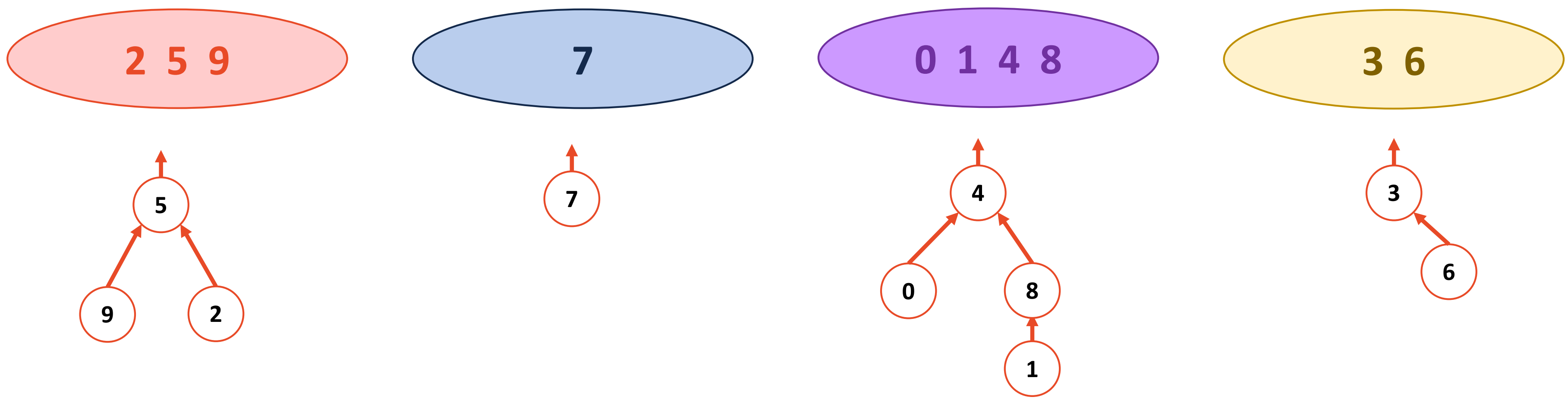
CS 225

Data Structures

October 14 – Disjoint Sets and Iterators

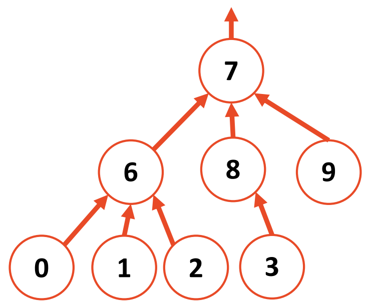
G Carl Evans

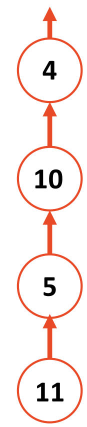
Disjoint Sets



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

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).

Union by Size

To show that every tree in a disjoint set data structure using union by size has a height of at most O(log n) we will show that the inverse.

Base Case

Inductive Hypothesis

Union by Size

Case 1

Union by Size

Case 2

Union by Height

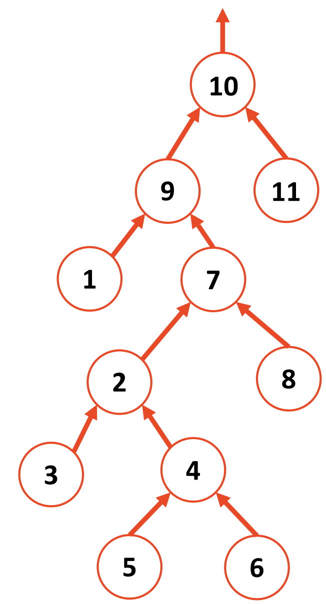
Much like before we will show the min(nodes) in a tree with a root of height $k \geq 2^k$

Base Case

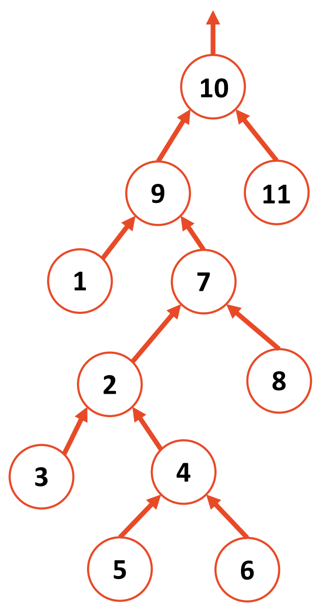
IH

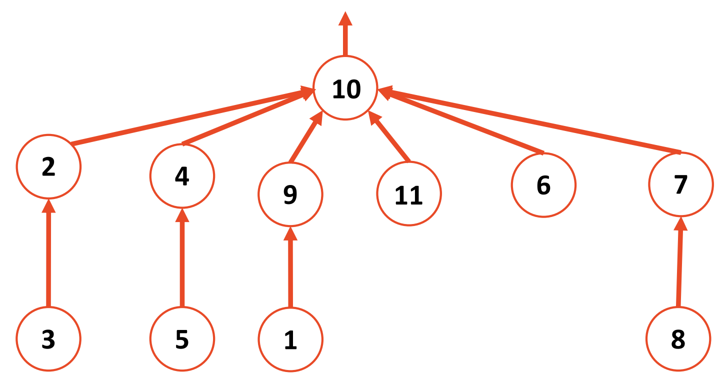
Disjoint Sets Find

```
int DisjointSets::find(int i) {
  if ( s[i] < 0 ) { return i; }
  else { return find( s[i] ); }
}
```



Path Compression





Union by Height - Rank

Base

New UpTrees have Rank =

When you join two UpTrees

Union by Rank

1. For all non-root nodes ***x, rank(x) < rank(parent(x))***

2. Rank only changes for roots and only up

Disjoint Sets Analysis

The **iterated log** function:

The number of times you can take a log of a number.

log*(n) =

0 , $n \leq 1$

1 + log*(log(n)) , $n > 1$

What is **lg*(2^{65536})**?

Disjoint Sets Analysis

In an Disjoint Sets implemented with smart **unions** and path compression on **find**:

Any sequence of **m union** and **find** operations result in the worse case running time of O(____________),

where **n** is the number of items in the Disjoint Sets.