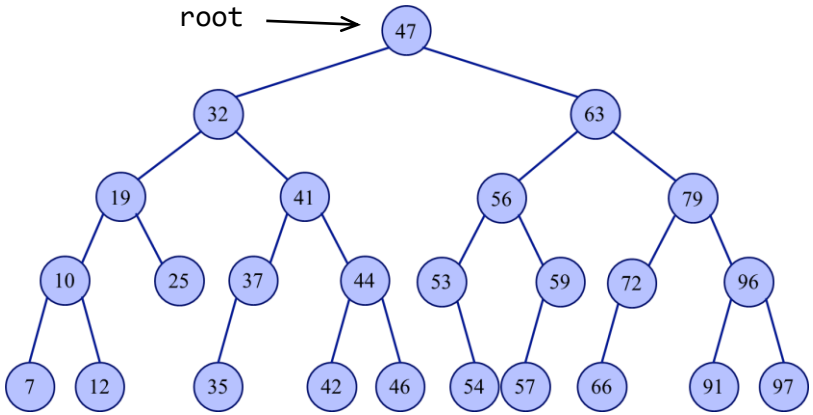
Binary Search Tree Property:

For every node n in the tree:
- All node's in n's left subtree have keys less than n
- All node's in n's right subtree have keys greater than n

Searching for an element in a BST:



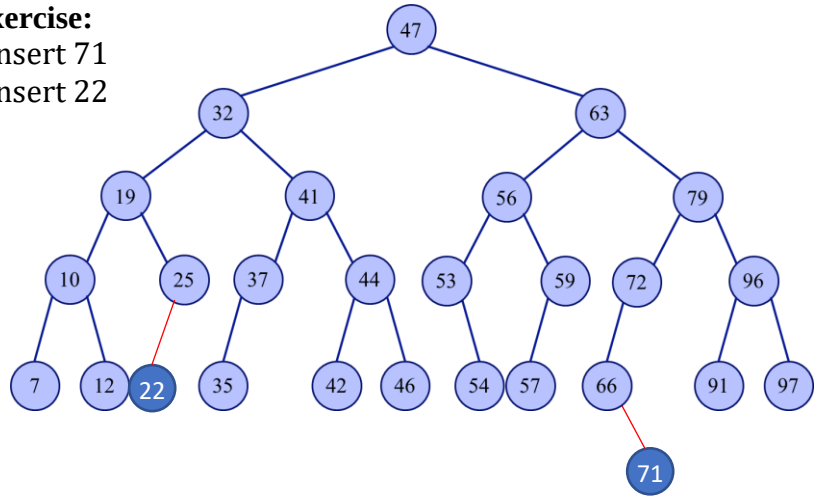
BST.java (partial)

```
31  public TreeNode find(int key) {
32    return find(root, key);
33  }
36  public TreeNode find( TreeNode cur, int key ) {
37    // if cur is null, key cannot be found
38     if (cur == null) {
39         return null;
40     }
41    // if we found a node with the correct key, return it
37
38    if (cur.data == key) {
39        return cur;
40    }
41    // search the left subtree for the key
42
43    if (key < cur.data) {
44        return find( cur.left, key );
45    }
46    // search the right subtree for the key
47
48    return find( cur.right, key );
49  }
```

Insertion into a BST:

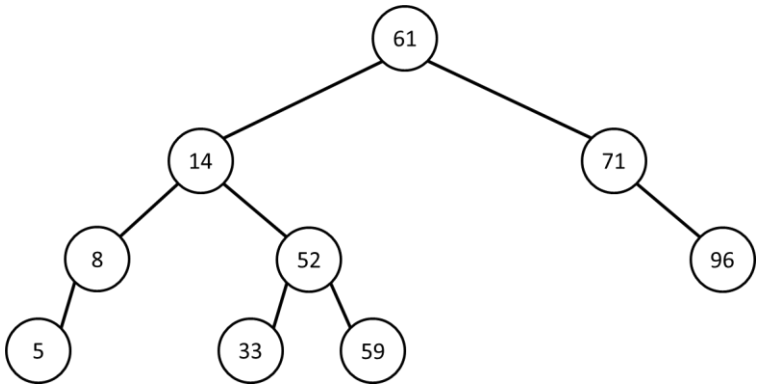
- Search the tree to identify the position to insert the new node
- Add the new node as a child of the leaf node

Exercise:
- insert 71
- insert 22



Insert the following elements into an initially empty BST:

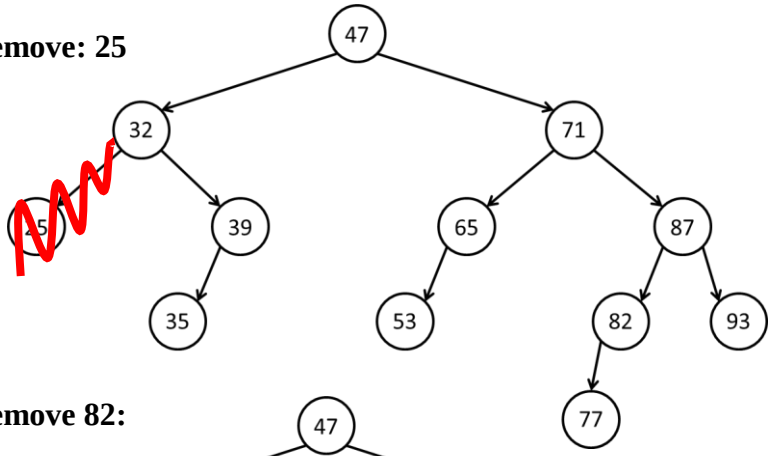
$61, 14, 52, 8, 71, 5, 96, 33, 59$



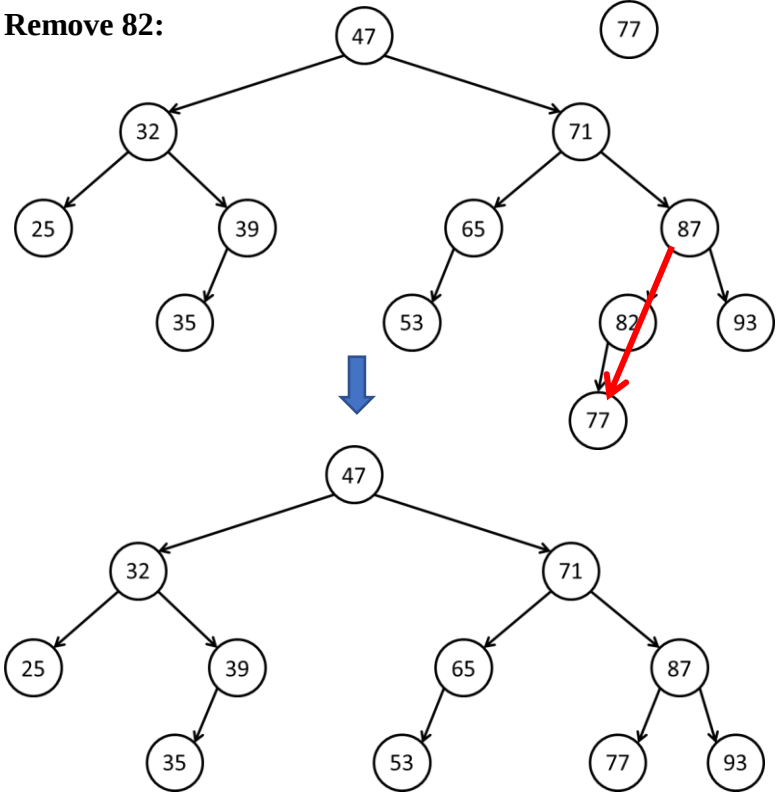
Removal from a BST:

1. Leaf node: Delete node, parent points to null instead of node.
2. Node with one child: Replace with child
3. Node with two children: Replace with predecessor or successor

Remove: 25



Remove 82:



Remove 47:

can choose to replace with either predecessor or successor