COMP 352: Data Structures and Algorithms

Assignment 3

Summer 2020, sections AA

June 12, 2020

Shadi Jiha #40131284

Question 1:

1. Complexity:

public int depth(Node<T> root) {  
 if (root == null)  
 return 0;  
 else {  
  
 int left\_depth = depth(root.left());  
 int right\_depth = depth(root.right());  
  
 if (left\_depth > right\_depth)  
 return left\_depth + 1;  
 else  
 return right\_depth + 1;  
 }  
}

public void traversePreOrder(Consumer<Node<T>> operation) {  
 traversePreOrder(root, operation);  
}  
  
private void traversePreOrder(Node<T> root, Consumer<Node<T>> operation) {  
  
 if (root == null)  
 return;  
  
 operation.accept(root);  
 traversePreOrder(root.left(), operation);  
 traversePreOrder(root.right(), operation);  
}

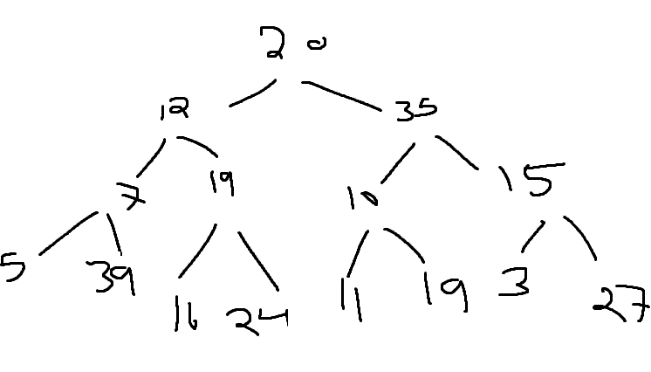
Then inside main we can call:

List<Integer> depth\_of\_nodes = new ArrayList<>();  
tree.traverseInOrder(e -> depth\_of\_nodes.add(tree.depth(e)));

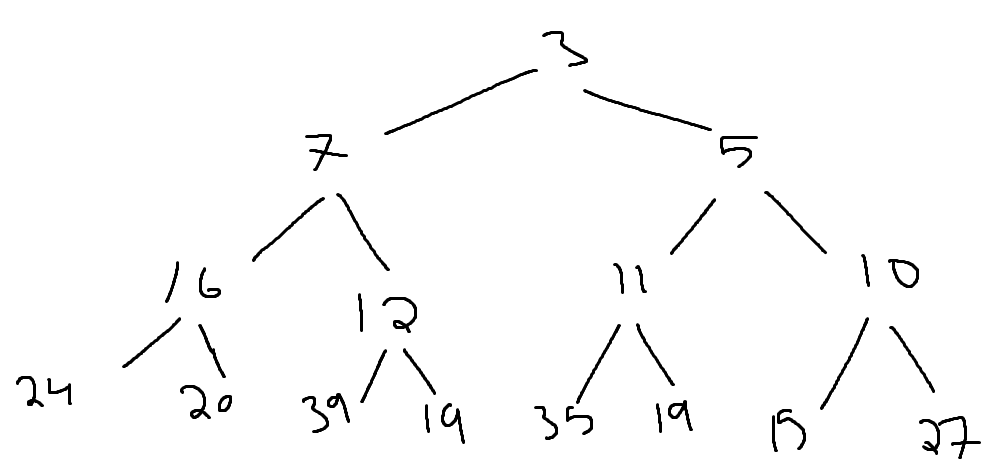
1. Complexity:

public static int Count\_Full\_Nodes(BinarySearchTree<?> tree) {  
  
 AtomicInteger count = new AtomicInteger();  
 tree.traverseInOrder(e -> {  
 if (e.hasLeft() && e.hasRight())  
 count.getAndIncrement();  
 });  
  
 return count.get();  
}

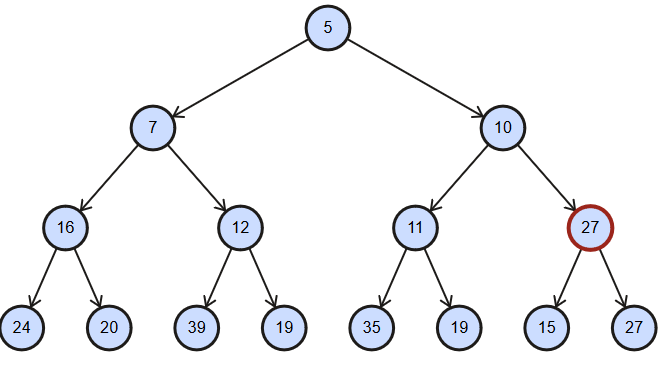
**Question 2:**

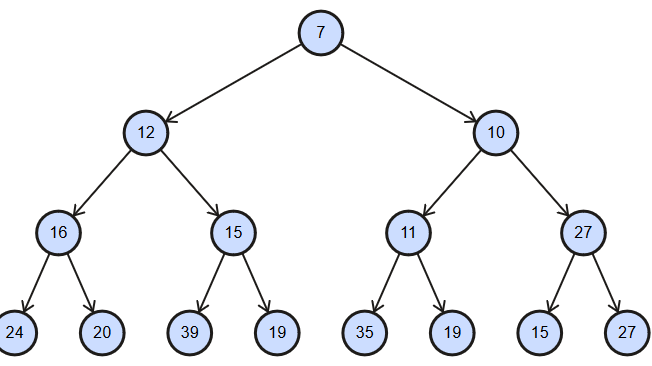
1. Initial Tree

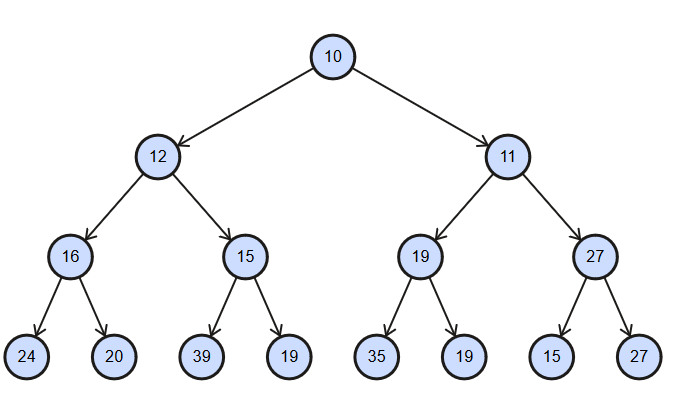
After Mini-Heap tree

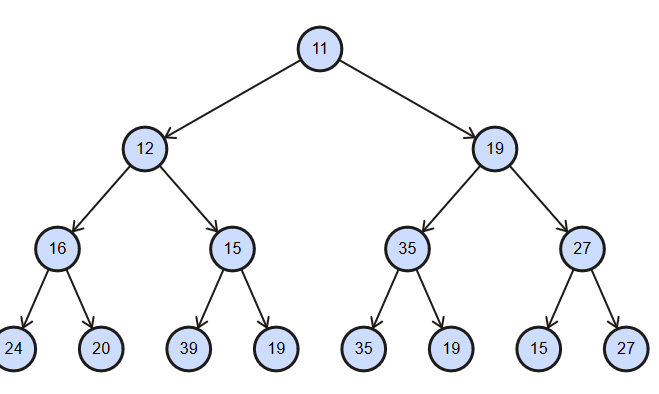


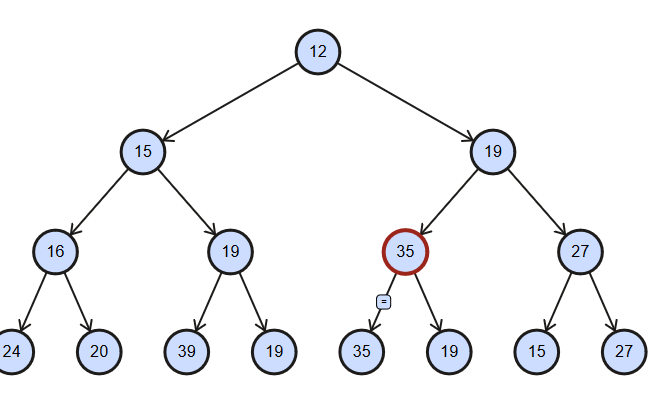
Here I used a tree drawer after each step:

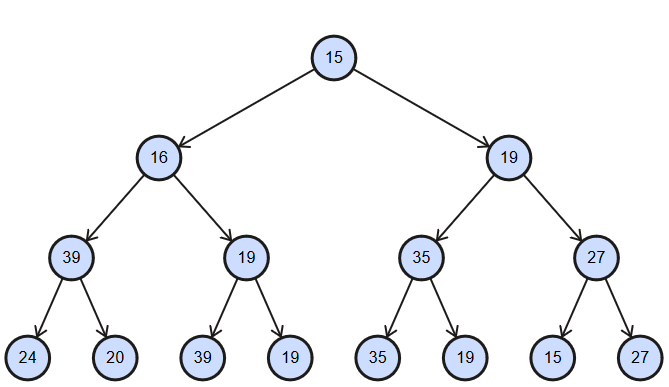












1. Since it will take too much space, I will show them as arrays

[ 20, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

[ 12, 20, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

[ 12, 20, 35, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

[ 12, 19, 35, 20, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

[ 7, 12, 35, 20, 19, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

[ 7, 12, 10, 20, 19, 35, 0, 0, 0, 0, 0, 0, 0, 0, 0]

[ 7, 12, 10, 20, 19, 35, 15, 0, 0, 0, 0, 0, 0, 0, 0]

[ 7, 12, 10, 20, 19, 35, 15, 24, 0, 0, 0, 0, 0, 0, 0]

[ 7, 12, 10, 16, 19, 35, 15, 24, 20, 0, 0, 0, 0, 0, 0]

[ 7, 12, 10, 16, 19, 35, 15, 24, 20, 39, 0, 0, 0, 0, 0]

[ 5, 7, 10, 16, 12, 35, 15, 24, 20, 39, 19, 0, 0, 0, 0]

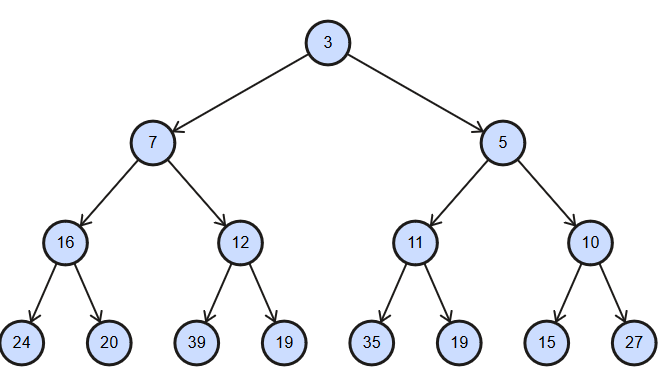
[ 5, 7, 10, 16, 12, 19, 15, 24, 20, 39, 19, 35, 0, 0, 0]

[ 5, 7, 10, 16, 12, 11, 15, 24, 20, 39, 19, 35, 19, 0, 0]

[ 3, 7, 5, 16, 12, 11, 10, 24, 20, 39, 19, 35, 19, 15, 0]

[ 3, 7, 5, 16, 12, 11, 10, 24, 20, 39, 19, 35, 19, 15, 27]

Here’s the final tree



**Question 3:**

1. [195=0, 91=0], [16=0, 94=0, 81=0], [147=0], [265=0], [32=0, 162=0], [189=0, 202=0], [21=0], [48=0], [75=0], [180=0, 37=0], [207=0, 77=0]
2. 7 Collisions

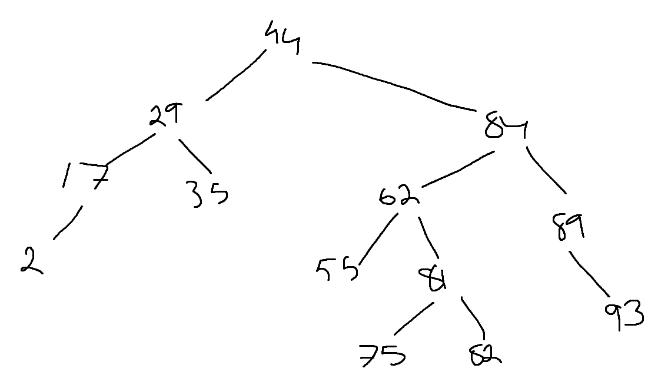
**Question 4:**

No, it doesn’t. The collision number went from 7 to 8.

**Question 5:**

1. [39, 29, 42, 31, 25, 25, 48, 35, 12, 35, 29, 31, 18]
2. 5
3. 11
4. Load factor =

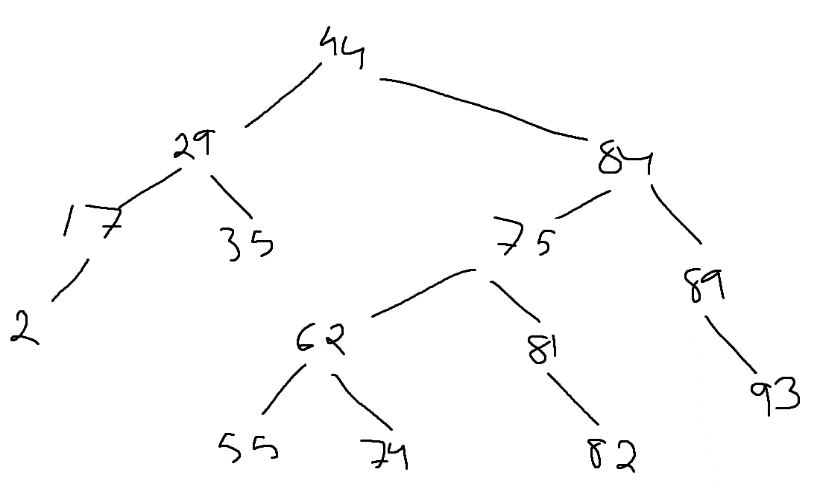
**Question 6:**

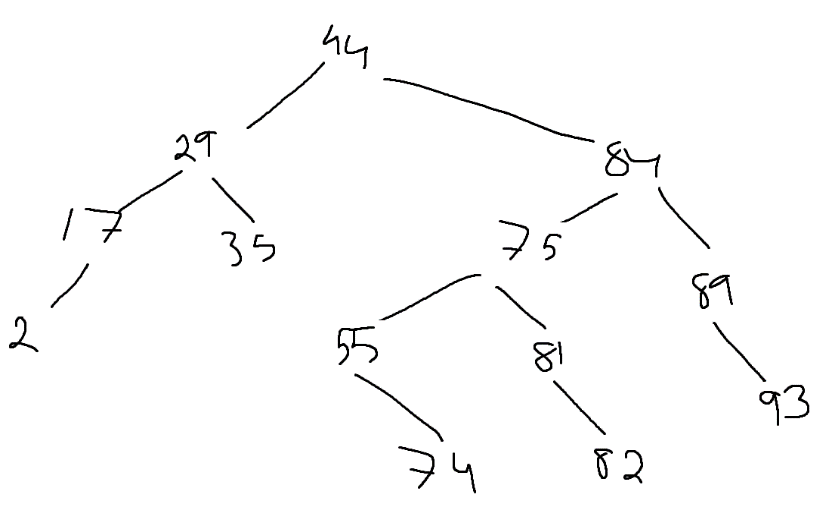
1. Here’s the correct balanced tree:

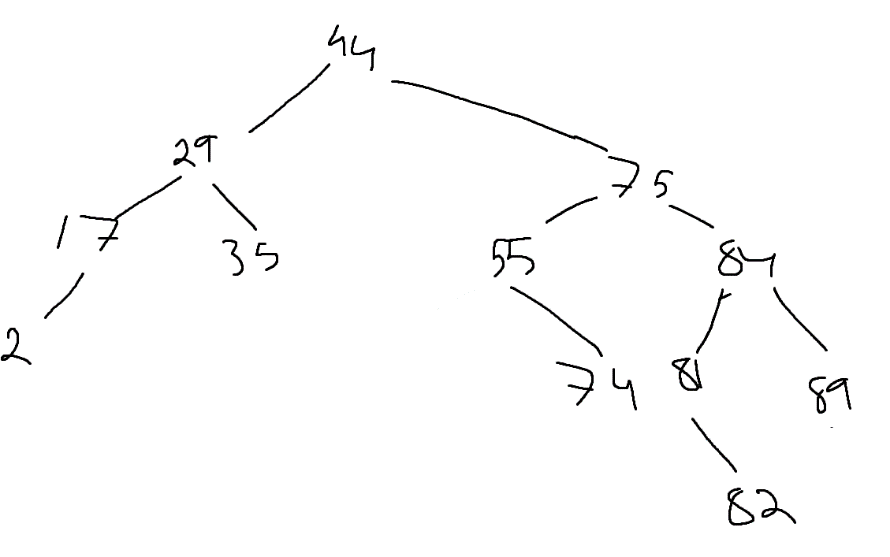
Wrong rotations, 82 has more than 2 height difference with 81

2 is inserted in the wrong place

1. :







**PART2:**

Part 2 was submitted separately in moodle