Path Compression

Learning Objective

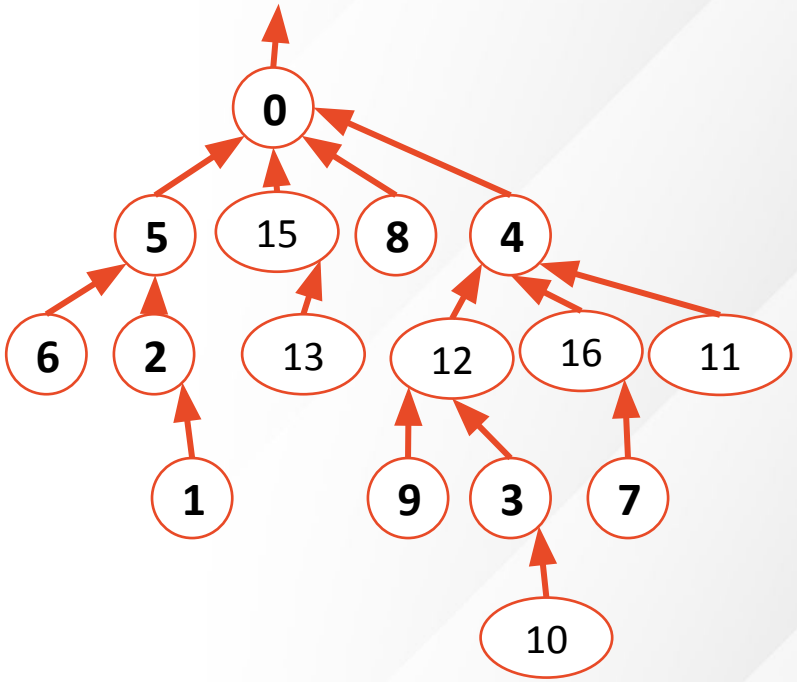
1. Construct possible uptrees
2. Understand Path Compression
3. Implement Union by Rank

Constructing Possible Uptrees

Disjoint Set ADT - Smart Union

Find - O(log n)

Union - O(1)



```
1  int DisjointSets::find(int i) {
2    if (data[i] < 0){
3      return i;
4    }  else{
5      data[i] = find(data[i]);
6      return data[i];
7    }
8  }
```

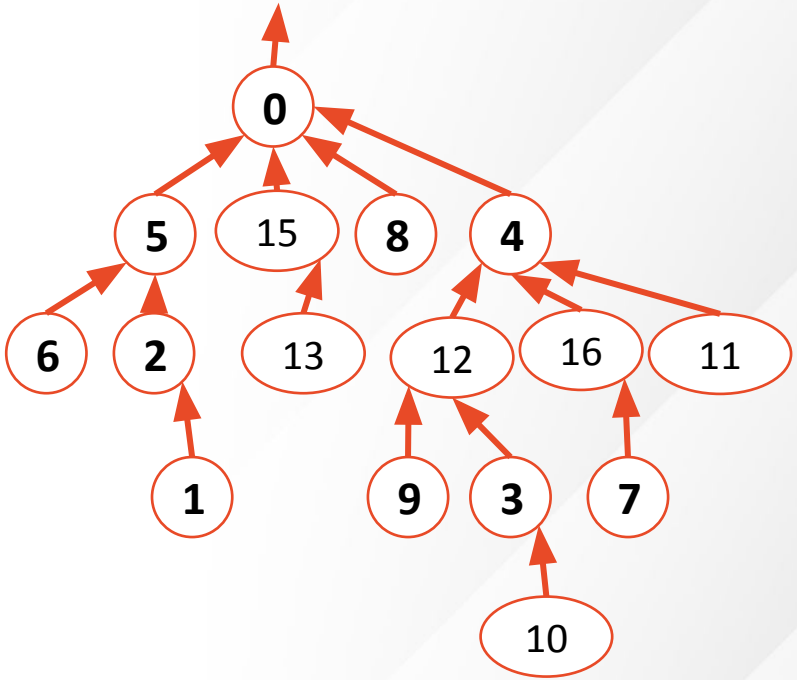
Path Compression

```
1  int DisjointSets::find(int i) {
2    if (data[i] < 0){
3      return i;
4    }  else{
5     data[i] = find(data[i]);
6     return data[i];
7    }
8  }
```

Path Compression

Find - O(log n)

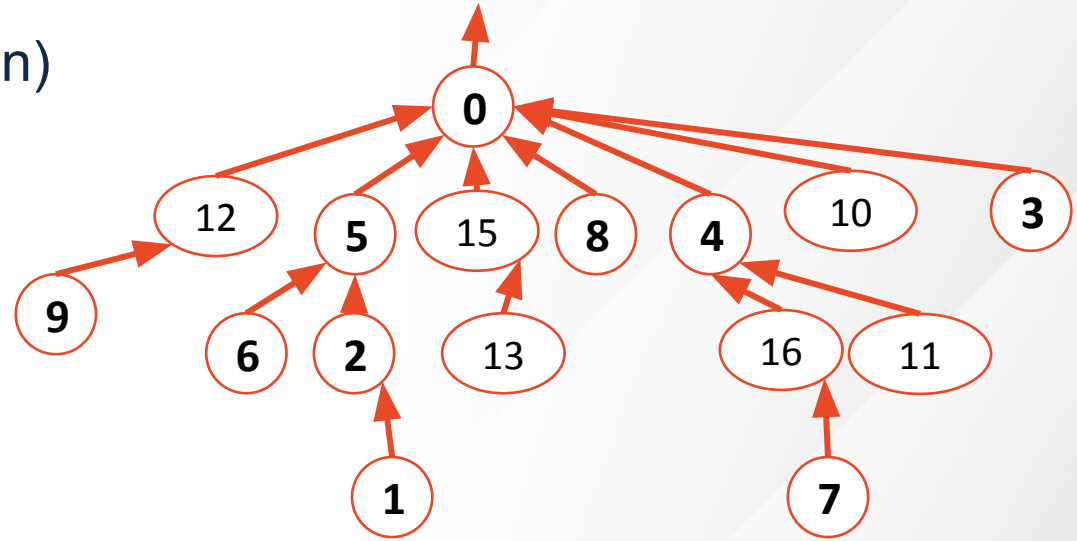
Union - O(1)



Path Compression

Find - O(log n)

Union - O(1)



Union by Rank

Base case:

- new uptrees have a rank of 0

Inductive case:

- if uptrees have the same rank, the rank of the new root increases by 1

Union by Rank Example

Union by Rank vs. Union by Height