William Argus
A12802324

ECE 208 Homework 1 Write-up

Recursive formula:

```
lca(u,v)
    if LCA[u][v] exists
        return that value
    else
        create it
    if u is an ancestor of v,
        add it LCA[u][v]
        return u
    else
        call lca(u,v) on all parents(u) and v
        add value of lca(parents(u),v) to LCA[u][v]
        return value of lca(parents(u),v)
```

This is derived by considering the fact that the parents of u will have the same common ancestors as u and v. So by iterating “up” the graph until that point is reached, or until the there is no parents, the LCA’s are found for the given set of u and v. The code at the beginning serves to initialize the dictionary entries as needed, making sure not to overwrite ones that already exist and instead using the LCA’s that have already been found, that’s the dynamic part of this programming assignment.

Base case: u is an ancestor of v (and since we have reached this case, none of u's descendants are ancestors of v, so u must be in LCA[u][v]). So for every pair, the base case will either be reached eventually, or the root will be found, which means that for that given recursion path, no LCA exists for the two pairs, (but may exist in other recursion paths, hence why the recursion is called for all parents of u). Using this information, the base case just mentioned is found naturally.

Run Time:

To find the ancestors of all nodes (done above lca(u,v) in the bigger function of find_LCAs), run time is looping through all nodes parents n times, so looping through each node at every node, hence for this part, the run time is n^2.

For the recursion, the LCA for a pair will only need to be found once, as it is re-used as necessary with the dynamic programming. So to find the LCA of each pair takes the amount of time of unique pairs there are, which is approximately n^2.

So the runtime in total is 2n^2.