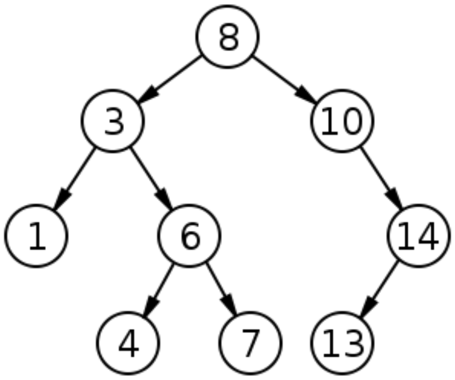
CSE 373 Section Spring 2014 April 17, 2014

# BST's and AVL Trees



## Show the Traversals

Pre-Order: 8, 3, 1, 6, 4, 7, 10, 14, 13

In-Order: 1, 3, 4, 6, 7, 8, 10, 13, 14

Post-Order: 1, 4, 7, 6, 3, 13, 14, 10, 8

### Definition of BST

- Collection of nodes that hold data
- Each node in the tree is connected to another
- A node can have no more than 2 “children”
- The left subtree of any given node will only contain data values less than the value of that node
- The right subtree of any given node will only contain data values greater than the value of that node

### Description of BST Node

- Field for holding data
- Field for accessing right subtree
- Field for accessing left subtree

### Definition of AVL Tree

- A binary tree that is self-balancing.

### Description of AVL Tree Node

- Field for holding data
- Field for accessing right subtree
- Field for accessing left subtree
- Field for keeping track of height

## Runtime Analysis:

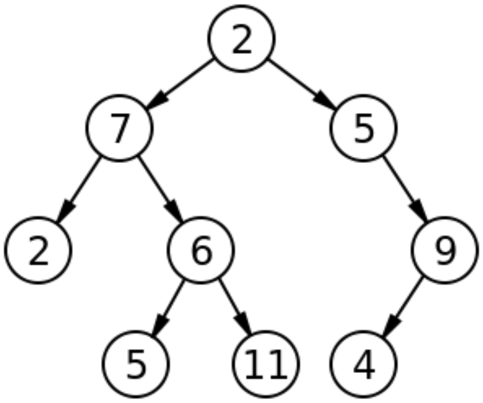
| | BST: | AVL Tree: |
|---|---|---|
| find(): | O(N) | O(log N) |
| insert(): | O(N) | O(log N) |
| delete() | O(N) | O(log N) |
| buildTree() | O(N^2) | O(N log N) |

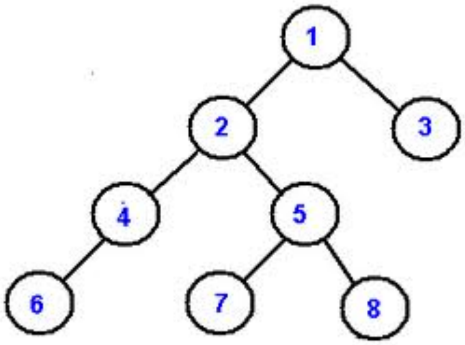
## AVL Operations:

- Single Rotation
- Double Rotation

## Practice Problems:

1. Is it a BST? Is it an AVL Tree? (If not, circle nodes that violate the rules of each)





BST: NO

AVL: NO

BST: NO

AVL: NO

2. Adding values to a BST in a certain order, what does the resulting tree look like? How about AVL?

2, 6, 8, 1, 9, 13, 7

```
        6
       / \
    2      9
   /      / \
  1      8   13
        /
       7
```