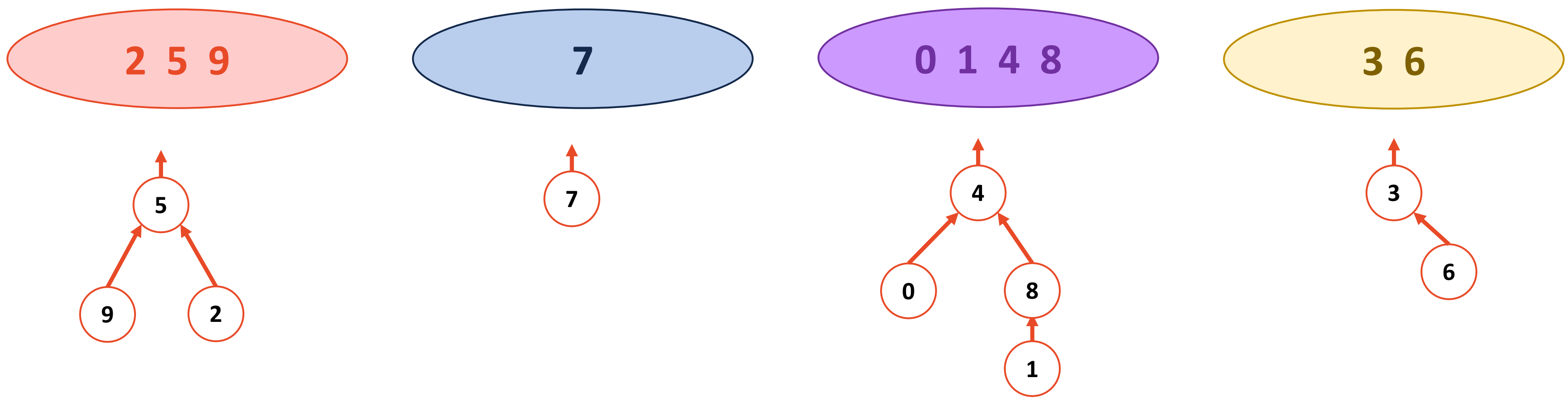
CS 225

Data Structures

October 17 – Disjoint Sets with Path Compression

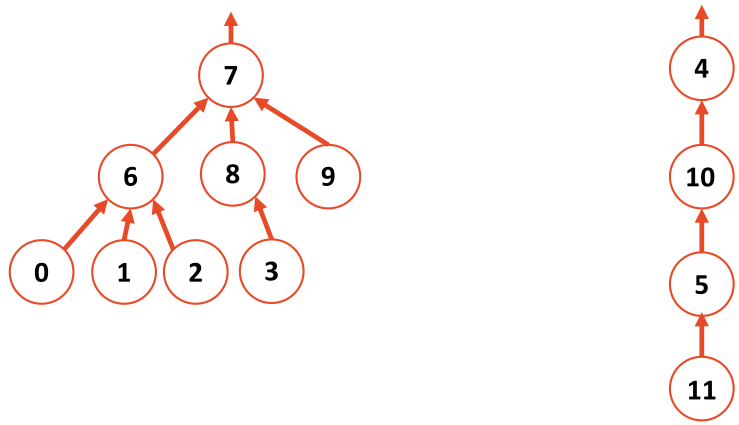
G Carl Evans

Disjoint Sets



0	1	2	3	4	5	6	7	8	9
4	8	5	-1	-1	-1	3	-1	4	5

Disjoint Sets – Smart Union



Union by height/rank

0	1	2	3	4	5	6	7	8	9	10	11
6	6	6	8	-4	10	7	-3	7	7	4	5

Idea: *Keep the height of the tree as small as possible.*

Union by size

0	1	2	3	4	5	6	7	8	9	10	11
6	6	6	8	-8	10	7	-4	7	7	4	5

Idea: *Minimize the number of nodes that increase in height*

We will show the height of the tree is: log(n).

Rank

Base

New UpTrees have Rank of 0

When you join two UpTrees with rank ***r*** the root of the merged tree will have a root changed to ***r* + 1**

Note: without path compression rank is height

Union by Rank - Proof

Much like before we will show that in a tree with a root of rank r there are $nodes(r) \geq \ 2^r$

Base Case: UpTree of rank = 0 has 1 node $2^0 = 1$

Inductive Hypothesis: for all trees of ranks $k, k < r,\ \ nodes(k) \geq 2^k$

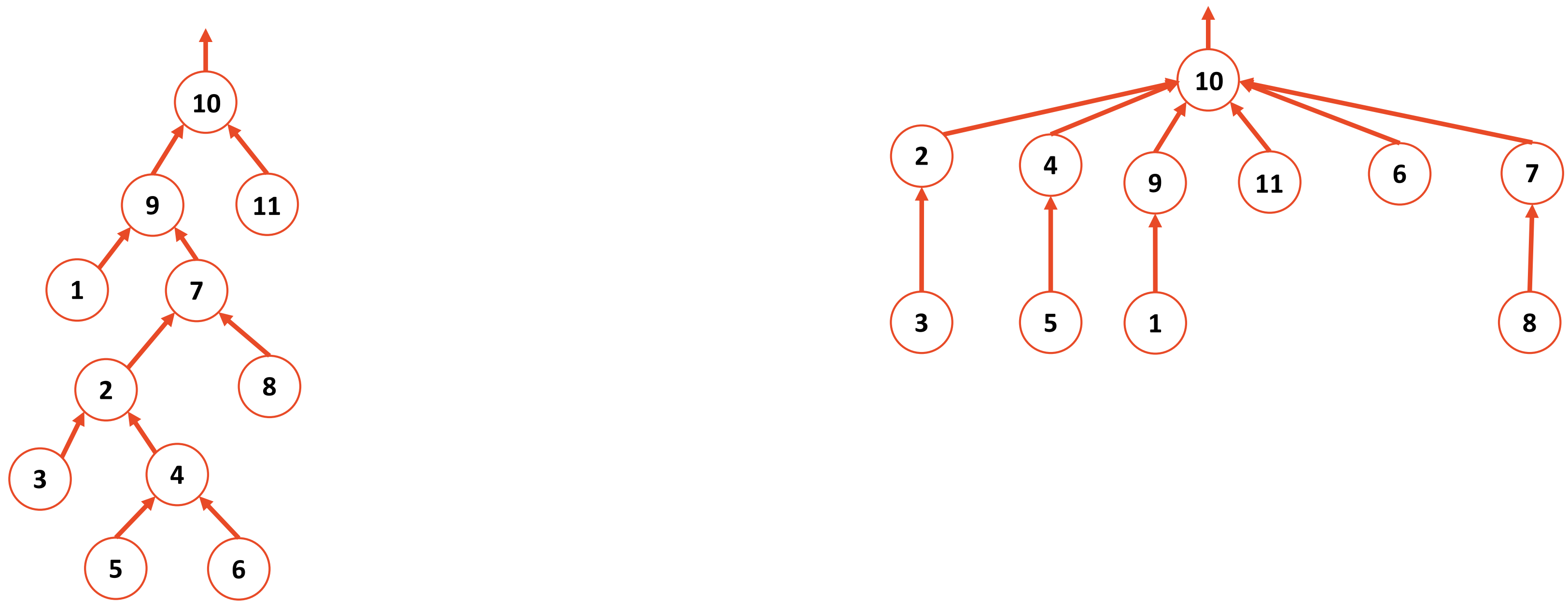
A root of rank r is created by merging two trees of rank $r - 1$

by IH each of those trees have $nodes(r - 1) \geq 2^{r-1}$

so, tree a of rank r has $nodes(r) \geq 2 \times 2^{r-1} \geq 2^r$

Taking the inverse, we get a height of $O(\log(n))$

Path Compression (rank != height)



Rank Properties

1. If x is not a root node, then $rank(x) < rank(parent(x))$.

2. If x is not a root node, then $rank(x)$ will never change again.

Rank Properties

3. If $parent(x)$ changes, then $rank(parent(x)) < rank(parent'(x))$.

4. min(nodes) in a set with a root of rank $r \geq 2^r$.

   This was shown before, and path compression does not change the number of nodes in a set.

Rank Properties

5. Since there are only n nodes the highest possible rank is $\lfloor \log n \rfloor$.

6. For any integer r, there are at most $n/2^r$ nodes of rank $\geq r$.

Amortized

Find needs to be amortized since running the same find multiple times will have different runtimes.

- Find on root and immediate children of root

- Find on everything thing else

Iterated Logarithm Function ($log^{*}n$)

$log^{*}n$ is piecewise defined as

0 if $n \leq 1$

otherwise

$1 + log^{*}(\log n)$

Buckets

- Put every non-root node in a bucket by rank
- The total number of buckets is O(log*n)
- Max nodes in a bucket is n divided by lower bound of the next bucket

Ranks	Bucket
0	0
1	1
2 - 3	2
4 - 15	3
16 – 65535	4
65536 – $2^{65536} - 1$	5

Find(x) How to charge the work

The work of find(x) is the steps taken on the path from a node x to the root of the uptree containing x

Case 1: The step from u to v moves from one bucket to another we charge that to x.

Find(x) How to charge the work

Case 2: The step from u to v and u and v are in the same bucket

1. The rank of u will never change

2. Every charge will increase the rank of parent

How many total charges of this kind in a bucket?

Final Result

Even Better

In case that seems to slow tightest bound is

$$\Theta(m\,\alpha(m,n))$$

Where $\alpha(m,n)$ is the inverse Ackermann function which grows much slower than log*n.

Proof well outside this class.