Australian National University

COMP3600/6466 Algorithms

Lecture 21

S2 2016

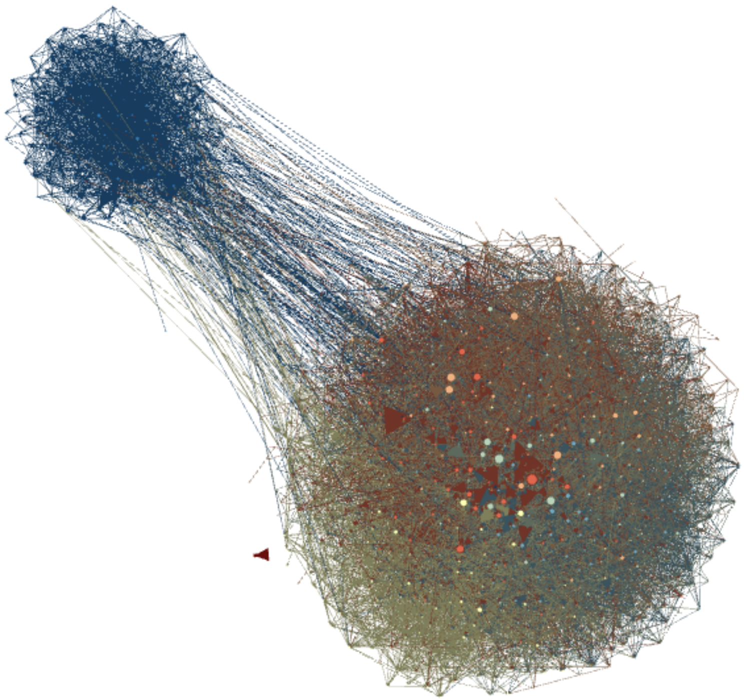
Dr. Hassan Hijazi

Prof. Weifa Liang

Twitter Graph

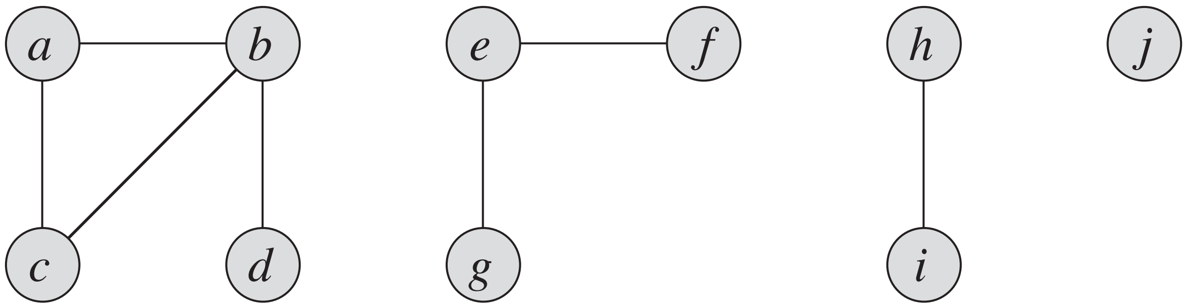
Detecting Clusters

- Liberals
- Conservatives



Related Problem

Connected Components of a Graph

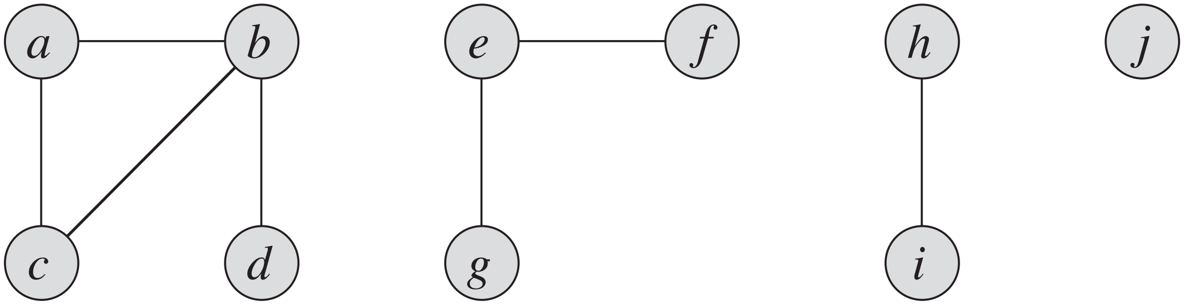


INPUT: $G(V, E)$ where V is a set of nodes

E a set of edges connecting a pair of nodes in V

OUTPUT: Disjoint sets of connected components

Dynamic Data Loading



If V has n nodes, what is the maximal number of edges we can have in E?

n(n-1)/2

Let $|V| = 100,000$

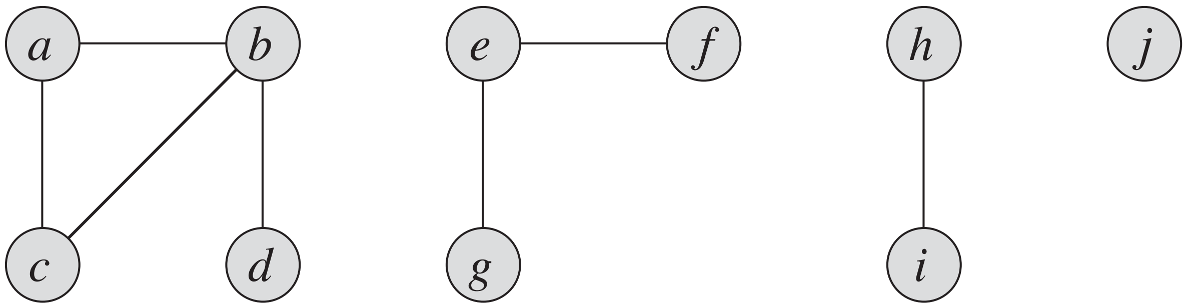
Give an upper bound on $|E|$

~ 5 billions

DO NOT MAKE A LOCAL COPY!

Edges are Loaded Dynamically

Connected Components of a Graph



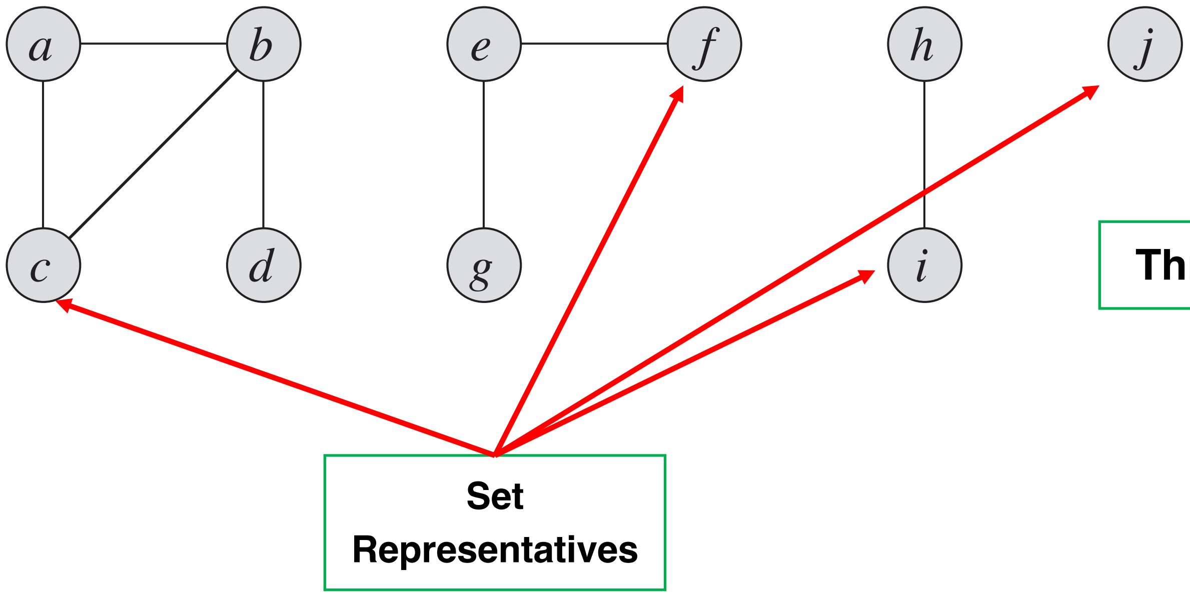
$$V = \{a,\ b,\ c,\ d,\ e,\ f,\ g,\ h,\ i,\ j\}$$

OUTPUT?

The Australian National University

Disjoint-Set Data Structure

What is the ideal data structure?



Three supported operations:

$\text{MAKE-SET}(x)$

$\text{UNION}(x, y)$

$\text{FIND-SET}(x)$

Disjoint-Set Data Structure

$$\text{MAKE-SET}(x)$$

Create a set whose only member is object x

x cannot be in any other set (Disjoint sets)

Australian National University

Disjoint-Set Data Structure

$$\textsc{Union}(x, y)$$

Let S_x denotes the set containing x, S_y the set containing y

Combine S_x and S_y, into a new set $S_x \cup S_y$

$S_x \cap S_y = \emptyset$ (Disjoint sets)

Disjoint-Set Data Structure

$$\textsc{Find-Set}(x)$$

Identify the set containing x

Returns a pointer to the representative of the (unique) set containing x

Disjoint-Set Data Structure

Connected Components of a Graph

$$V = \{a,\ b,\ c,\ d,\ e,\ f,\ g,\ h,\ i,\ j\}$$

On a server

$$E = \{(b,d),\ (e,g),\ (a,c),\ (h,i),\ (a,b),\ (e,f),\ (b,c)\}$$

```
CONNECTED-COMPONENTS(G)
for each vertex v ∈ G.V
        MAKE-SET(v)
for each edge (u, v) ∈ G.E
        if FIND-SET(u) ≠ FIND-SET(v)
                UNION(u, v)
```

Server request

```
CONNECTED-COMPONENTS(G)
for each vertex v ∈ G.V
    MAKE-SET(v)
for each edge (u, v) ∈ G.E
    if FIND-SET(u) ≠ FIND-SET(v)
        UNION(u, v)
```

$$V = \{a,\ b,\ c,\ d,\ e,\ f,\ g,\ h,\ i,\ j\}$$

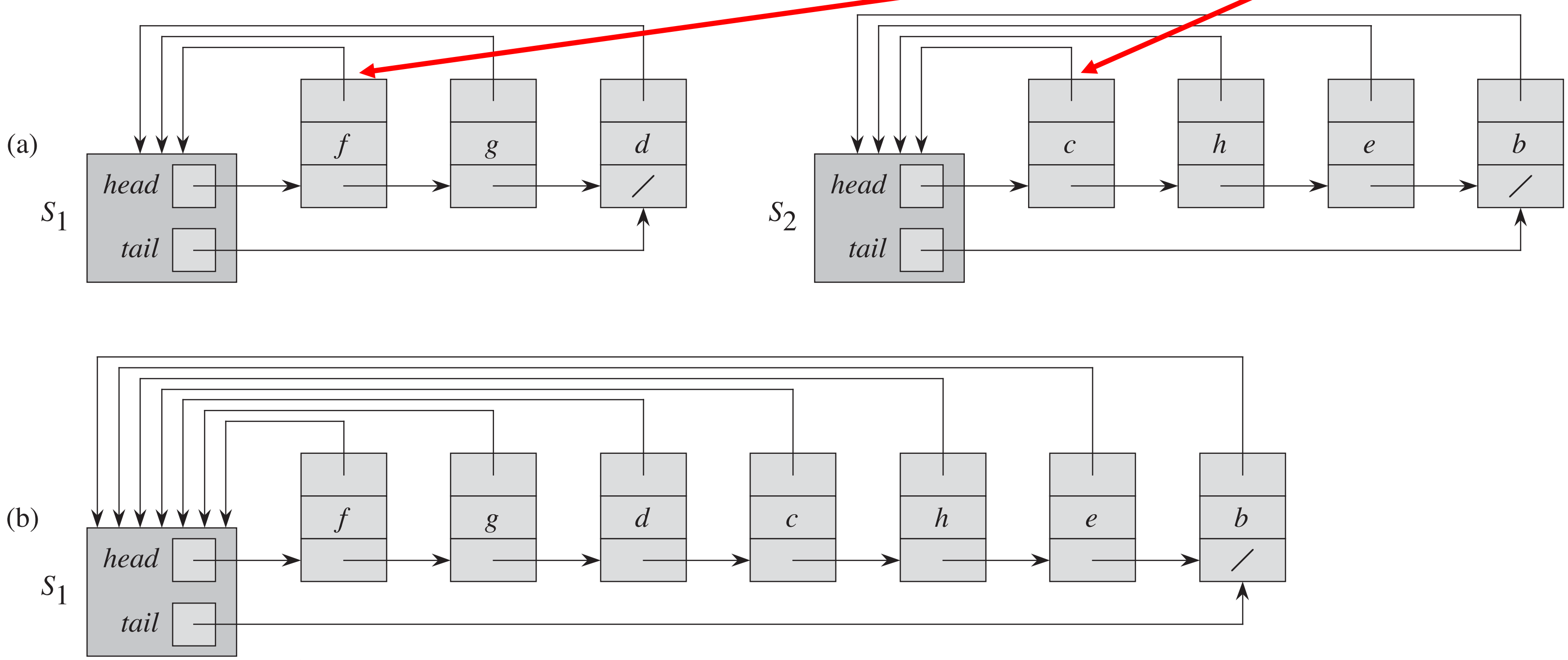
$$E = \{(b,d),\ (e,g),\ (a,c),\ (h,i),\ (a,b),\ (e,f),\ (b,c)\}$$

Edge processed	Collection of disjoint sets									
initial sets	$\{a\}$	$\{b\}$	$\{c\}$	$\{d\}$	$\{e\}$	$\{f\}$	$\{g\}$	$\{h\}$	$\{i\}$	$\{j\}$

Disjoint-Set Data Structure

Linked List Implementation

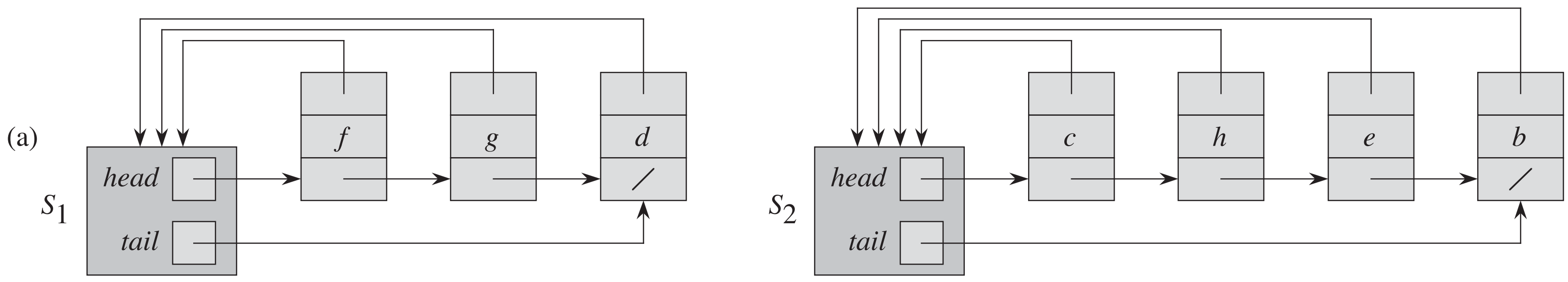
Set Rep



$\textsc{Union}(g, e)$

Disjoint-Set Data Structure

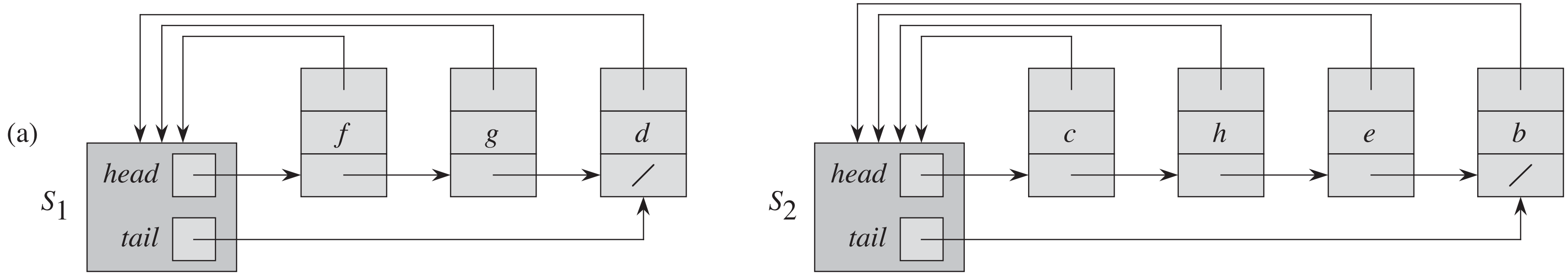
Naïve Union



$UNION(x, y)$ always appends y's list onto the end of x's list

Disjoint-Set Data Structure

Linked List Implementation



MAKE-SET(x)	$O(1)$
FIND-SET(x)	$O(1)$
UNION(x, y)	?

Disjoint-Set Data Structure

Union Performance

Let n be the number of initial disjoint sets

Claim. The worst case running time for $n-1$ calls to Union is $\Theta(n^2)$.

Operation	Number of objects updated
MAKE-SET(x_1)	1
MAKE-SET(x_2)	1
$\vdots$	$\vdots$
MAKE-SET(x_n)	1
UNION(x_2, x_1)	1
UNION(x_3, x_2)	2
UNION(x_4, x_3)	3
$\vdots$	$\vdots$
UNION(x_n, x_{n-1})	$n-1$

Updating back-pointers

$$\sum_{i=1}^{n-1} i = \Theta(n^2)$$

Disjoint-Set Data Structure

Union Weighted-Sum Heuristic

Add a length attribute in the list object

Always append the smaller list between S_y and S_x

Better asymptotical running time?

Claim. The worst case running time for $n-1$ calls to Union is $O(n \lg n)$.

Disjoint-Set Data Structure

Union Weighted-Sum Heuristic

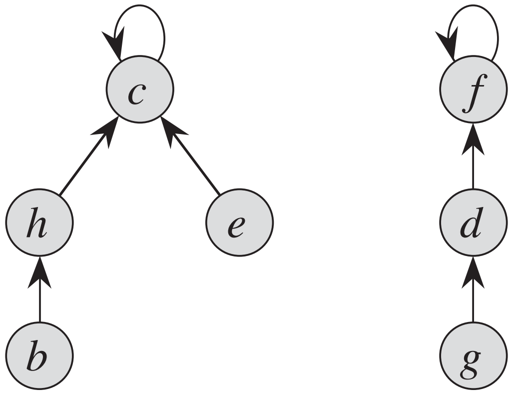
Claim. The worst case running time for $n - 1$ calls to Union is $O(n \lg n)$.

Proof. Each time x's pointer is updated, x must be in the smaller set. The first time x's pointer was updated, the resulting set must have had at least 2 members. Similarly, the next time x's pointer was updated, the resulting set must have had at least 4 members. Continuing on, we observe that for any $k \leq n$, after x's pointer has been updated $\lg k$ times, the resulting set must have at least k members. Since the largest set has at most n members, each object's pointer is updated at most $\lg n$ times.

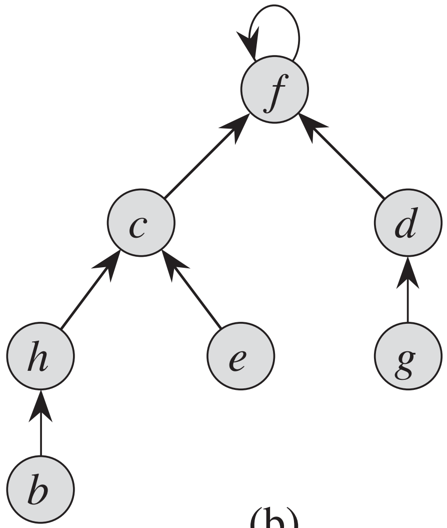
Disjoint-Set Data Structure

A Better Implementation?

Disjoint-Set Forests



(a)

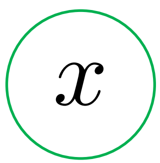


(b)

Disjoint-Set Data Structure

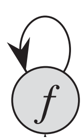
A Better Implementation?

Disjoint-Set Forests



$x.p$, x's parent in the tree.
$x.rank$, an upper bound on x's subtree height.

```
MAKE-SET(x)
x.p = x
x.rank = 0
```




```
FIND-SET(x)
while x ≠ x.p
    x ← x.p
return x
```

Disjoint-Set Data Structure

A Better Implementation?

Union by Rank

$\textsc{Union}(x, y)$

1 $\textsc{Link}(\textsc{Find-Set}(x), \textsc{Find-Set}(y))$

roots of the tree containing the original elements

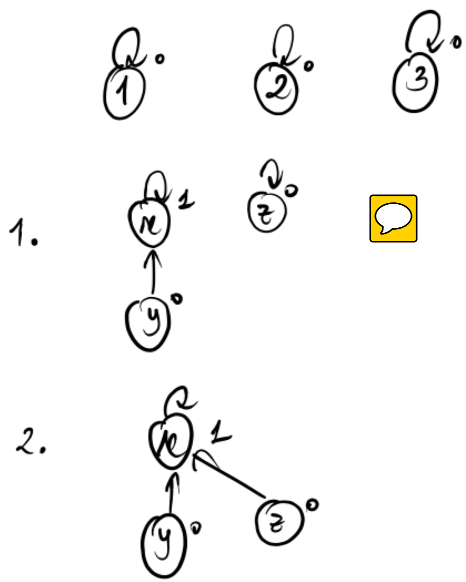
$\textsc{Link}(x, y)$

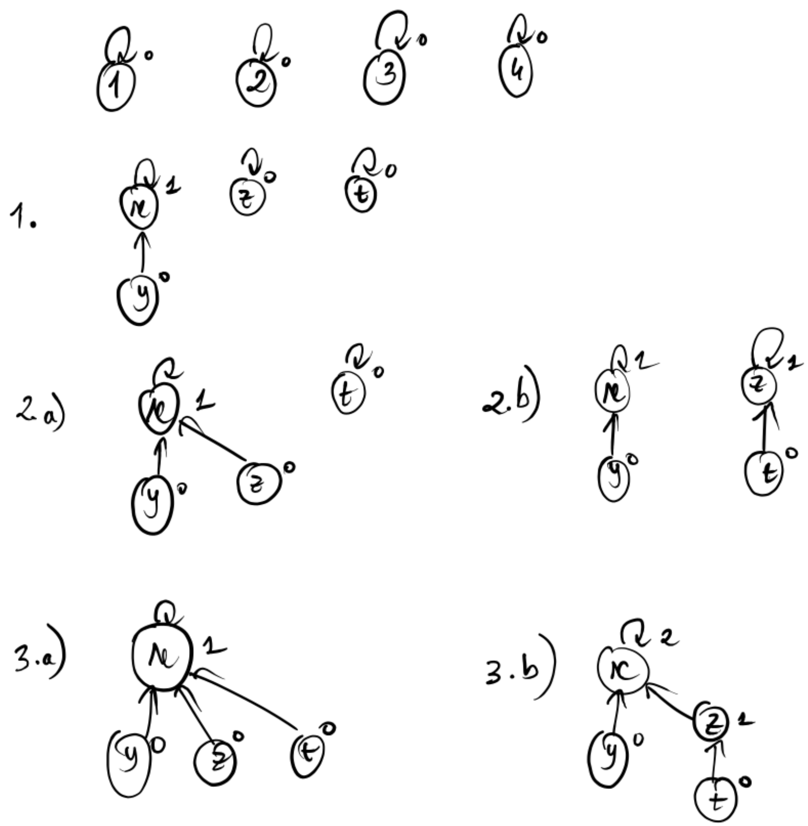
1 **if** $x.rank > y.rank$
2 $\quad y.p = x$
3 **else** $x.p = y$
4 $\quad$ **if** $x.rank == y.rank$
5 $\quad\quad y.rank = y.rank + 1$

Intuition: If one set is a forest with less levels, appending it to the root of the deeper forest will not change the max number of levels.
If they both have equal ranks, appending one to the other will increase the height by one

Exercise 21.1

Given n nodes, start with n disjoint sets and represent all possible forests you get after $n-1$ Union operations. Do this for $n = 3$ and $n = 4$.



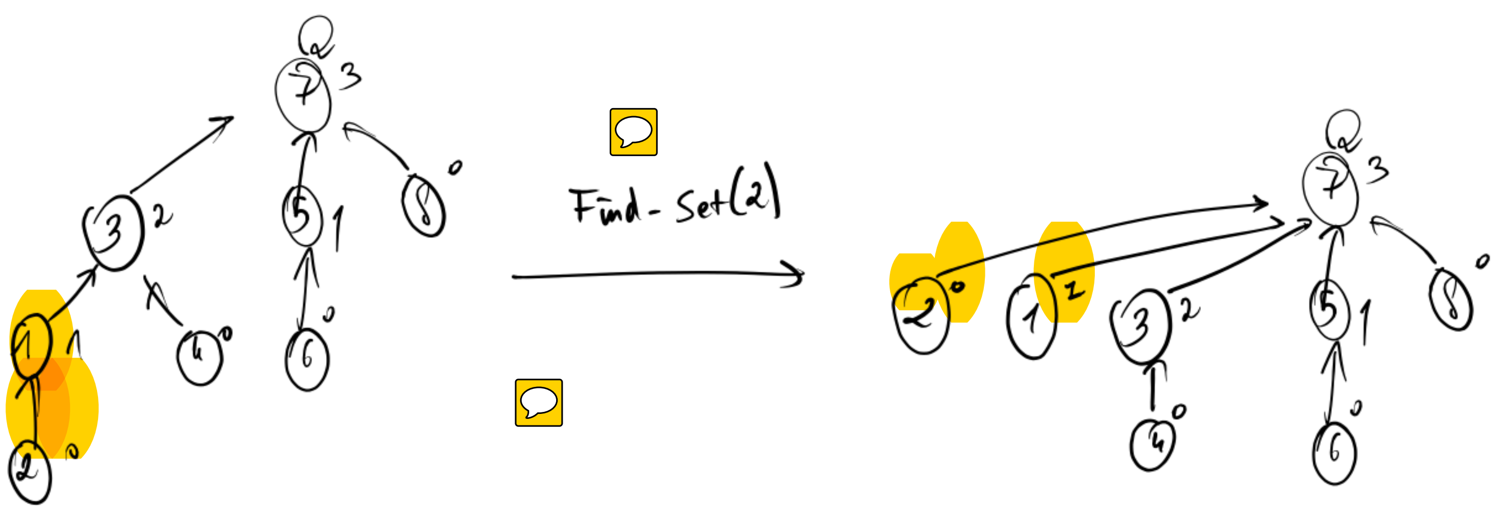


Disjoint-Set Forests

Another Complexity Reduction Trick

FIND-SET(x)
if $x \neq x.p$
    $x.p = \text{FIND-SET}(x.p)$
return $x.p$

PATH COMPRESSION:
Update **x.p (.p.p…)** to point to the tree root



Disjoint-Set Forests

Union by Rank + Path Compression

Claim. The worst case running time for $n-1$ calls to Union is $O(n\ \alpha(n))$.

$\alpha(n)$ is similar to the inverse of Ackermann's function and we have:

$$\alpha(n) \leq 4,\ \forall n,\ 0 < n < 2^{2048}$$

which is much bigger than the number of atoms in the observable universe.

We went from $O(n^2)$ to $O(n \lg n)$ to $O(n\ \alpha(n))$.

Exercise 21.2

Write a nonrecursive version of FIND-SET with path compression.