Jacob Archer
6/24/24

Program 3 Part B

To make finding the closest shared ancestor between two fork nodes more efficient, we can tweak the CryptoForkNode class by adding a parent pointer. This little addition will let us find the ancestor in O(h) time, where h is the height of the tree.

Here's the gist of the new approach. First, we modify the CryptoForkNode class to include a parent pointer. This makes it easy to trace the path from any node back up to the root.

The algorithm works like this:

1. Calculate the depth of both nodes. This means moving from each node up to the root and counting the levels.

2. Once we know the depths, we bring both nodes to the same level by moving the deeper node up until it's even with the shallower node.

3. With both nodes at the same depth, we move up the tree together until we find the common ancestor.

Here's some pseudocode to illustrate:

```
CryptoForkNode* findClosestSharedAncestor(CryptoForkNode* node1, CryptoForkNode* node2) {
    // Get the depths of both nodes
    int depth1 = getDepth(node1);
    int depth2 = getDepth(node2);

    // Make sure both nodes are at the same depth
    while (depth1 > depth2) {
        node1 = node1->parent;
        depth1--;
    }

    while (depth2 > depth1) {
        node2 = node2->parent;
        depth2--;
    }

    // Move up together until we find the common ancestor
    while (node1 != node2) {
        node1 = node1->parent;
        node2 = node2->parent;
    }

    return node1; // or node2, they are the same at this point
}
```

```
int getDepth(CryptoForkNode* node) {
    int depth = 0;
    while (node->parent != nullptr) {
        node = node->parent;
        depth++;
    }
    return depth;
}
```

In the worst-case scenario, this algorithm might need to trace the path from both nodes all the way up to the root. But even then, each step—calculating depths, aligning depths, and finding the common ancestor—only takes O(h) time. So, the overall time complexity remains O(h).

By adding a parent pointer to each node and using this simple algorithm, we can quickly and efficiently find the closest shared ancestor of two nodes, even if they are deep in the tree. This makes the whole process much faster and more efficient, even in the worst-case scenario.