1.5.14: Weighted quick-union by height. Develop a UF implementation that uses the same basic strategy as weighted quick-union but keeps track of tree height and always links the shorter tree to the taller one. Prove a *logarithmic upper bound* on the height of the trees for N sites with your algorithm.

Overview: Max height will have a total of $logN$ height. Initially each node seperate and has height of 1. 2 arrays are required. One will be to keep track of the height. The other will keep track of parent/root nodes.

Pseudocode below:

```
//Pseudocode: Weighted quick-union by height

int height[N]
int parent[N]

count = n //counts number of components

for i in range(n): //initialize array trackers
    height[i] = 1 //initially all nodes have height of 1
    parent[i] = i

function find(p): //finds root of a node iteratively
    while(p != parent[p]):
        p = parent[p]

    return package

function is_connected(p, q):
    return find(p) equals find(q) //returns a boolean

void function Union(p, q):
    root_p = find(p)
    root_q = find(q)

    if(root_p equals root_q): //already connected
        return

    else if(height[root-p] > height[root_q])
        parent[root_q] = root_p

    //connect smaller cluser to larger one using parent
    else if(height[root_p] < height[root_q]):
        parent[root_p] = root_q

    else:
        //if equal but not yet connected, connect q to p's parent

        parent[root_q] = root_p
        height[root_p] += 1 //since height of p now also consists of queryCommandEnabled

        count-- //since once less component in list of clusters
```