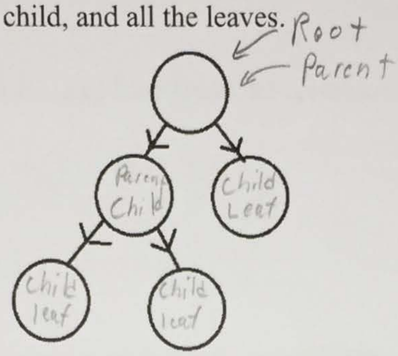
EECS 233 SI Session 6
Leader: Bertram Su
September 26, 2019

Objectives:
Upon completion of this SI session, participants will be able to:
1. Recognize binary tree terminology
2. Determine what pre, in, and post-fix traversal of a binary tree would do

Joke: Why do computer scientists draw trees upside down?

Foundations:
1. Label the root, a parent, a child, and all the leaves.



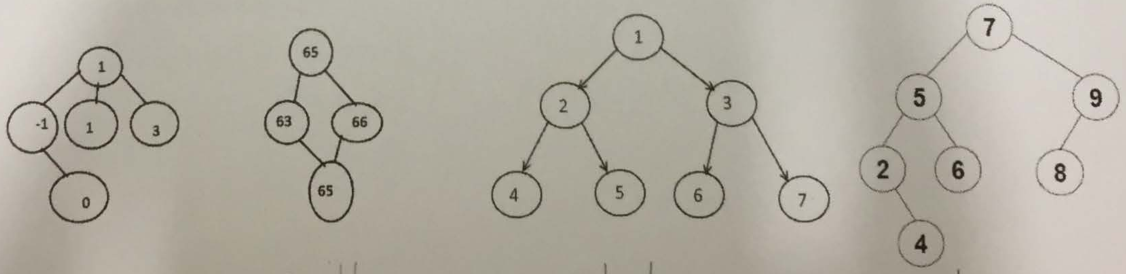
2. What things should each node in a binary tree have? Try to list three of them.

key*, left, right, data

3. Fill in the instructions for pre, in, and post-fix traversal
*Hint: We always travel left to right, but the prefix determines when we print the node

Pre : Node, left, right
in : Left, Node, Right
post : Left, right, node

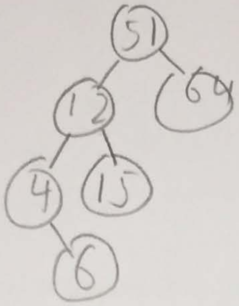
4. Are the following trees most accurately binary search trees, binary trees, or neither?



Neither neither bt bst

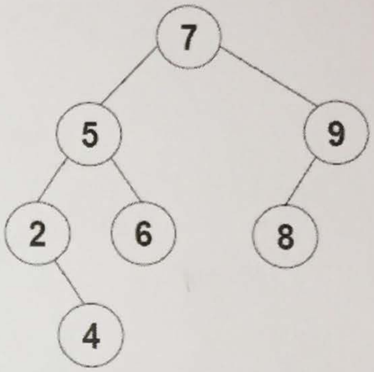
5. What must every class that wants to be a binary tree have? It acts like the head of a linked list, top of a stack, or front of a queue. root

6. Draw the binary search tree after the following operations: Add 51, 64, 12, 4, 6, 15



Exercises:

7. Write the Pre-Order, In-Order, and Post-Order traversal for the tree in-order: Left, node, right



pre: 7, 5, 2, 4, 6, 9, 8

in: 2, 4, 5, 6, 7, 8, 9

Post: 4, 2, 6, 5, 8, 9, 7

8. Finish the pre-order print method.*These methods will be called on root

```
public void preorderPrint( ){
    System.out.println( data );
    if (left != null )
      left.preorderPrint( );
    if ( right != null )
      right.preorderPrint( );
}
```

9. Finish the in-order print method. *Hint: do what you did for preorder but move a statement

```
public void inorderPrint( ){
    if (left != null)
        left.inorderPrint();
    S.o.p(data);
    if (right != null)
        right.inorderPrint();
}
```

Summary

10. Finish the postorder print method

```
public void postorderPrint( ){
    if (left != null)
        left.pop();
    if(right != null);
        right.post print();
    sy.o.p(data
}
```

11. What's the big O of searching for something in a binary search tree? Give some input that would result in such a tree.

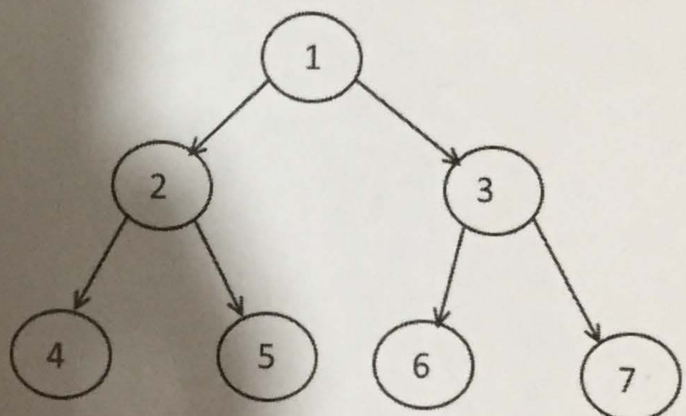
O(n) 1, 2, 3, 4, 5 or < = >

12. Write the Pre-Order, In-Order, and Post-Order traversal for the tree

re: 1, 2, 4, 5, 3, 6, 7

n: 4, 2, 5, 1, 6, 3, 7

ost: 4, 5, 2, 6, 7, 3, 1



Upcoming Events and Suggestions for Further Study:

Events:

- Next SI session is Sunday from 1:00 to 2:30 at Sears 325

Further Study:

- bigocheatsheet.com
    - A great graph that visualizes the big o complexity chart. It also has the big O time of data structures and algorithms that we will cover in the future.
- https://www.geeksforgeeks.org/tree-traversals-inorder-preorder-and-postorder/
    - An alternative implementation of traversals

EECS 233 SI Session 7
Leader: Bertram Su
September 29, 2019

Objectives:

Upon completion of this SI session, participants will be able to:

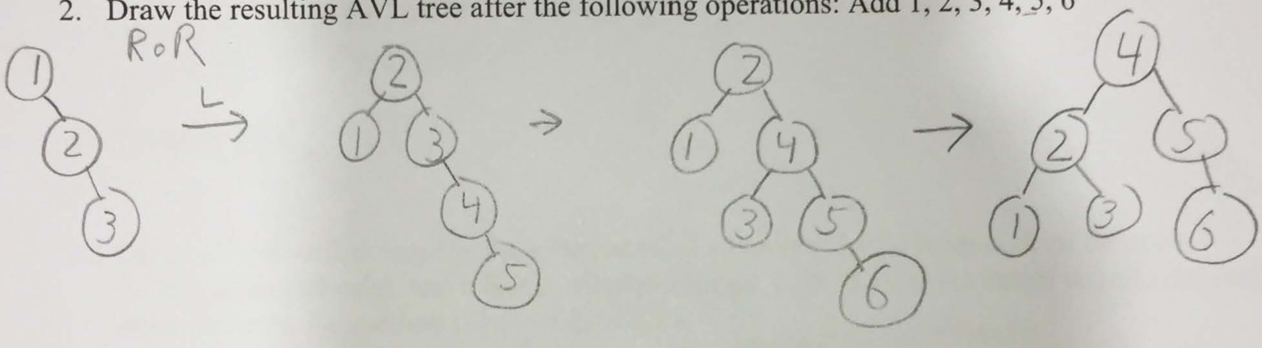
1. Recognize what remove would do to a binary search tree
2. Implement simple add and rotation in AVL trees

Foundations:

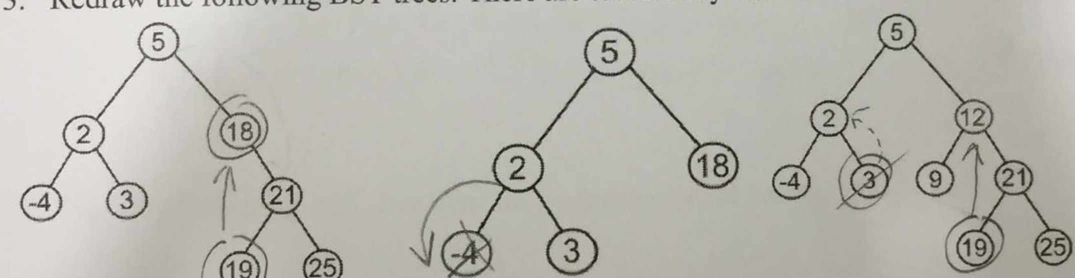
1. What's the difference between binary trees, binary <u>search</u> trees, and AVL trees? What additional pointer does an AVL tree have vs a BST.

   BST have order (left smaller, right ≥ the parent)
   AVL are BST that automatically balance

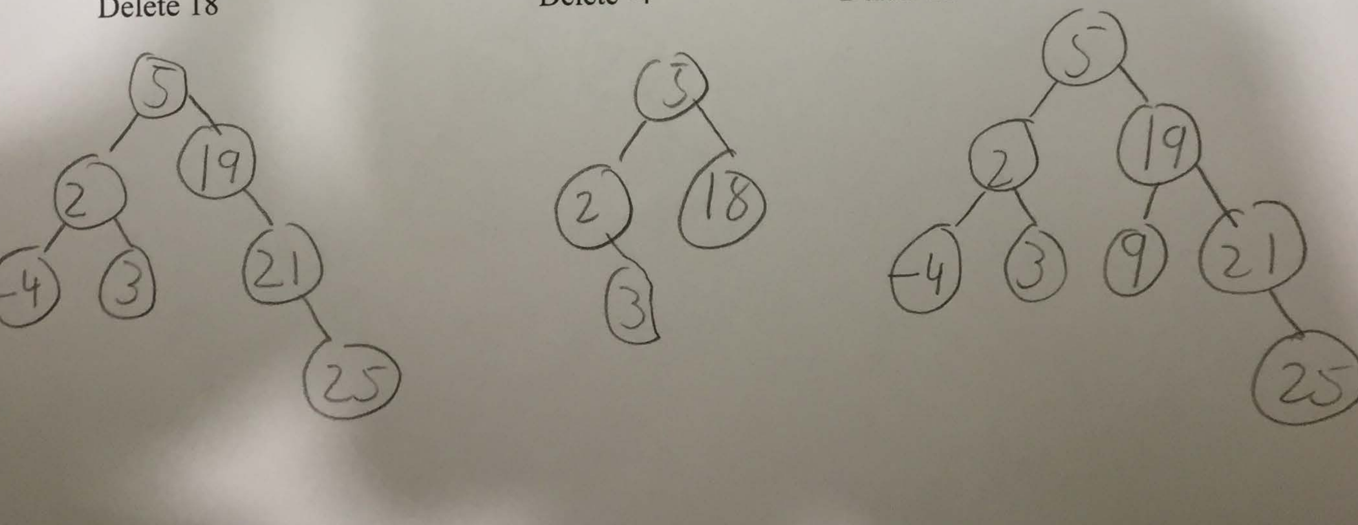
2. Draw the resulting AVL tree after the following operations: Add 1, 2, 3, 4, 5, 6



3. Redraw the following BST trees. There are technically two correct answers for each



Delete 18 | Delete -4 | Delete 12



Exercises:

4. In words, describe the algorithm to delete a node from a BST.

if the target node has no children

ex: change targetNode's parent to null or if root change root to null

if the target node has a right child

get the the leftmost from the right subtree of target Node
Replacct target's data and key, delete letmost

if the target has a left child but no right

target's parent points to target's left

5. Write a method to get the leftmost Node of a Node (this will be in our node class)

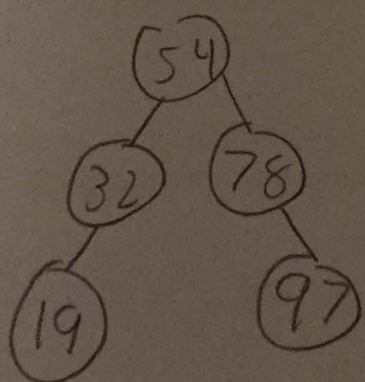
```
public Node getLeftMost(){
Node ptr = this;
  while (ptr.left != null) {
      ptr = ptr.left;
  }
  return ptr;
}
```

6. Given a Node called targetNode, a Node called leftmost (The leftmost node of the right subtree of targetNode), and a Node called pLeftmost write the code to delete targetNode. Just write the code. No method is header is needed

parent of leftmost

```
targetNode.data = leftmost.data
targetNode.key = leftmost.key
pLeftmost.left = null or leftmost.left
```

7. Draw the following AVL tree after the operations: Add 54, 32, 78, 97, 19



8. !Write the code for a ~~right~~ left rotation on node y. Assume y is a right child and not root

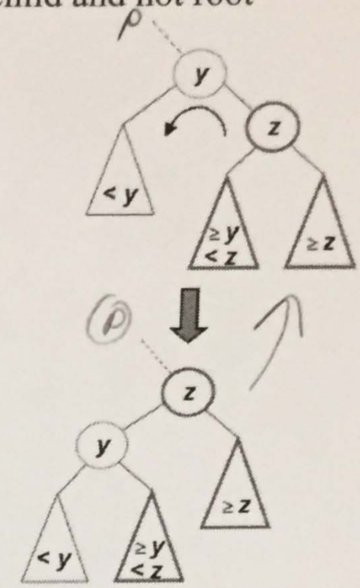
```
private void LeftRotation(Node y){
        Node z = y.right
        Node p = y.parent

        p.right = z

        y.right = z.left
        y.parent = z

        z.left = y
        z.parent = p
        y.right.parent = y

}
```



Summary

9. Write the code for a ~~left~~ righ rotation on node y. Assume y is a right child and not root

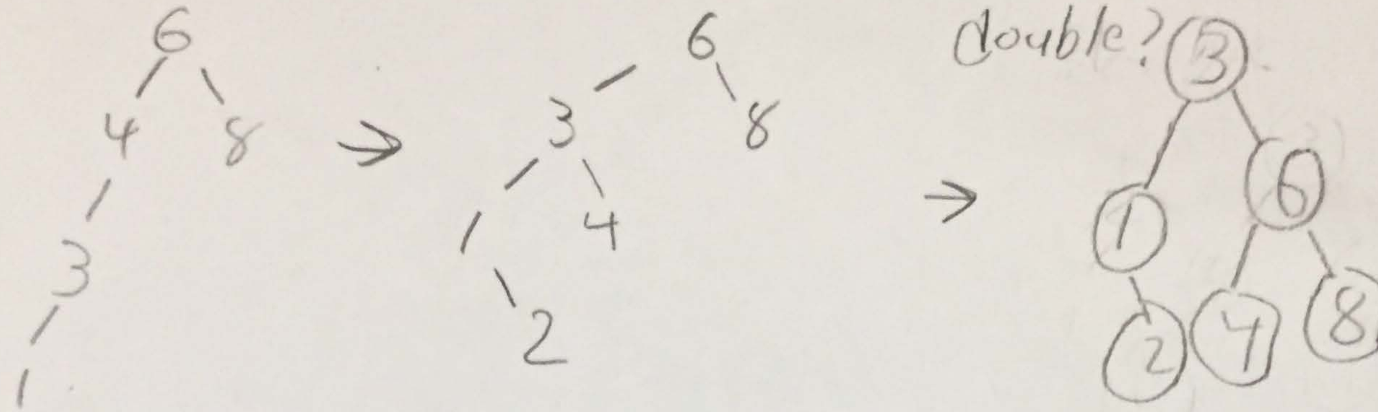
```
private void rightRotation(Node y){
        Node z = y.left
        Node p = y.parent

        p.right = z
        y.left = z.right
        y.parent = z
        z.right = y
        z.parent = p
        y.left.parent = y
}
```

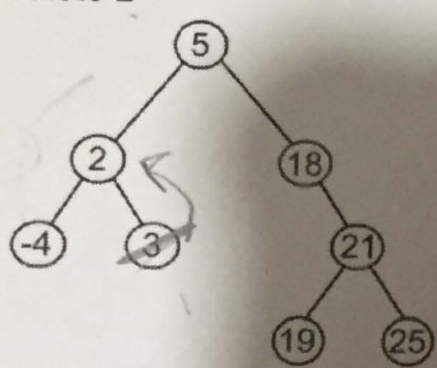
10. President Snyder wants to store an unknown amount of sorted Case IDs and asks you what data structure to use. What data structure would you recommend to her? Why not just use an array?

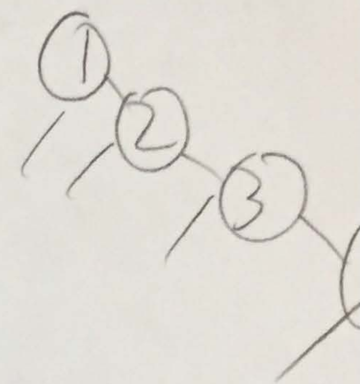
AVL because unknown amount and no need to scramble

11. Draw the tree after the following operations for an AVL tree
Add 6, 8, 4, 3, 1, 2



12. Draw the tree after the following operations for a BST
Delete 2





Upcoming Events and Suggestions for Further Study:
Events:

- Next SI session is Thursday from 6:00 to 7:30 at Sears 336
- It will most likely cover the removal of nodes from AVL trees (double rotations)

Further Study:

- bigocheatsheet.com
  - A great graph that visualizes the big o complexity chart. It also has the big O time of data structures and algorithms that we will cover in the future.
- https://www.cs.usfca.edu/~galles/visualization/BST.html
  - This website visually builds adding and removing integers into a binary search tree. WARNING: it uses the rightmost of the left subtree
- https://www.cs.usfca.edu/~galles/visualization/AVLtree.html