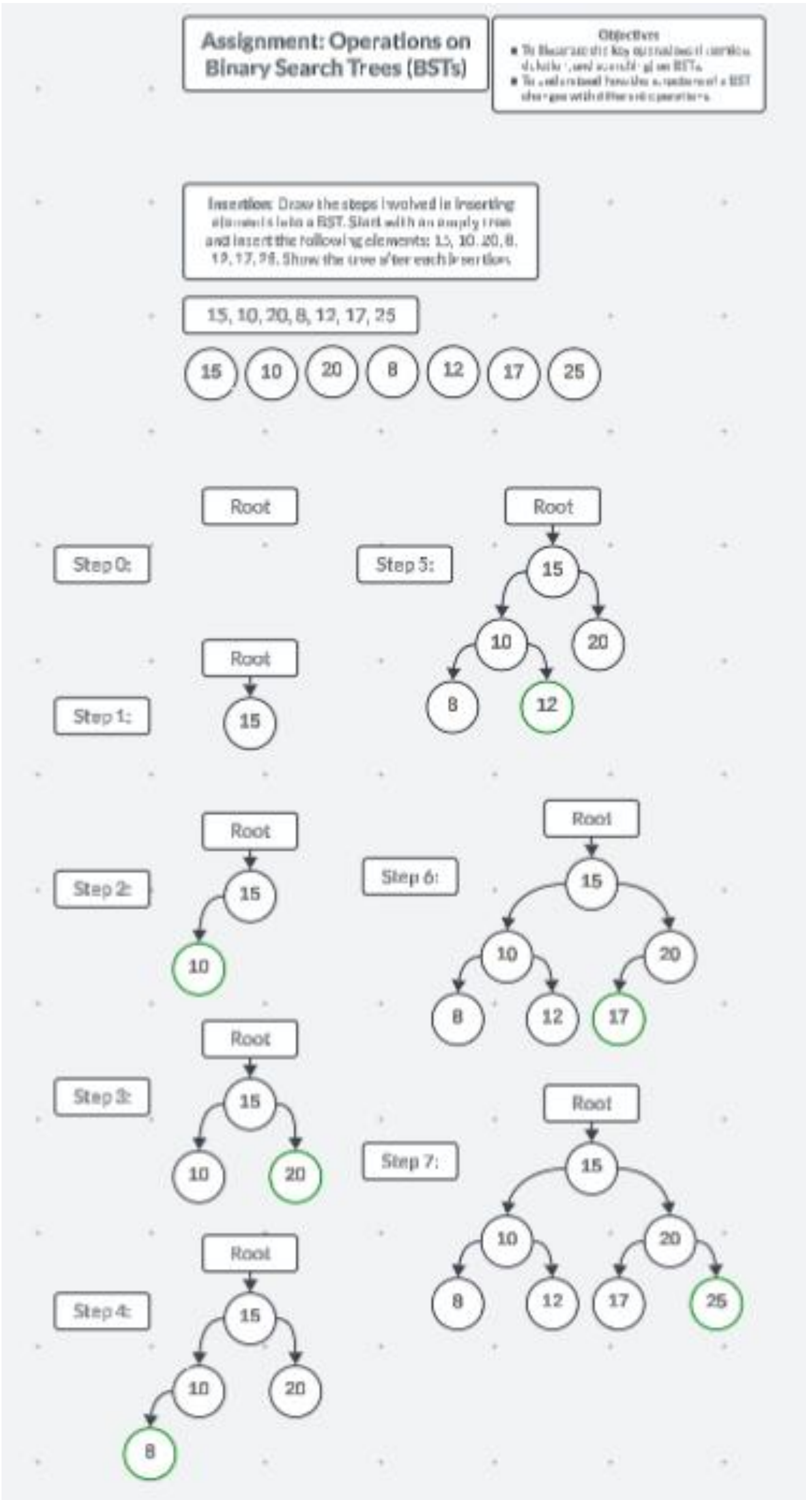
Keiffer Tan

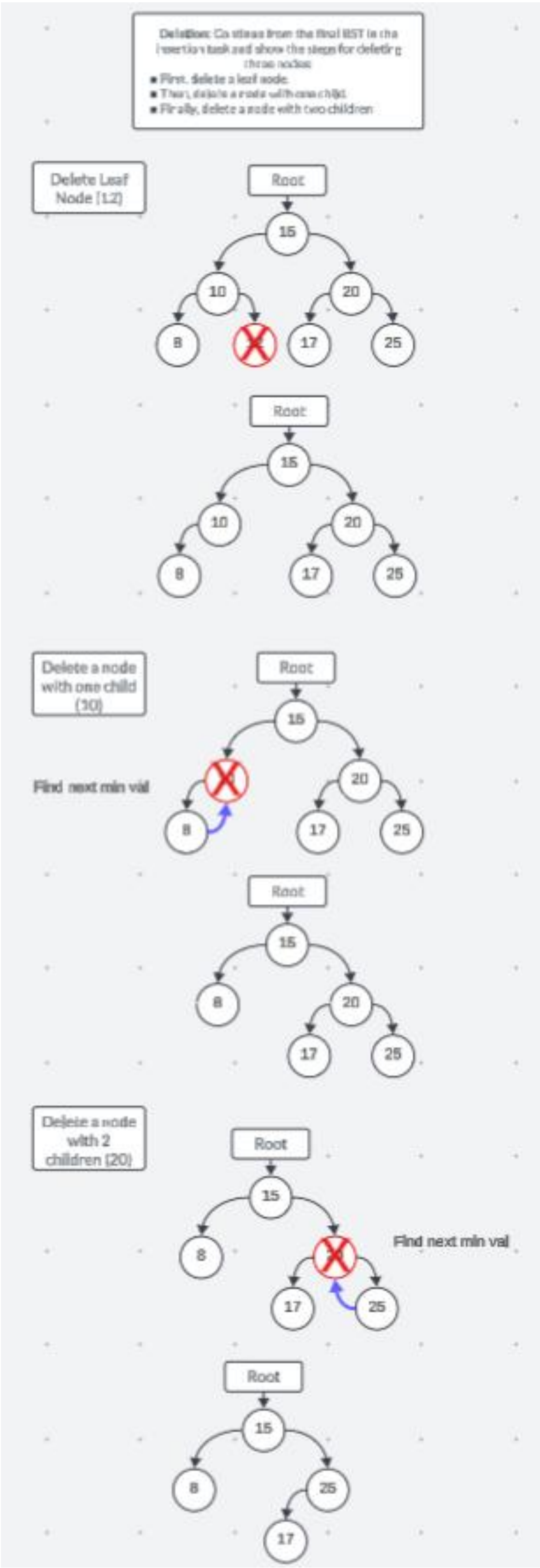
Assignment: Operations on Binary Search Trees (BSTs)



For insertion, if the tree is empty, the first node inserted will be the root node.

For each subsequent insertion, we compare the inserted value to the root and then every consecutive node. The comparison is whether the inserted value is “less than” or “greater than” the current node. From there, we traverse either to the left or right child.

Keiffer Tan



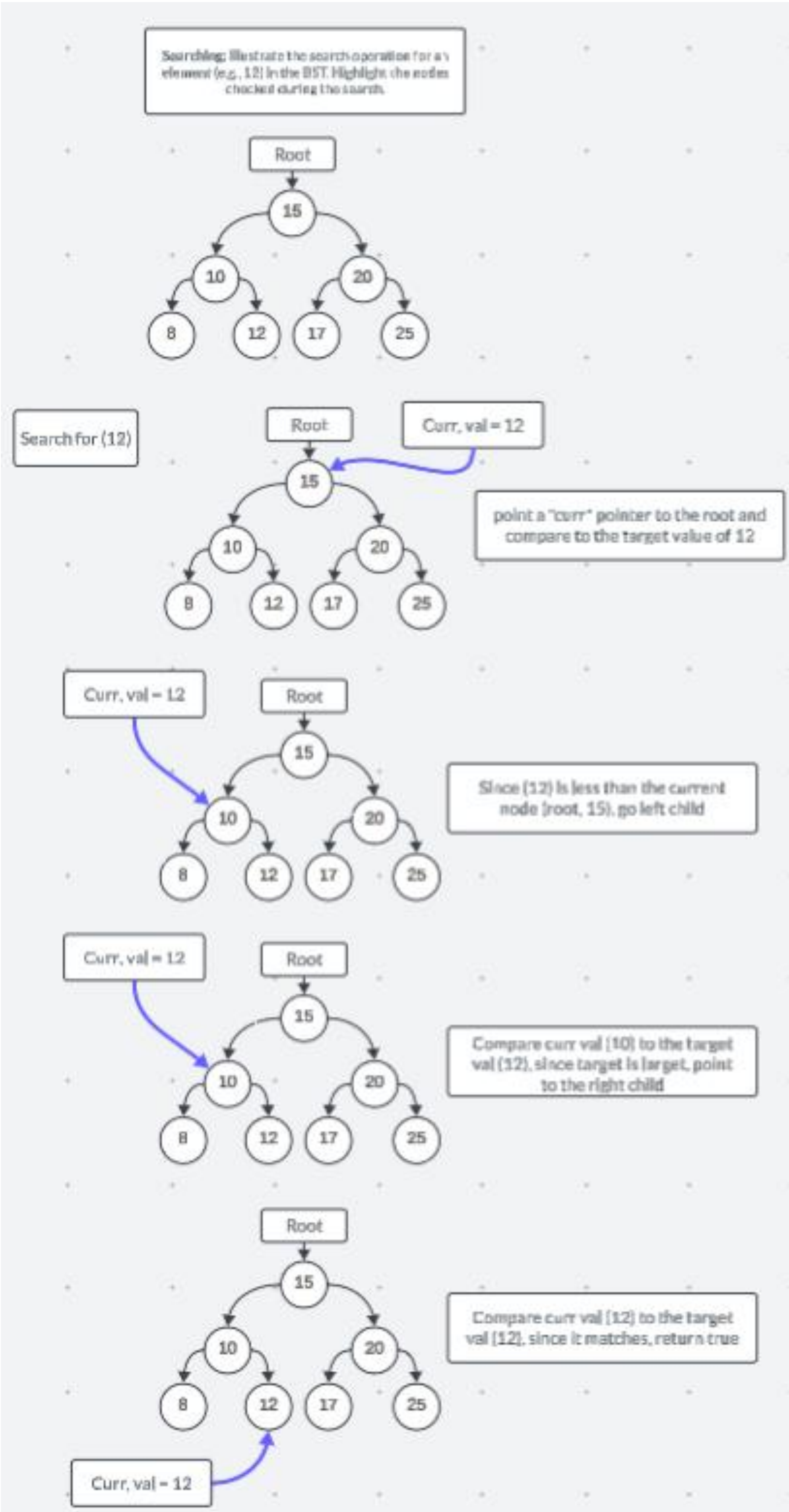
With deletion, we traverse the tree starting with the root node and compare whether the value of the node matches the target value that is to be deleted.

In the first operation, I wanted to delete a leaf node (12), so I traverse the tree first compared with the root (15). The target (12) is less than, so I go to the left child (10) and compare that to the target value (12). Since it the target value is less than, we further traverse to the right child (12). We compare the current node value (12) with the target value (12) and find that it is a match. So, we delete that node from the tree

I want to delete a node with one child (10). Once we traverse the tree to delete the node with one child, we first need to find the next smallest sequential value. Then move it to the position of the deleted node

Deleting a node with 2 children (20), we want to find the next sequential of the node. We go to the right child and then check if the right child has any left children. If so, traverse down, but in this case there are none. We take (25) and place it in deleted node.

Keiffer Tan



As long as the BST is balanced, the search method will always be super-efficient! Since we the root will essentially be the "middle" value and once we go down one branch, we are narrowing our options down by half every time.