Assignment 03: Self Balancing Tree

BROCK DAVIS

Goal

- Make a generic self-balancing tree
  - Generic: Can be used with any Comparable data type
  - Self-Balancing: Rotates node connections so search takes O(n*log(n))

Approach Overview

- AVL Tree
- Create a Node class
  - Has generic type
  - Keeps trach of left, right, and parent Node
  - Recursive functions for add, remove, etc.
    - Self-balances up through call stack

Add

```
if newVal < thisVal:
   if left is null:
      left = newVal
      left.parent = this
   else:
      left.add(newVal)
//Same with right//
updateDepth
rebalance
```

Remove

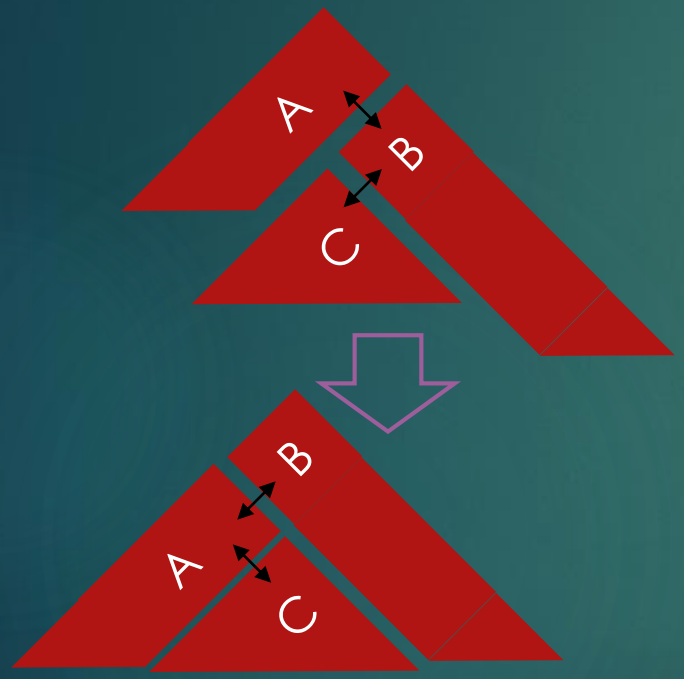
```
if removeVal == this.value:
    if left isNull && right isNull:
        parent.remove(this)
    else if right not isNull
         this.value = right.min
        right.remove(this)
    else if left not isNull
        this.value = left.min
        left.remove(this)
else if removeVal < this.value
    left.remove(removeVal)
//Same with right//
updateDepth()
parent.balance(this)
```

Rebalance

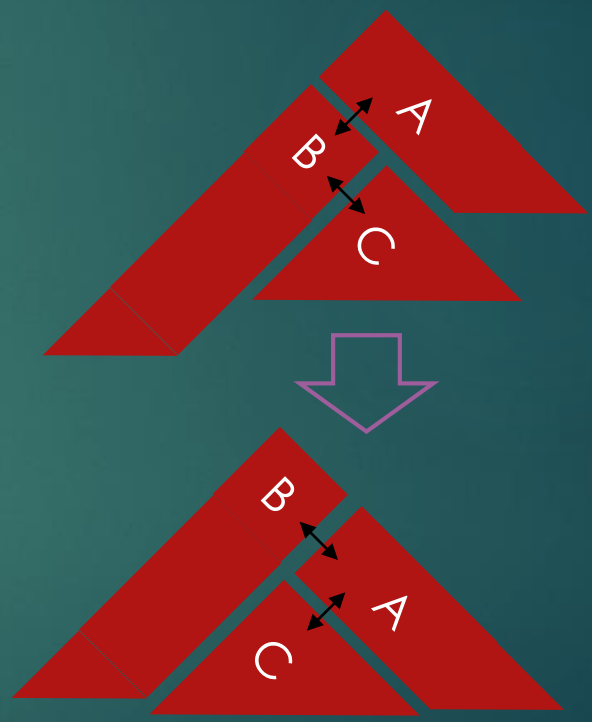
```
if left.depth - right.depth > 1:
    if left.right.depth > left.left.depth:
        left.rotateLeft()
    this.rotateRight()
if left.depth - right.depth < -1:
    if right.left.depth > right.right.depth:
        right.rotateRight()
    this.rotateLeft()
```

Rotation

- Left Rotation



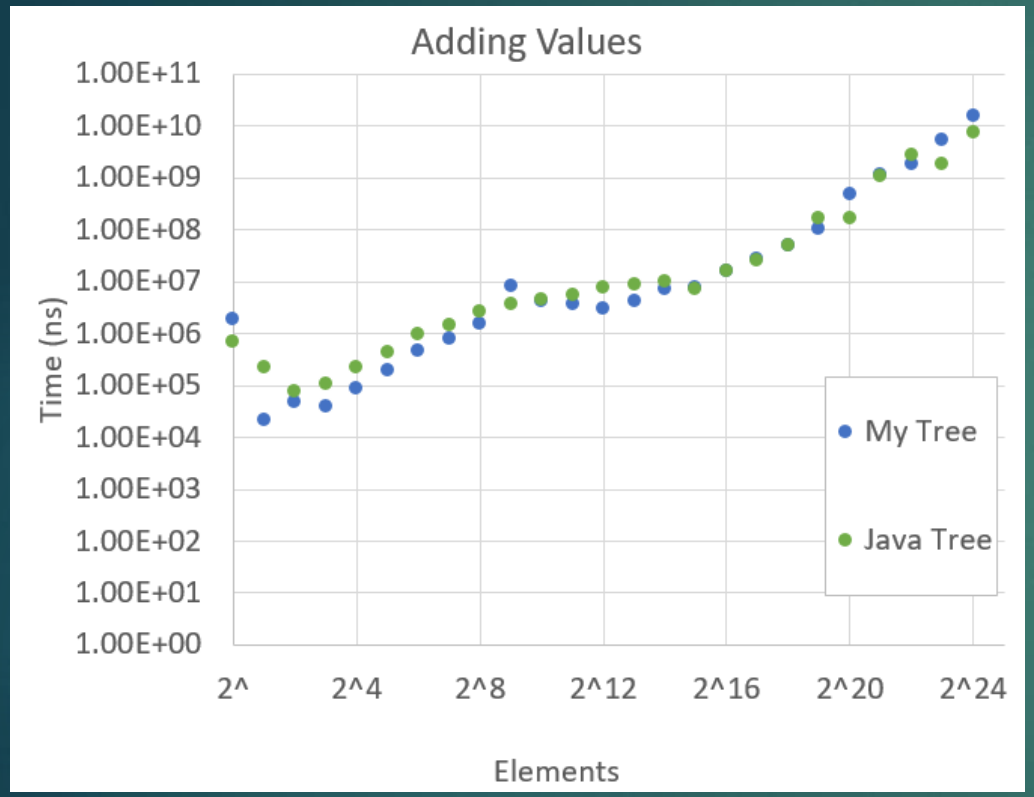
- Right Rotation

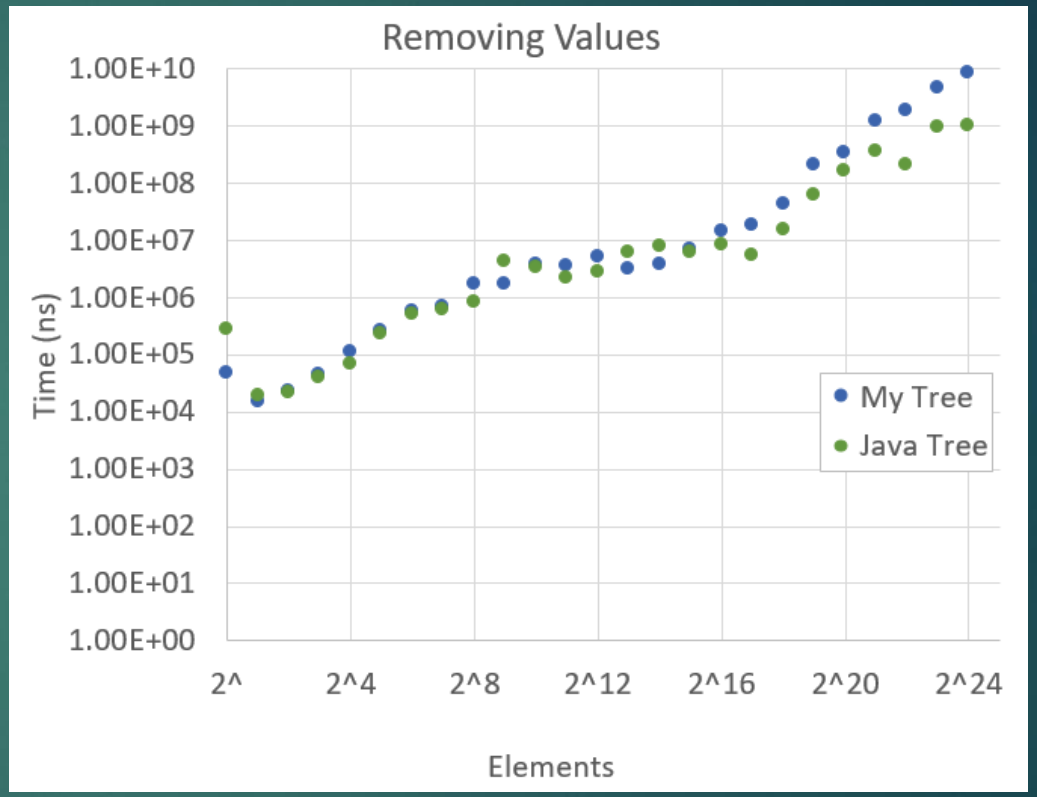


Results

- The Implemented AVL Tree was compared to the Java TreeSet for time of addition and removal of consecutive integers
- Depth of tree was always $\log_2(n)$ where n is # of elements

Results


Adding Values
1.00E+11
1.00E+10
1.00E+09
1.00E+08
1.00E+07
1.00E+06
1.00E+05
1.00E+04
1.00E+03
1.00E+02
1.00E+01
1.00E+00
Time (ns)
2^
2^4
2^8
2^12
2^16
2^20
2^24
Elements
My Tree
Java Tree


Removing Values
1.00E+10
1.00E+09
1.00E+08
1.00E+07
1.00E+06
1.00E+05
1.00E+04
1.00E+03
1.00E+02
1.00E+01
1.00E+00
Time (ns)
2^
2^4
2^8
2^12
2^16
2^20
2^24
Elements
My Tree
Java Tree