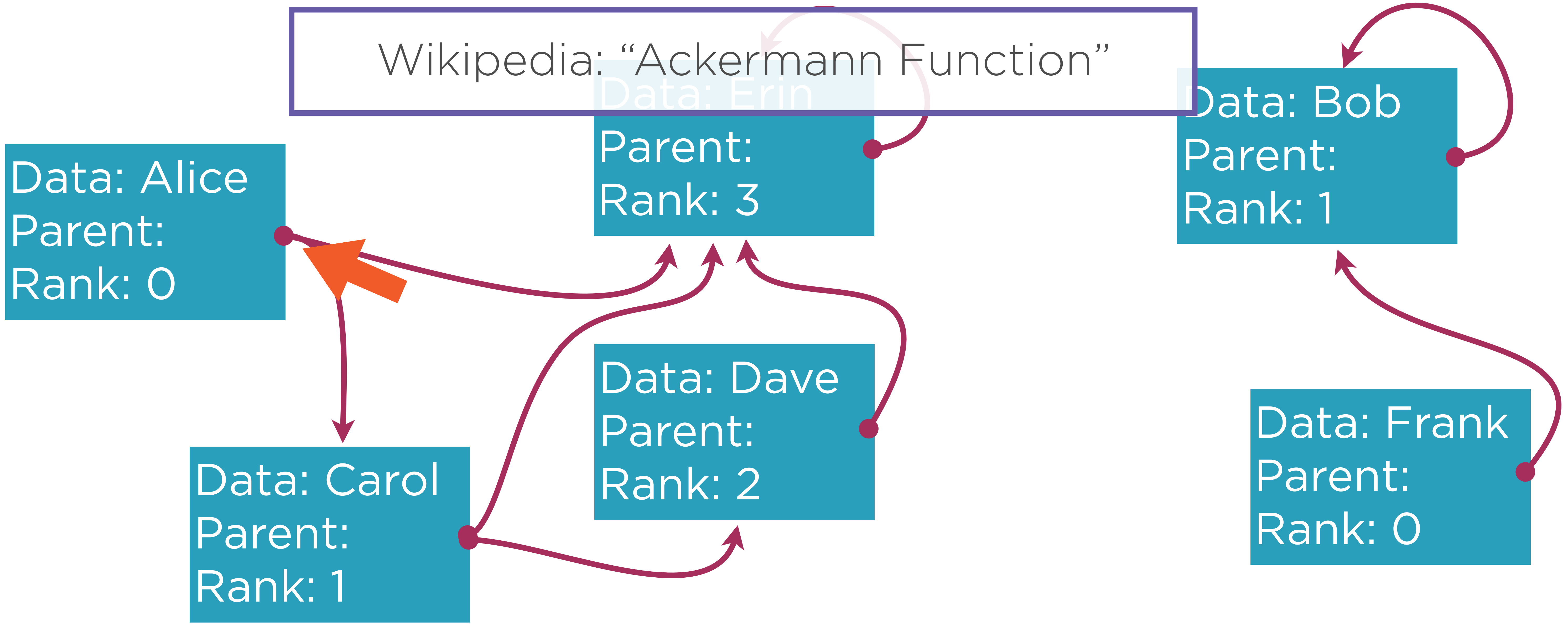
FindSet(x) Using Forest

Complexity: $O(\alpha(N))$ for N items

Complexity: $O(1)$

$\alpha(N)$: Inverse Ackermann function <5 in practice

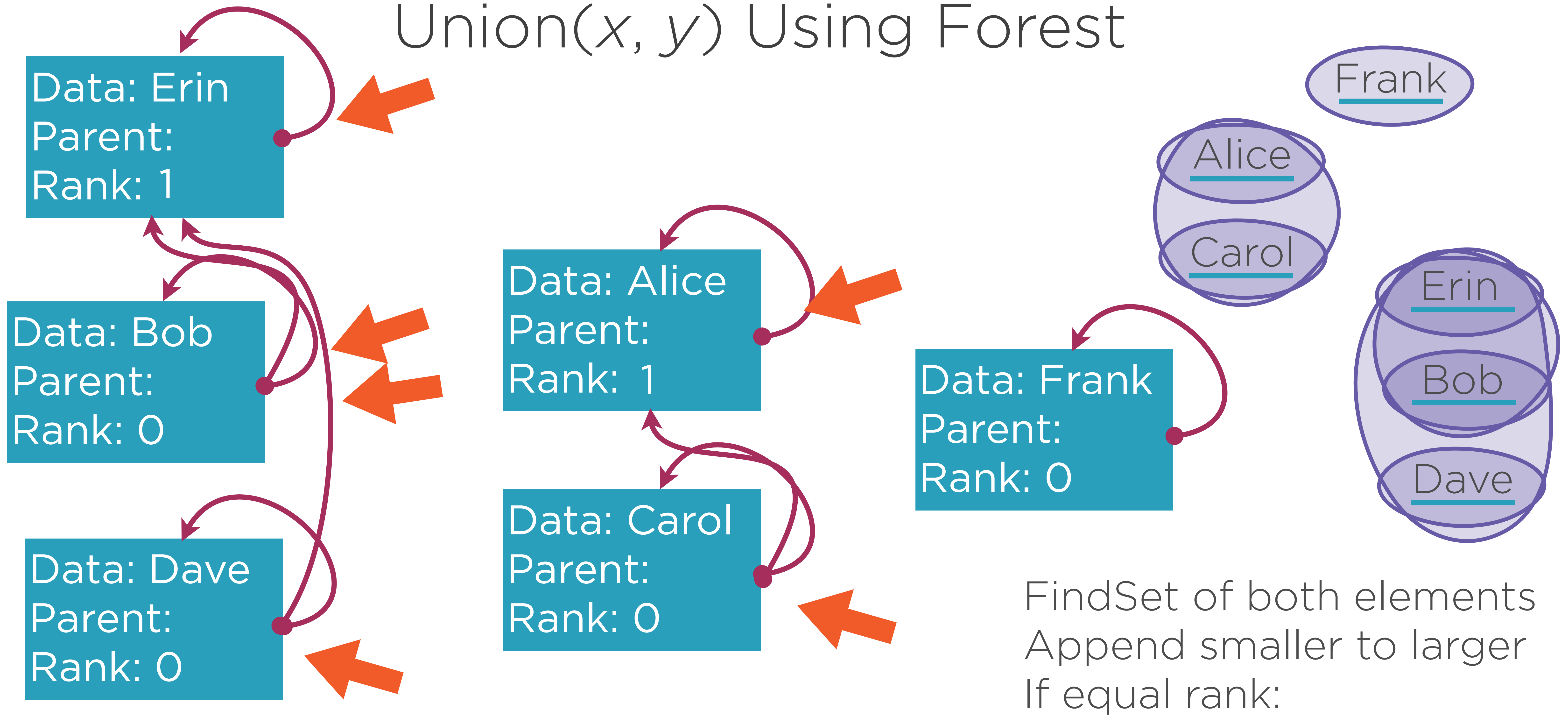
Wikipedia: “Ackermann Function”



FindSet(Alice) = Erin

Recursively set Parent = FindSet(Parent)

Union(*x*, *y*) Using Forest

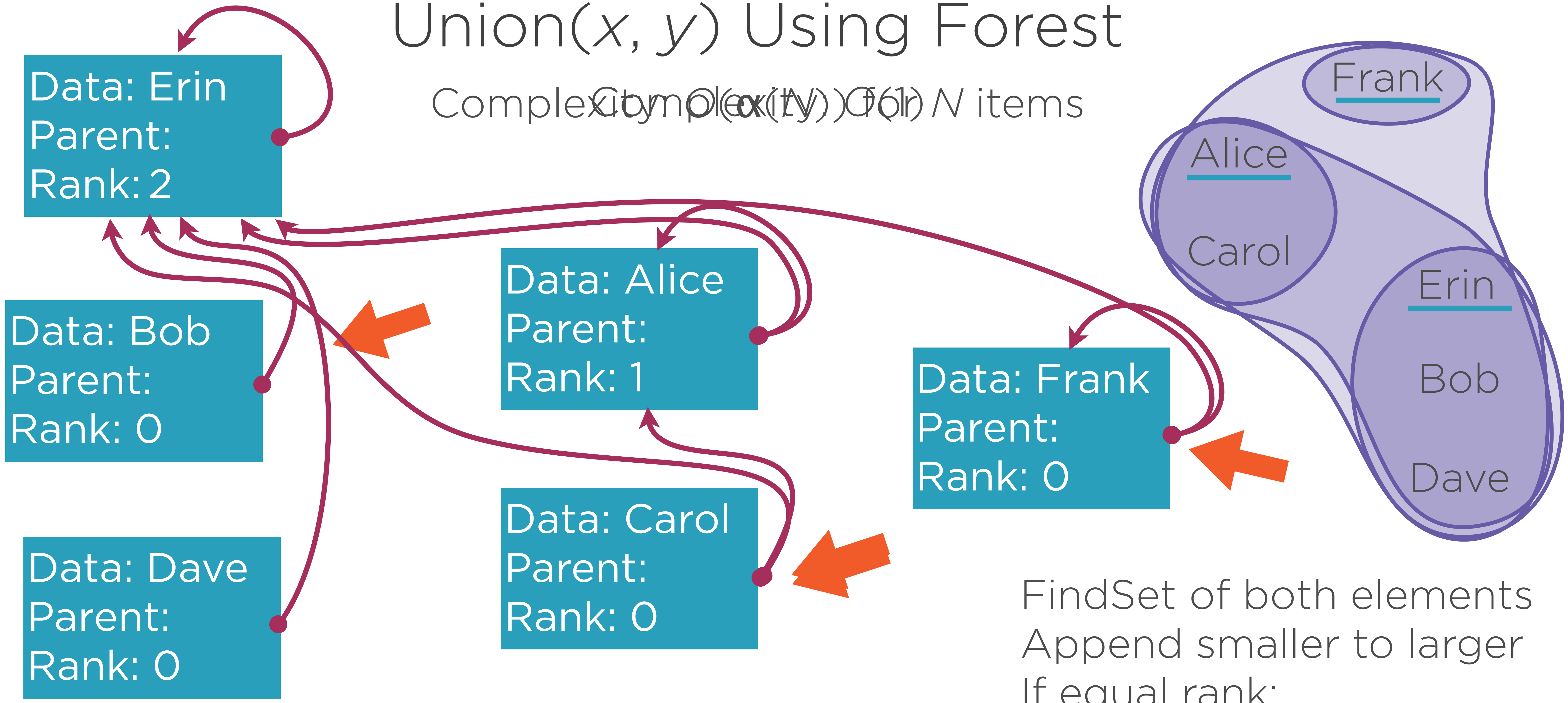


Union(Bob,Erin)
Union(Dave,Erin)
Union(Alice,Carol)

FindSet of both elements
Append smaller to larger
If equal rank:
- Chose one arbitrarily
- Increase its rank

Union(*x*, *y*) Using Forest

Complexity: O(α(*N*)) O(1) for *N* items

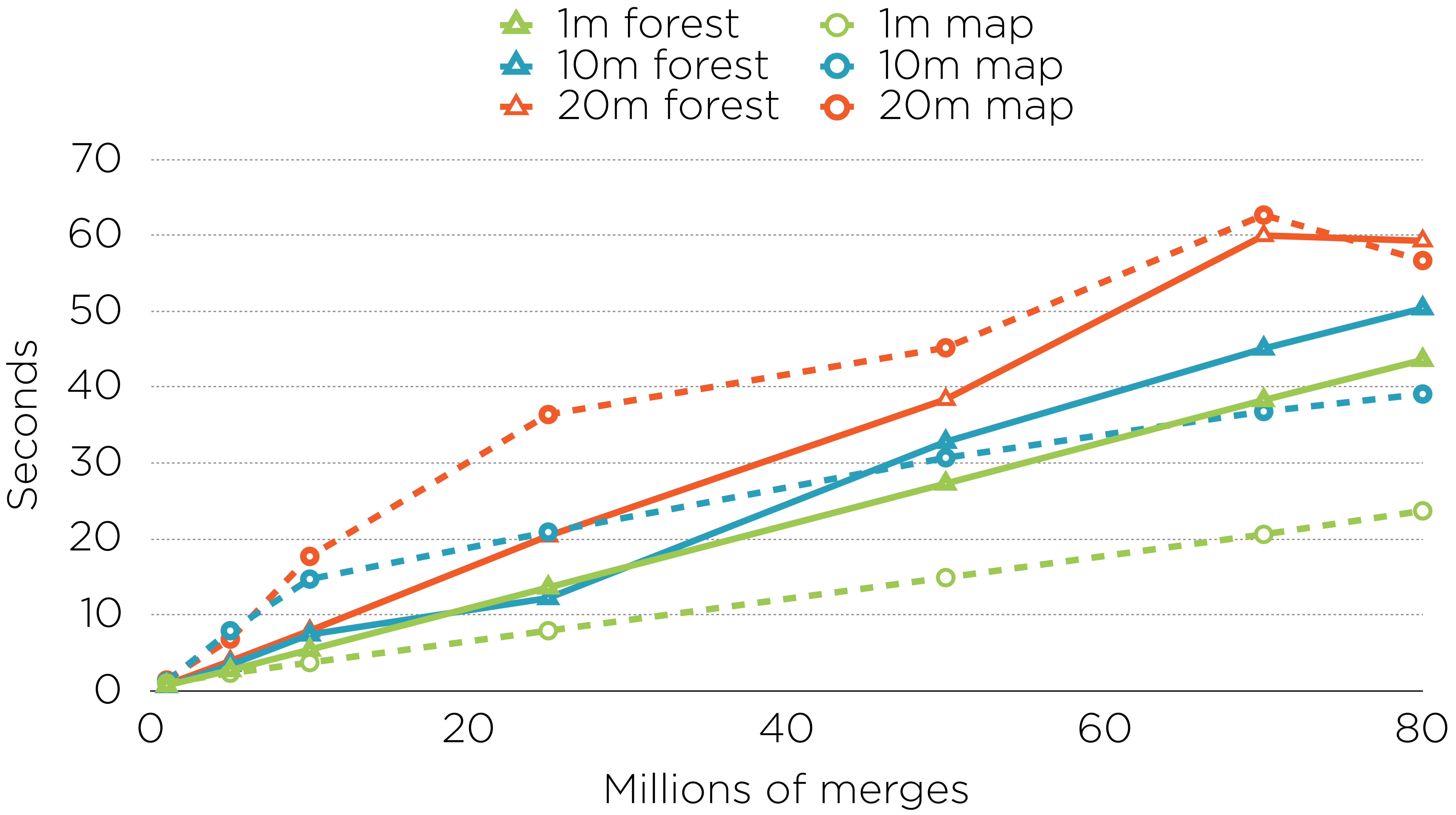


Union(Frank, Carol)

FindSet of both elements
Append smaller to larger
If equal rank:
  Chose one arbitrarily
  Increase its rank

Demo

Implementing the Interaction Monitor


1m forest
10m forest
20m forest
1m map
10m map
20m map
Seconds
70
60
50
40
30
20
10
0
0
20
40
60
80
Millions of merges

Lessons Learned

Efficient for finding connected components

Representative element identifies set

Disjoint-set supports three operations

MakeSet(*x*) | Union(*x*, *y*) | FindSet(*x*)

Using maps
Constant FindSet(*x*)
Linear Union(*x*, *y*)

Using forest
Linear FindSet(*x*)
Linear Union(*x*, *y*)

Rank + path compression
Constant FindSet(*x*)
Constant Union(*x*, *y*)