CSC508 Data Structures

Topic 9 : Tree 2

Binary Search Tree

Compiled & edited by: Zahid Zainal

Recap

- Tree Definition
- Tree Terminologies
- Binary Tree
- Binary Tree Representation
- Binary Tree Traversal

Topic Structure

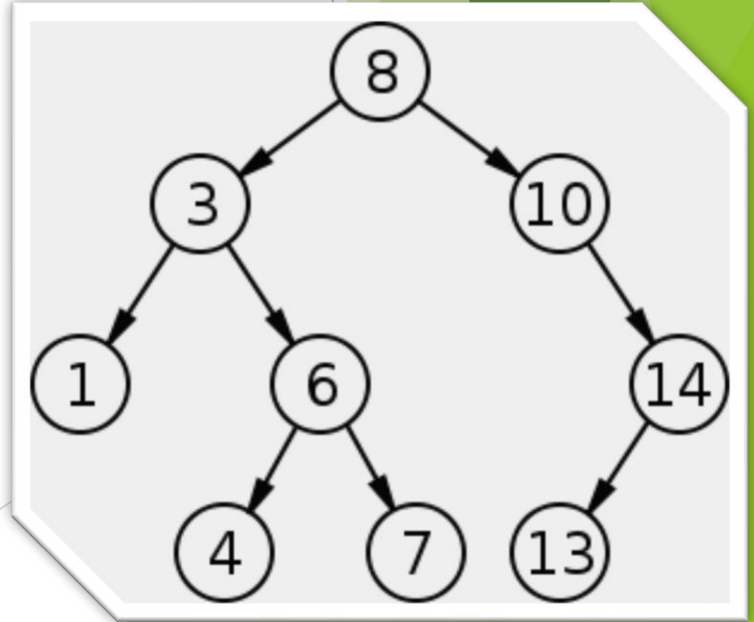
- Binary Search Tree (BST)
- BST Operation
- AVL Tree

Learning Outcomes

- At the end of this lesson, students should be able to:
  - Describes the properties of a binary search tree
  - Explain main operations on BST : Searching, Insertion, & Deletion
  - Describe AVL Tree and its algorithm

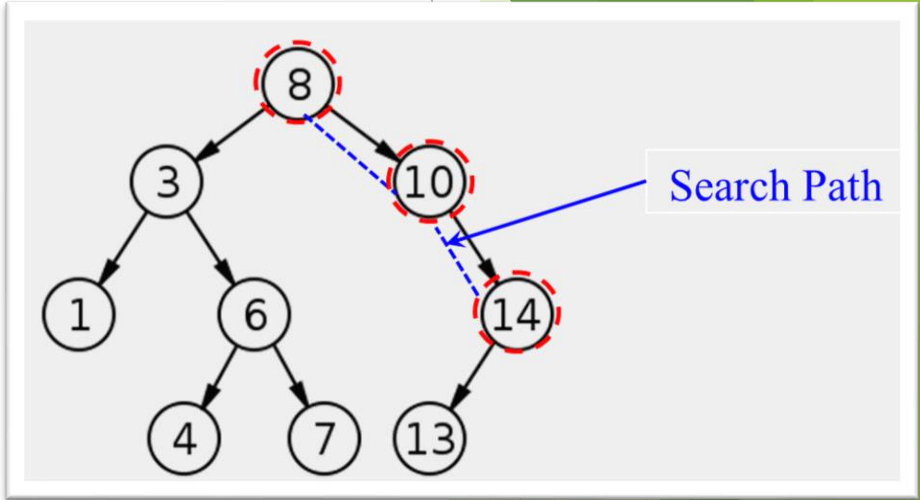
Binary Search Tree

- A type of binary trees, where nodes' data has a key element, and the tree has the following properties:
  - The left subtree of any node contains only nodes with keys less than the node's key
  - The right subtree of any node contains only nodes with keys greater than the node's key
  - There must be no duplicate keys among the nodes



Searching in BST

- A faster searching algorithm can be done using BST.
- Example : Find node 14
  - BST : 8 – 10 – 14   **3 nodes visited**
  - Pre-Order : 8 – 3 – 1 – 6 – 4 – 7 – 10 – 14
  - In-Order : 1 – 3 – 4 – 6 – 7 – 8 – 10 – 13 – 14
  - Post-Order : 1 – 4 – 7 – 6 – 3 – 13 – 14
- If the tree is complete (or near complete) and contains n elements, the worst-case scenario for time spent to find a node is $Log_2(n)$



Main Operations on BST

- Searching
- Adding a new node
- Removing as node

BST - Searching

- A recursive function
  - Start from the root node.
  - Check if the node value equal to key
  - If yes, return. If you no, check whether the node value is less than key.
  - If yes, traverse to the right child. If no, traverse to the left child.
  - Repeat until the value is found or reach the end of the tree branch

```
public Node search(Node root, int key){
    if (root==null || root.key==key)
        return root;

    if (root.key < key)
        return search(root.right, key);
    else
        return search(root.left, key);
}
```

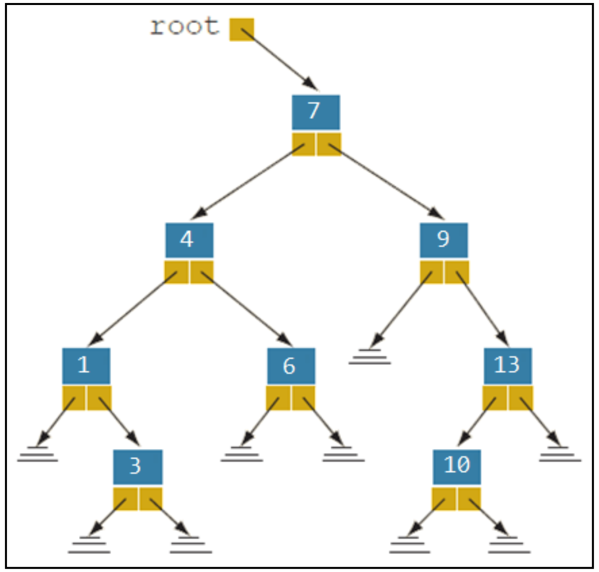
Minimum and Maximum Value

- Maximum-key node is the rightmost node in the tree
- Minimum-key node is the leftmost one

```
public Node max(){
   if (root==null)
      return null;

   while (root.right != null)
      root = root.right;

   return root;
}
```



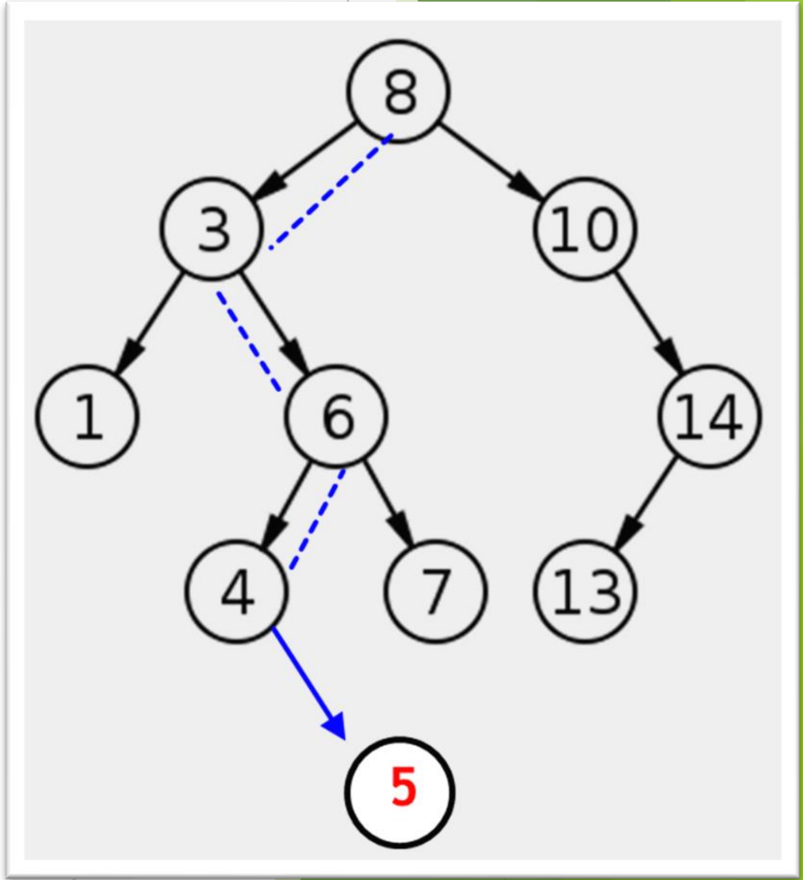

```
public Node min(){
   if (root==null)
      return null;

   while (root.left != null)
      root = root.left;

   return root;
}
```

Inserting a New Node into a BST

- New node will always be added as a leaf node.
  - Apply a search to find the value of the node
  - If found, then no insertion takes place (no duplicate keys)
  - Otherwise, insert the node at the end of the search path by linking it to its parent node as a left or right child
- The shape of the new BST is depending on the order of nodes insert.

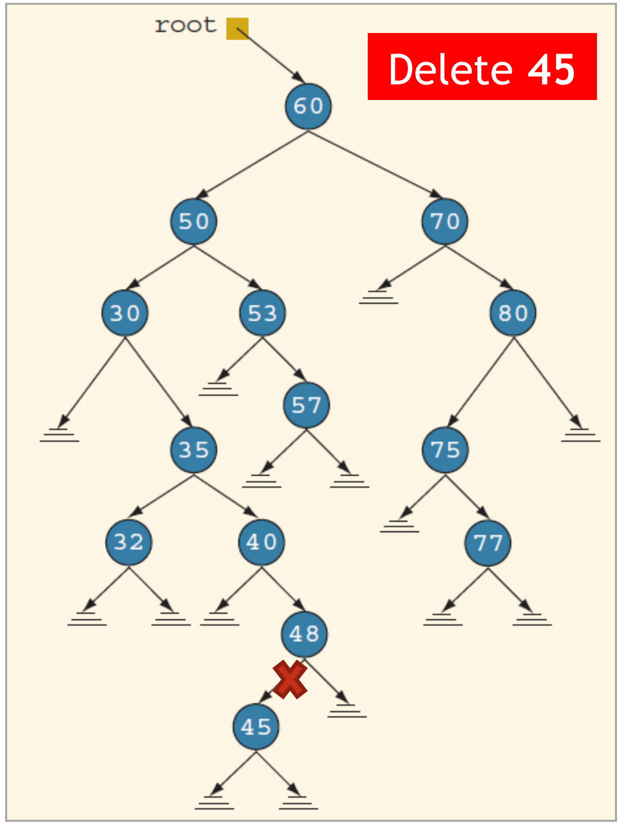


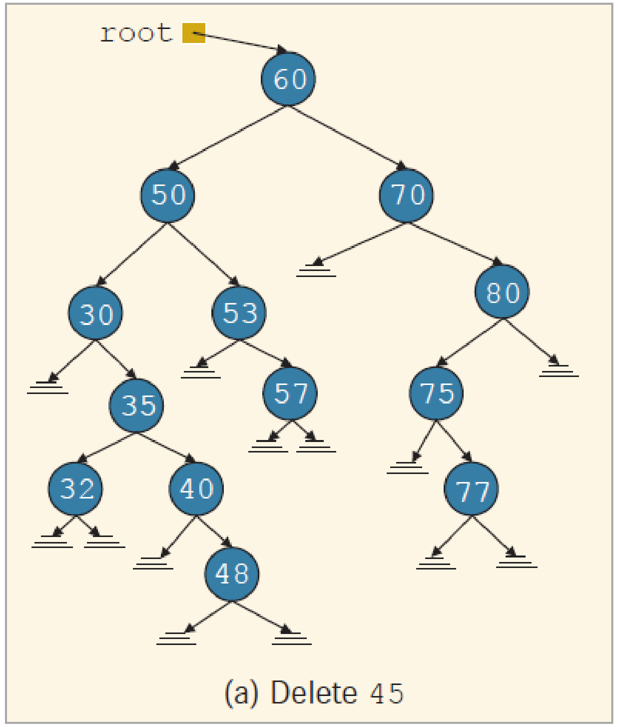
Deleting a Node from a BST

- After deleting the desired item, the resulting tree must remain a binary search tree structure
- Therefore, delete operation must handle these four cases:
  - The node to be deleted is a leaf
  - The node to be deleted has no left subtree
  - The node to be deleted has no right subtree
  - The node to be deleted has nonempty left and right subtrees

Delete cases

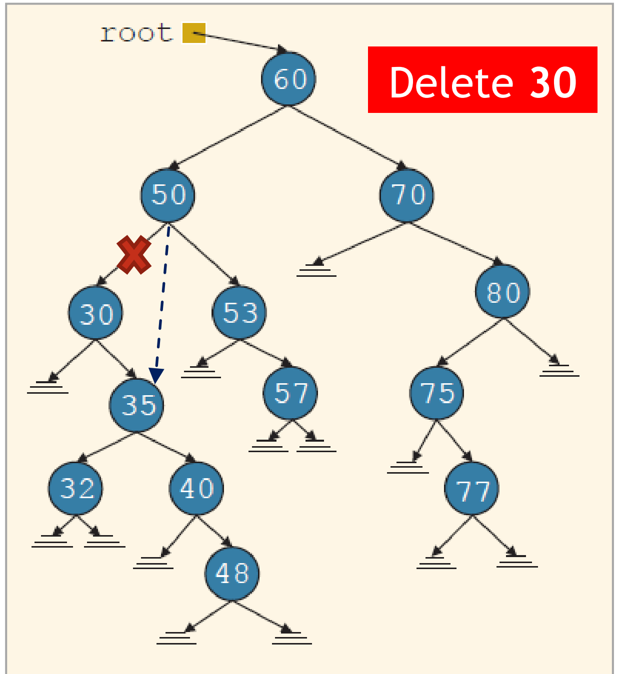
- Case 1: The node to be deleted is a leaf



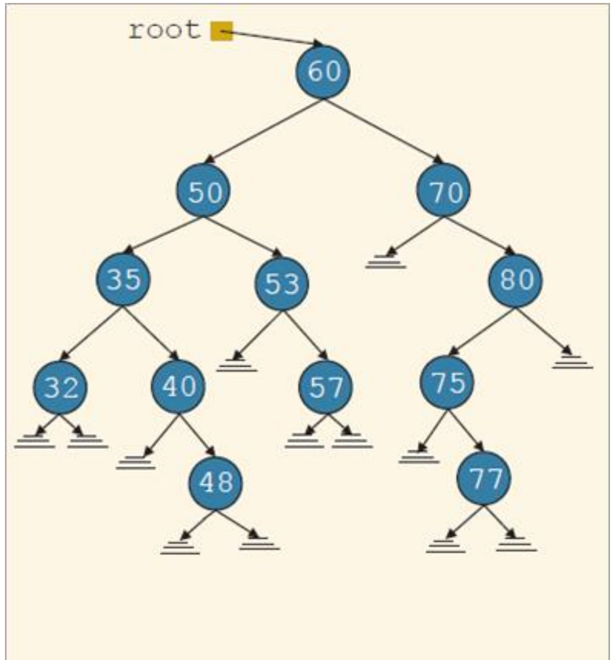


Delete cases

- Case 2: The node to be deleted has no left subtree



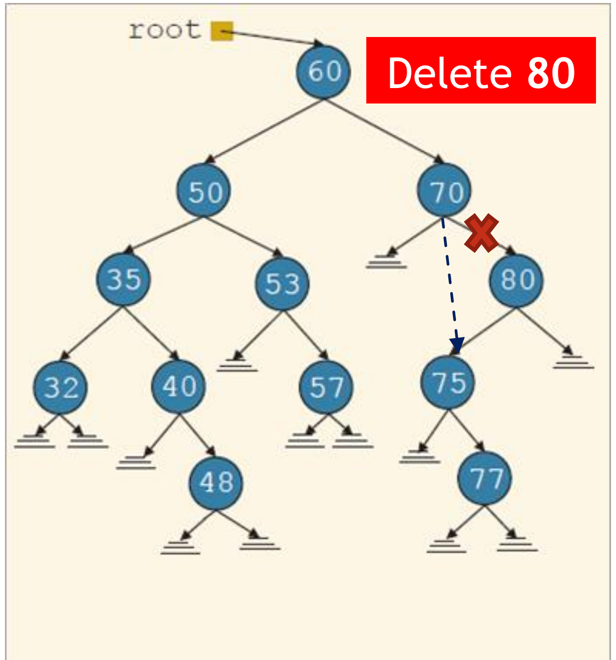
(a) Delete 45



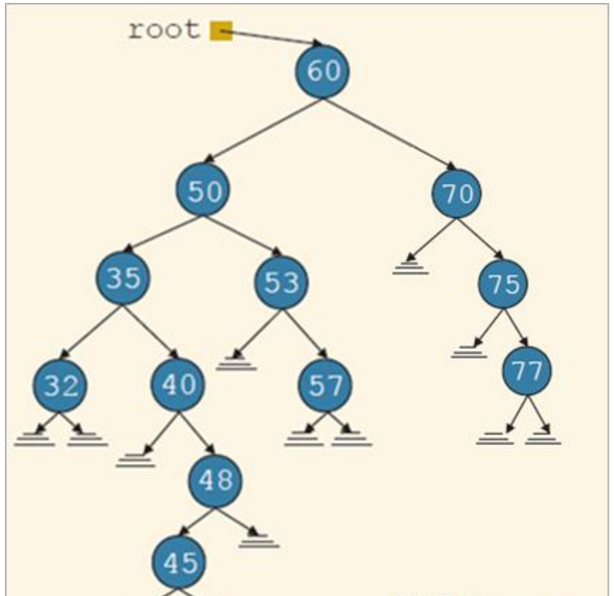
(b) Delete 30

Delete cases

- Case 3: The node to be deleted has no right subtree



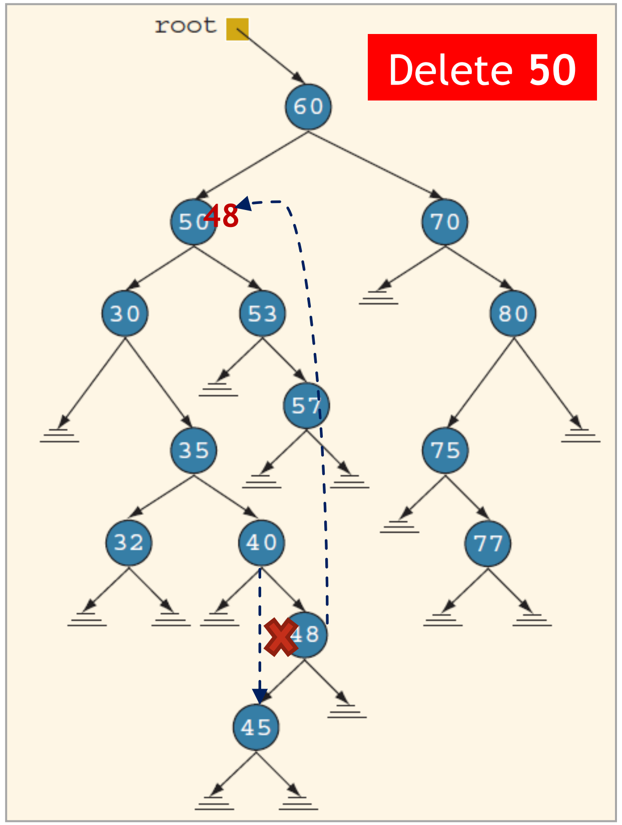
(b) Delete 30

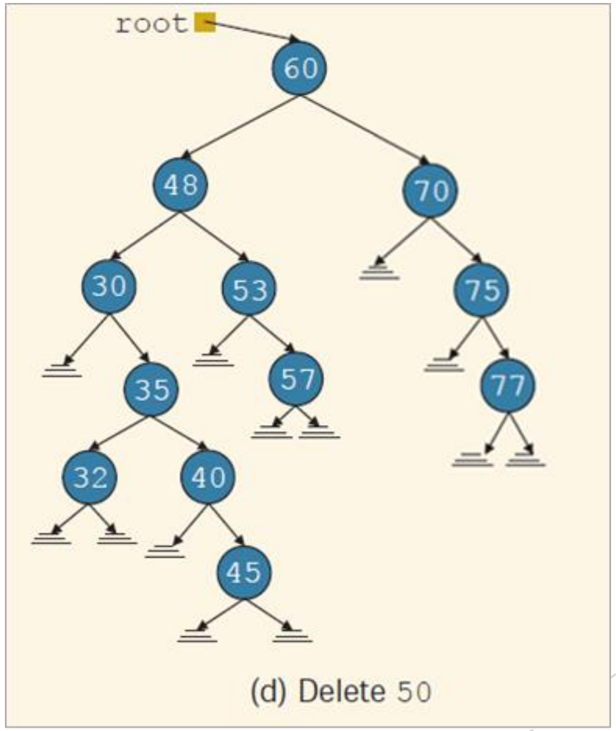


(c) Delete 80

Delete cases

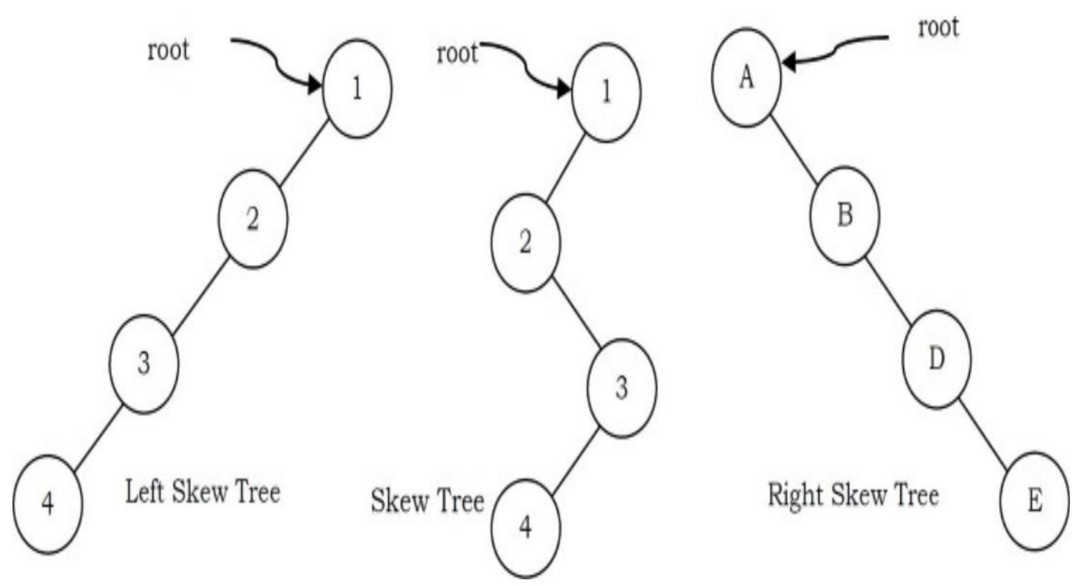
- Case 4: node has left and right subtrees
  - Replace with max of left subtree (or min of right subtree)





Balance BST

- The addition and deletion of node in the BST will change its structure.
- An unbalanced BST, e.g. skewed tree, may have **O(n)** complexity in the worst-case scenario.
- A balanced BST is required to achieve **O(log n)** complexity.
- Adelson-Velsky Landis (AVL) Tree is a balanced BST.



AVL Tree

- A self-balancing BST, where the difference between heights of left and right subtrees cannot be more than one for all nodes.
- The balance factor of x, written bf(x), is defined as

  $$bf(x) = x_r - x_l$$

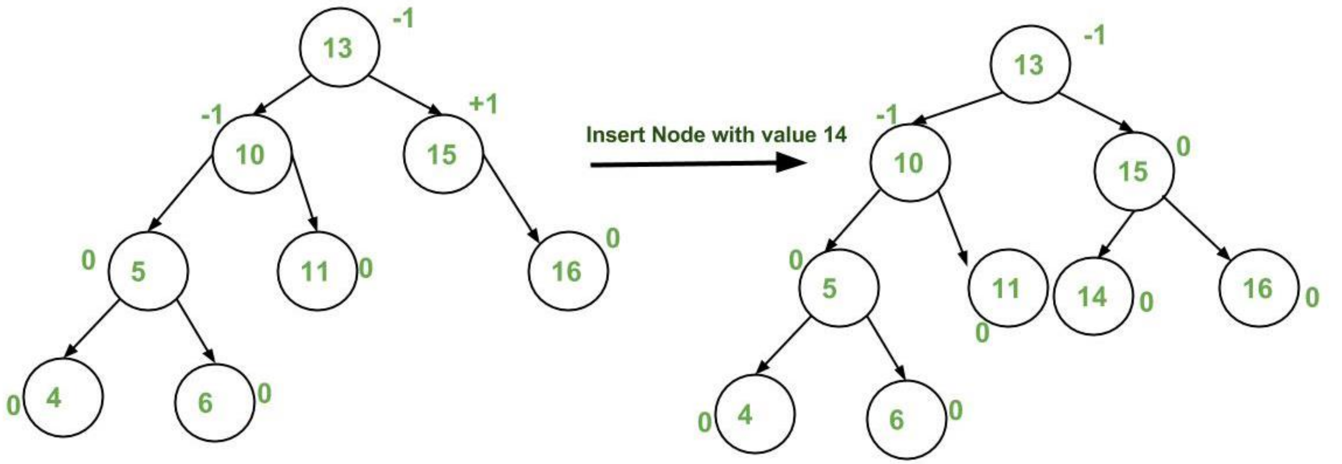
- Let x be a node in the AVL tree T. Then:
  - If x is left high, then bf(x) = -1
  - If x is equal high, then bf(x) = 0
  - If x is right high, then bf(x) = 1
- Node x violates the balance criteria if $|x_r - x_l| > 1$, that is, the height of the left and right subtrees of x differ by more than 1.

AVL Tree Rotation

- The balance factor of nodes is calculated after every insertion and deletion.
- Balancing an AVL tree is done through rotation at any given node, x.
  - Left rotation: certain nodes from the right subtree of x move to its left subtree; the root of the right subtree of x becomes the new root of the reconstructed subtree.
  - Right rotation at x: certain nodes from the left subtree of x move to its right subtree; the root of the left subtree of x becomes the new root of the reconstructed subtree

AVL Tree Examples

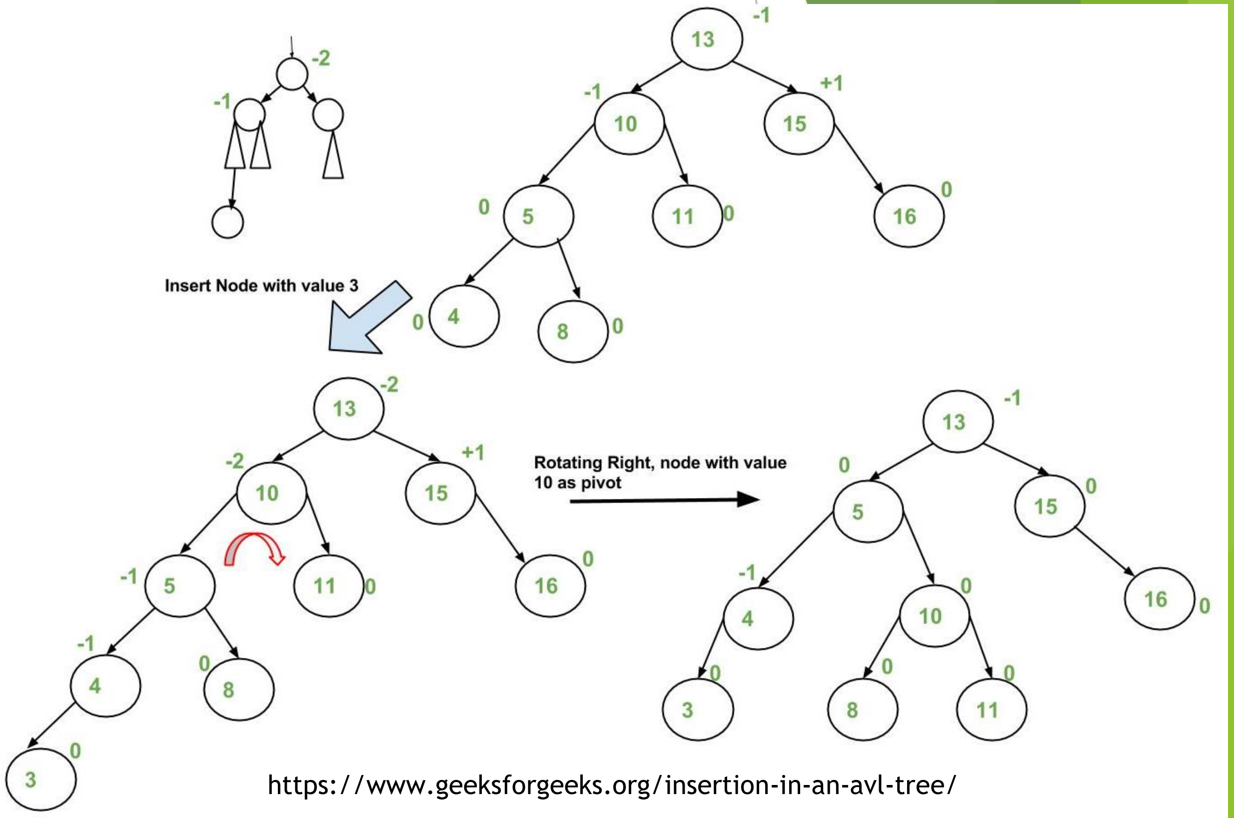
- Node 14 is inserted as left child of node 15.
- Balance factor of node 15 updated from +1 to 0.
- Balance factors for each node in this tree are still within range -1, 0, or +1. No balancing (rotation) required



https://www.geeksforgeeks.org/insertion-in-an-avl-tree/

AVL Tree Examples (cont.)

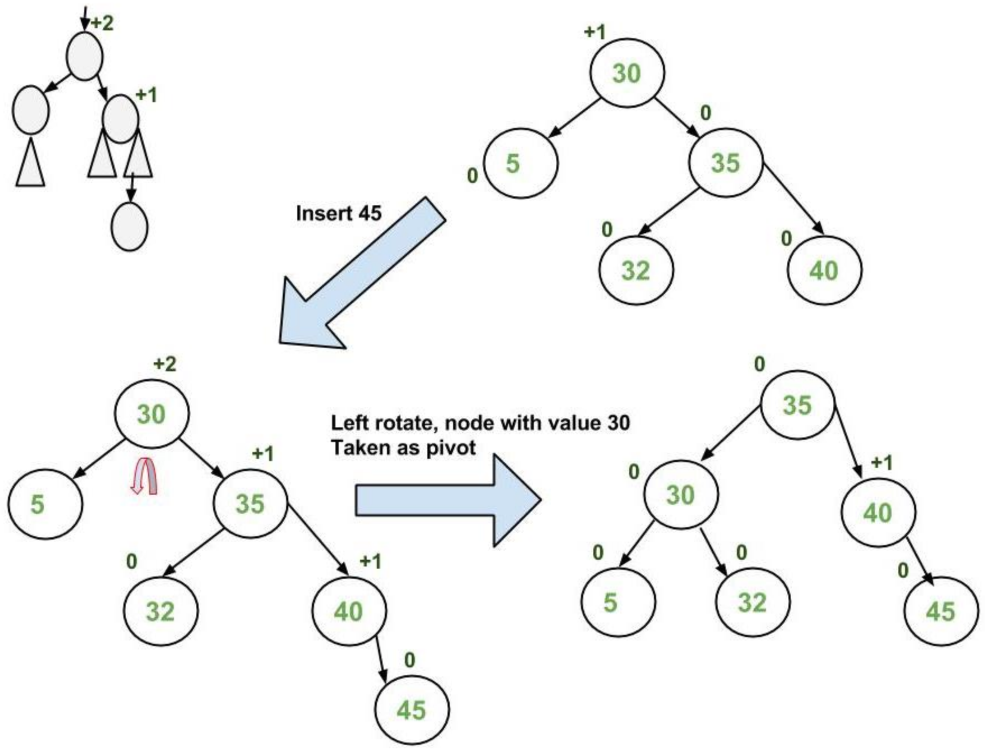
- Node 3 is inserted as left child of node 4.
- Balance factor of node 10 updated from -1 to -2.
- Right rotation at node 10 is done to balance the tree.



https://www.geeksforgeeks.org/insertion-in-an-avl-tree/

AVL Tree Examples (cont.)

- Node 45 is inserted as right child of node 40.
- Balance factor of node 30 is updated from +1 to +2.
- Right rotation at node 30 is done to balance the tree.



https://www.geeksforgeeks.org/insertion-in-an-avl-tree/

Summary

- BST is a binary tree which left subtree of a node has value less that the node and right subtree has value more than the node.
- Maximum value: right-most node, Minimum value: left-most node
- AVL Tree is a self-balancing BST.
- Balancing in AVL is done through rotation based on the balance factor

Next Topic…

- Sorting algorithms.

References

- Carrano, F. & Savitch, W. 2005. *Data Structures and Abstractions with Java, 2nd ed. Prentice-Hall*.
- Malik D.S, & Nair P.S., Data Structures Using Java, Thomson Course Technology, 2003.
- Rada Mihalcea, CSCE 3110 Data Structures and Algorithm Analysis notes, U of North Texas.