RBT

Jingle Ballers

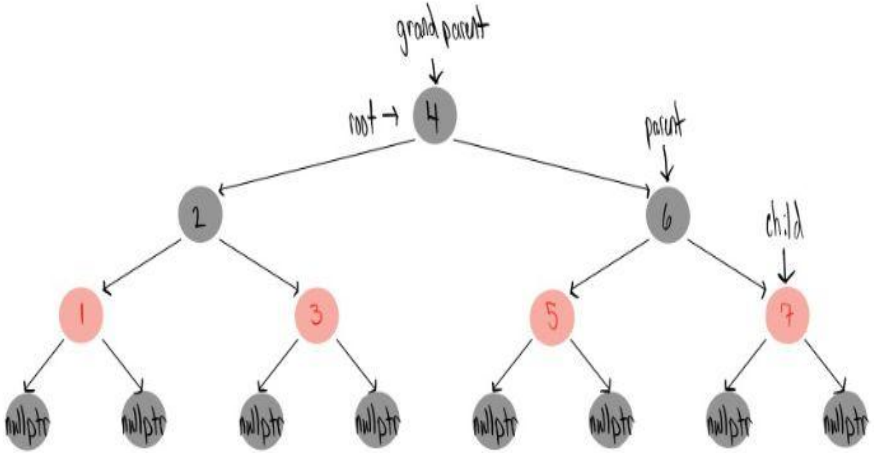
Initial Implementation

- We started implementation in two separate groups, Visualization and the Tree
- For the tree, we started out with the code found in the textbook
- Our tree was over balancing, making the tree complete
- We had to pivot to a new design

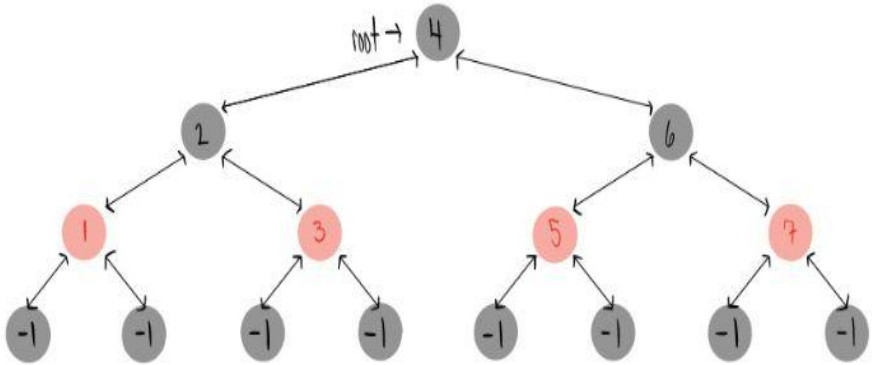
Redesign

- Changed nodes to be doubly linked between parent and child
- Changed null nodes to be represented with a sentinel value of -1 rather than using nullptr

Before:



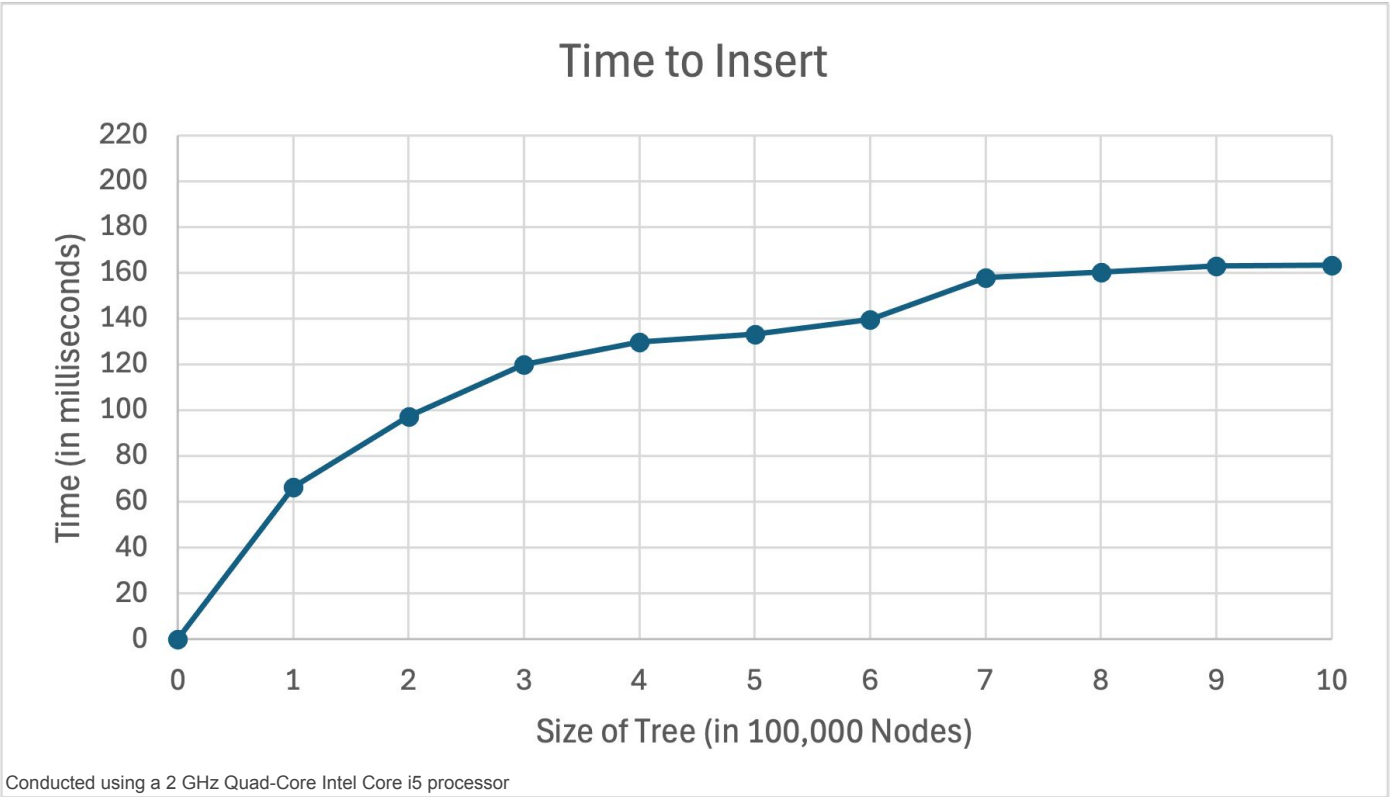
After:



Issues Encountered

- Tree wasn’t rebalancing when root was deleted
- Certain situations caused a rotation in the opposite direction they were supposed to
- The parent data member of the new node wasn’t properly being reassigned
- Certain rebalance cases were executing when they weren’t supposed to
- Deleting a node rebalanced the wrong side of the tree
- Various bugs were left unfixed from when null nodes were represented using nullptr rather than a sentinel value
- Determining when to rotate after removing a node

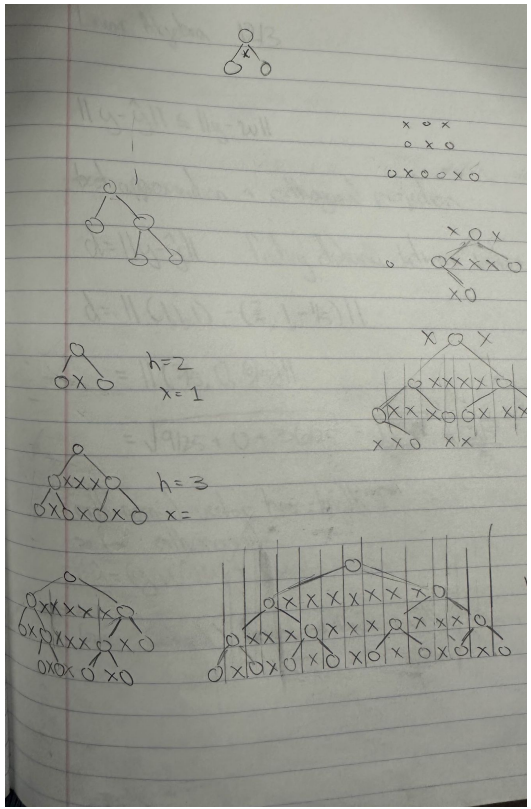
Performance

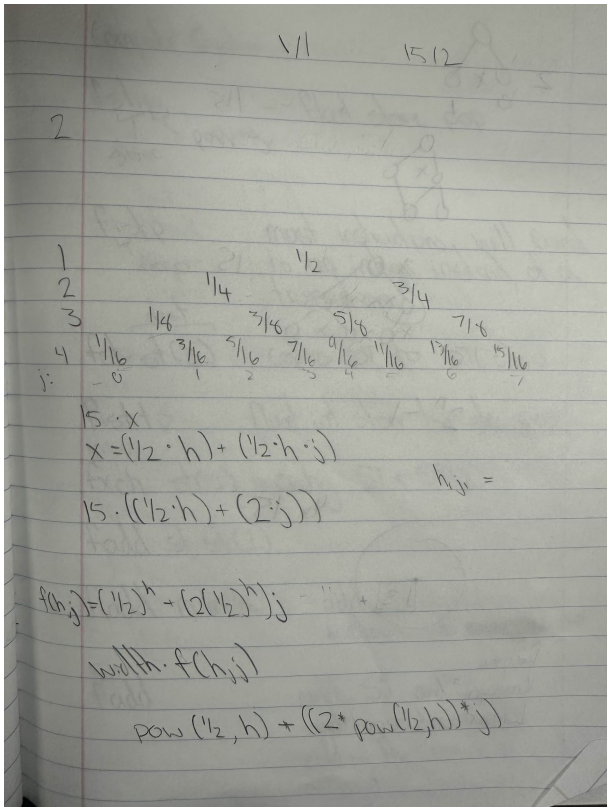


Visual Side

Noah and Ethan P.

Early Problems

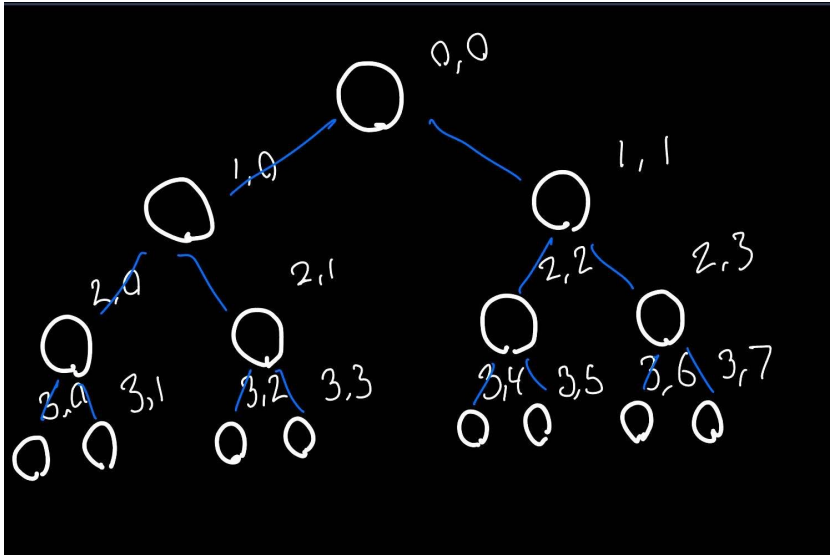


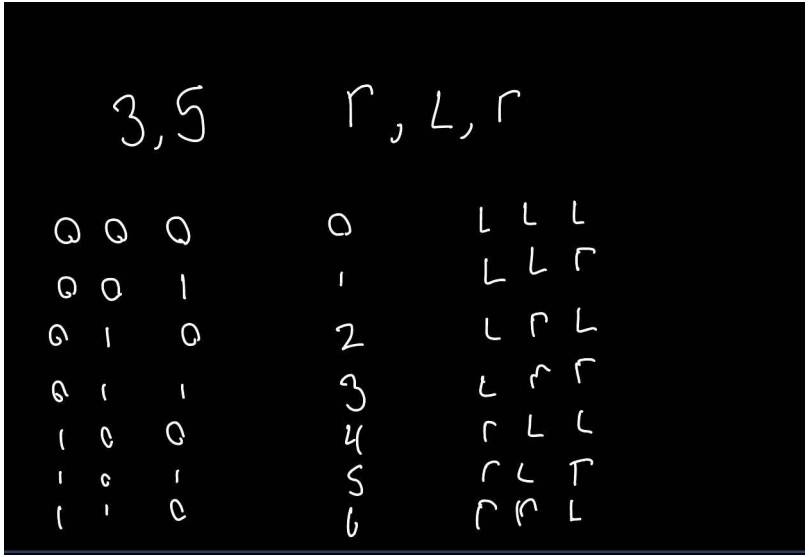


Making it Modular

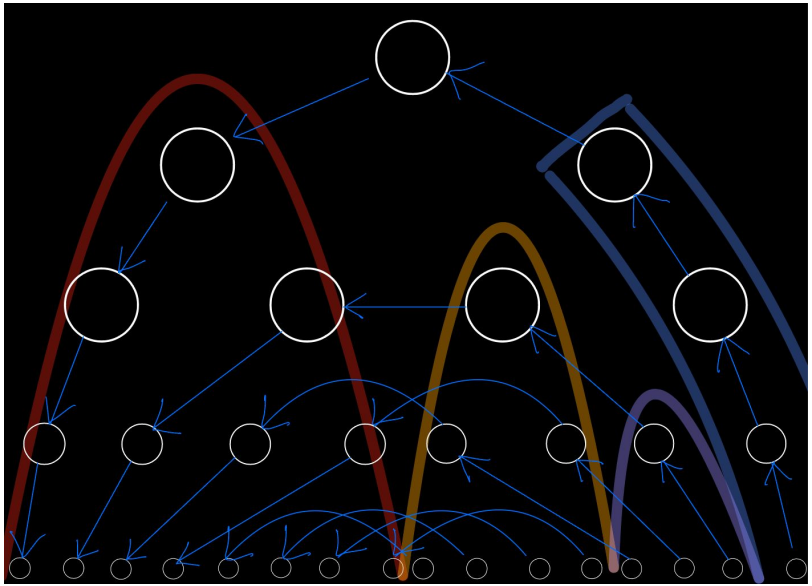
```
24  struct Instructions {
25      RedBlackTree* treeBefore;
26      bool right;
27      int depth;
28      int breadth;
29
30      Instructions(RedBlackTree* tree, bool r, int d, int b)
31              : treeBefore(tree), right(r), depth(d), breadth(b) {}
32  };
33
34  queue<Instructions> treeInstructions;
35
```

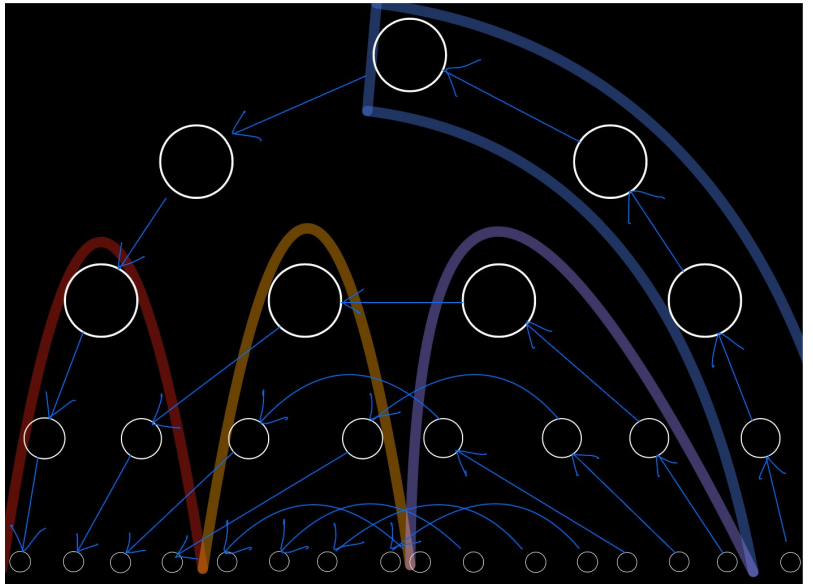
Finding the Node of Rotation



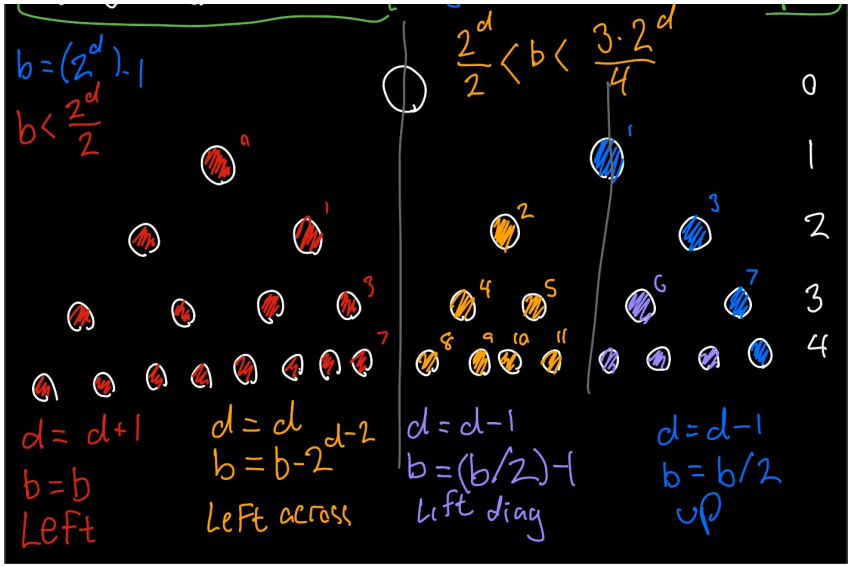


Deciding Where the Nodes Go





Moving the Nodes




```
switch (dir) {
    case 0://Up
        x2 =(1920 * (2 * (breadth/2) + 1)) / (pow(2, (depth-1) + 1));
        y2 =(1000 * ((depth-1) + 1))/(scale+1);
        break;
    case 1://Left
        x2 =((1920 * (2 * (breadth*2)+1)) / (pow(2, ((depth)+1) + 1)));
        y2 =(1000 * ((depth+1) + 1))/(scale+1);
        break;
    case 2://Right
        x2 =(1920 * (2 * (2*breadth+1) + 1)) / (pow(2, (depth+1) + 1));
        y2 =(1000 * ((depth+1) + 1))/(scale+1);
        break;
    case 3://Left Across
        x2 =(1920 * (2 * (breadth - pow(2,depth-2) + pivY) + 1)) / pow(2, depth + 1);
        y2 =(1000 * ((depth) + 1))/(scale+1);
        break;
    case 4://Right Across
        x2 =(1920 * (2 * (breadth + 1 - pivY ) + 1)) / (pow(2, depth + 1));
        y2 =(1000 * ((depth) + 1))/(scale+1);
        break;
    case 5://Left Diagonal
        x2 =(1920 * (2 * (breadth/2) - 1)) / (pow(2, (depth-1) + 1));
        y2 =(1000 * ((depth-1) + 1))/(scale+1);
        break;
    case 6://Right Diagonal
        x2 =(1920 * (2 * (breadth) + 1)) / (pow(2, (depth-1) + 1));
        y2 =(1000 * ((depth-1) + 1))/(scale+1);
        break;
}
```