CS 333
Disjoint Sets

Disjoint sets are used to create different and non-overlapping sets. This is needed in Kruskals because the edges are selected according to weight and it is not always possible to have a single set of edges. As such, there are many different sets that are required. Overtime, these will be merged (union) together to complete a single tree.

The implementation for Kruskals is a disjoint set.

The size is the number of vertices.

The code is from the Cormen book and path compression and rank are present.
Three functions for a disjoint set:

MAKESET

Initializes a set.

```
MAKE-SET(x):
    p[x] = x
    rank[x] = 0
```

FIND

Finds the root of x in the disjoint set. It also does path compression, as such, it will place the parent of x to whatever it finds.

```
FIND-SET(x)
    if x !=  p[x]
    {
        p[x] = FIND-SET(p[x])
    }
    return p[x]
```

UNION/LINK

Takes two sets and combines them together. It will keep the rank of each set. When a set is combined, if their rank is the same, it will be incremented by 1. Otherwise, it will not be incremented. The smaller tree will attached itself to the larger tree's root, according to rank.

```
UNION(x,y)
    if rank[x] > rank[y]
    {
        p[y] = x
    }
    else
    {
        p[x] = y
        if rank[x] == rank[y]
        {
            rank[y]++
        }
```

}

EXAMPLE
Let's do an example.

N = 4. Lets make a disjoint set.
MAKESET:
Our array size of p is 4 and the contents are: 1,2,3,4.
Then we go through and set all of the ranks (1-4) to 0.

***The indexing of this example is 1 based, NOT 0!**

p: [1] [2] [3] [4]
r: [0] [0] [0] [0]

Let's create 2 sets, linking 1 and 2 together, and 3 and 4.
So:

STEP 1
Union 1 and 2:
Union(1,2):
x = 1, y = 2.
Is the rank of 1 (0) is greater than the rank of 2 (0)? No, they are both 0.
So, we do the ELSE from the first if statement in the Union function.
p[1] = 2.
The ranks are the same – as they are both 0.
rank[2] = rank[2] + 1 -> 1

This is what our disjoint set looks like now:
p: [2] [2] [3] [4]
r: [0] [1] [0] [0]

STEP 2
Union 3 and 4:
Union(3,4):
x = 3, y = 4
Is the rank of 3 (0) greater than the rank of 4 (0)? No, they are both 0.
So we do the ELSE from the first if statement in the Union function.
p[3] = 4
The ranks are the same – as they are both 0.
rank[4] = rank[4] + 1-> 1

This is what our disjoint set looks like now:
p: [2] [2] [4] [4]
r: [0] [1] [0] [1]
STEP 3
Find 1:
if 1 != p[1] ----> 1 != 2

This statement is successful so we go into the if statement.
p[1] = FIND-SET(2) (as p[1] = 2)
So we check again:
if 2 != p[2] ----> 2 == 2
So we just return p[2], which is 2.
Thusly from above, p[1] becomes 2 and we return p[1] , which is 2.
Thusly, our answer is 2.

STEP 4
Find 3:
if 3 != p[3] ----> 3 != 4
This statement is successful so we go into the if statement.
p[3] = FIND-SET(4) (as p[3] = 4)
So we check again:
if 4 != p[4] ----> 4 == 4
So we just return p[4], which is 4.
Thusly from above, p[3] becomes 4 and we return p[3], which is 4.
Thusly, our answer is 4.

STEP 5
Union 2 and 3:
This is our current disjoint set.
p: [2] [2] [4] [4]
r: [0] [1] [0] [1]
Union(2,3)
x = 2, y = 3
The rank of x (1) is greater than the rank of y (0).
As such, we do the p[y] = x line.
p[3] = 2
Our new disjoint set is now:
p: [2] [2] [2] [4]
r: [0] [1] [0] [1]

STEP 6
Now, lets do another find. Let's do a find on 2. Our answer will be 2 because 3 goes to 2.
Find 3:
if 3 != p[3] ----> 3 != 2
This statement is successful so we go into the if statement.
p[3] = FIND-SET(2) (as p[3] = 2)
So we check again:
if 2 != p[2] ----> 2 == 2
So we just return p[2], which is 2.
Thusly from above, p[2] becomes 2 and we return p[2], which is 2.
Thusly, our answer is 2.
STEP 7
Union 1 and 4:
Union(1, 4)
x = 1, y = 4
The rank of x (0) is not greater than the rank of y (1).

As such, we do the else statement: p[1] = 4.
Are the ranks equal? No.
Our new disjoint set is now:
p: [4] [2] [2] [4]
r: [0] [1] [0] [1]

STEP 8
Union 2 and 4.
Union(2, 4)
x = 2, y = 4
The rank of x (1) is not greater than y (1).
As such, we do the else statement: p[2] = 4.
Are the ranks equal? Yes. We increment the ranks.
rank[y] = rank[y] + 1
Our new disjoint set is now:
p: [4] [4] [2] [4]
r: [0] [1] [0] [2]

STEP 9
Find 3. We should return 4, because 3 points to 2, which in turn points to 4.
1.) if 3 != p[3] ----> 3 != 2
This statement is successful so we go into the if statement.
p[3] = FIND-SET(2) (as p[3] = 2)
So we check again:
2.) if 2 != p[4] ----> 2 ! = 4
This statement is successful so we go into the if statement.
p[2] = FIND-SET(4) (as p[2] = 4)
So we check again:
3.) if 4 != p[4] ----> 4 == 4
So we just return p[4] to list item 2, which is 4.
p[2] becomes 4. We then return 4 to list item 1. p[3] is now 4.
Our answer is 4.

Our new disjoint set is now:
p: [4] [4] [4] [4]
r: [0] [1] [0] [2]

Path compression has broken this down such that every lookup is efficient.