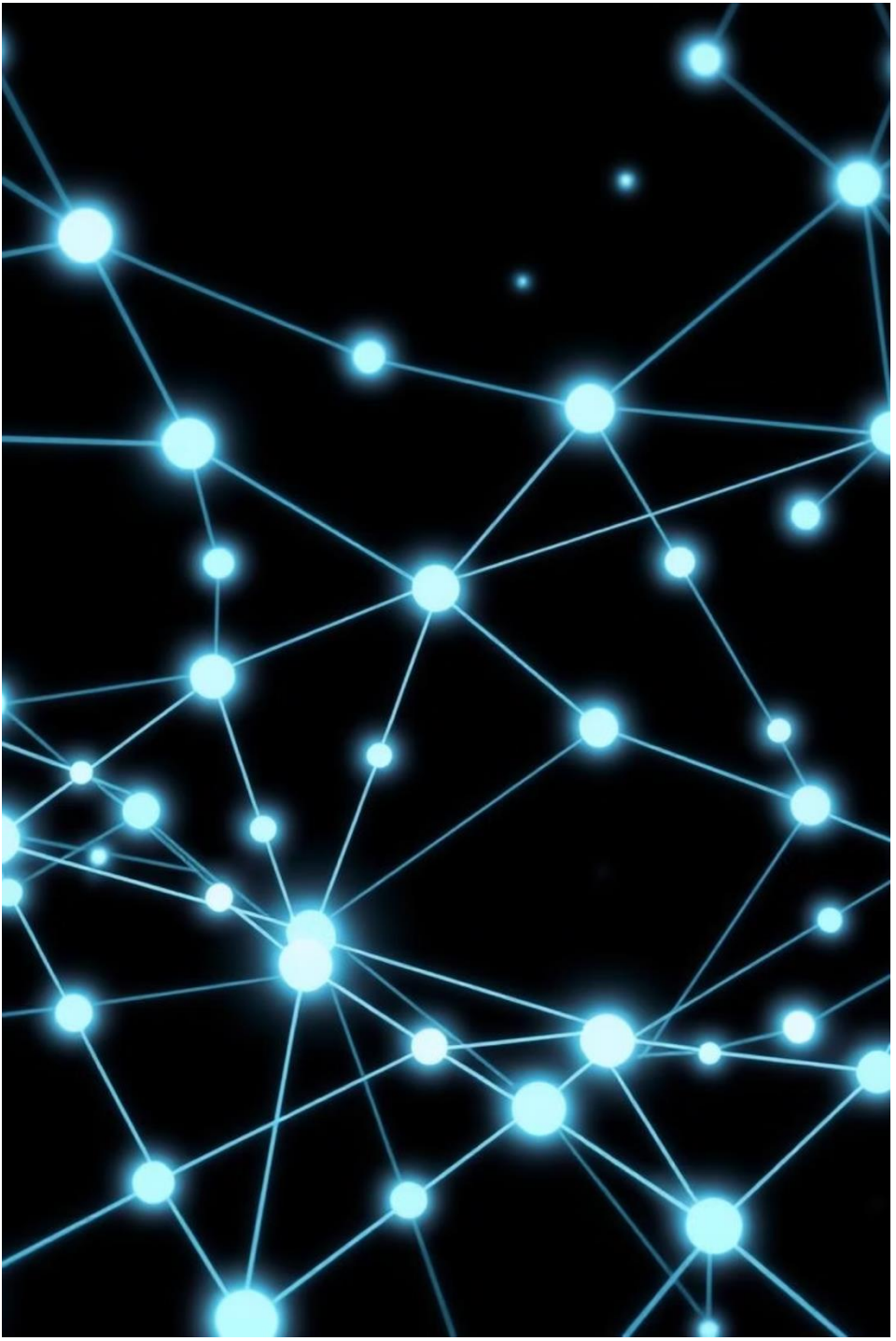


Data Structures

Lecture 6: BST

Instructor:
Md Samsuddoha
Assistant Professor
Dept of CSE, BU

Contents

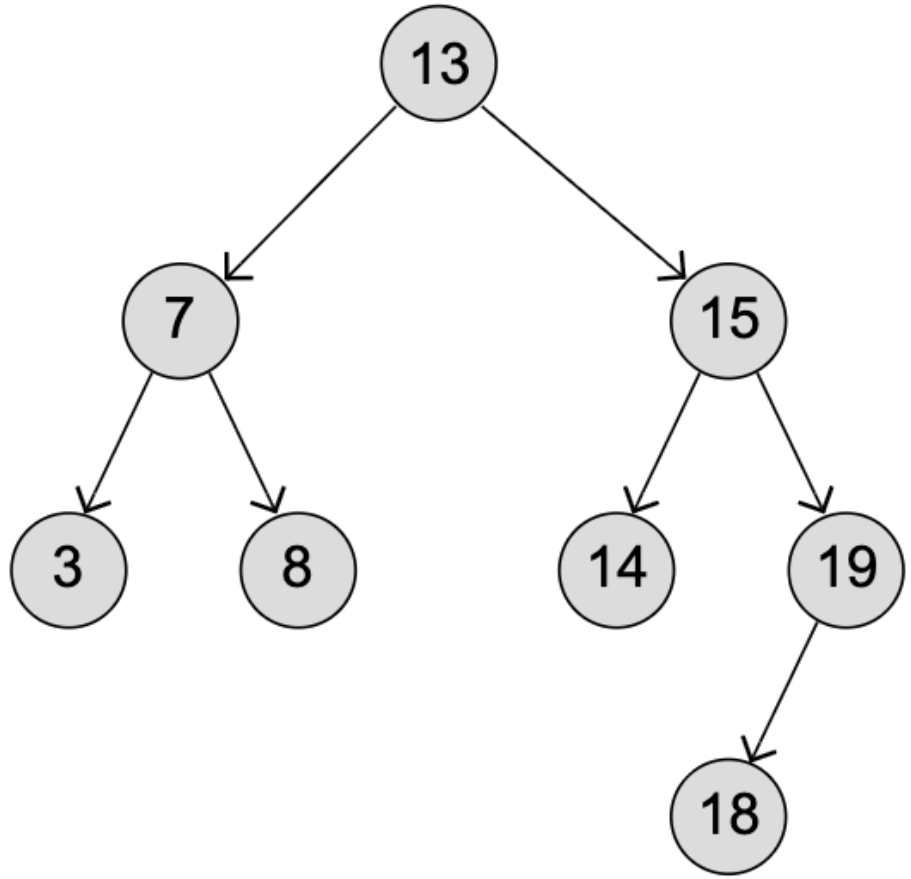
- Concept of BST
- Operations in BST

Binary Search Tree (BST)

- A **BST** is a ***Binary Tree*** where every node's ***left child has a lower value***, and every node's ***right child has a higher value.***

  [Left < Root < Right]

- A clear advantage with BST is that operations like ***search, delete, and insert*** are fast and done without having to shift values in memory.
- Following properties must be true for any node ***X*** in the tree:
  - The ***X*** node's left child and all of its ***descendants (children, children's children, and so on)*** have lower values than X's value.
  - The right child, and all its descendants have higher values than X's value.
  - Left and right subtrees must also be Binary Search Trees.


13
7
15
3
8
14
19
18

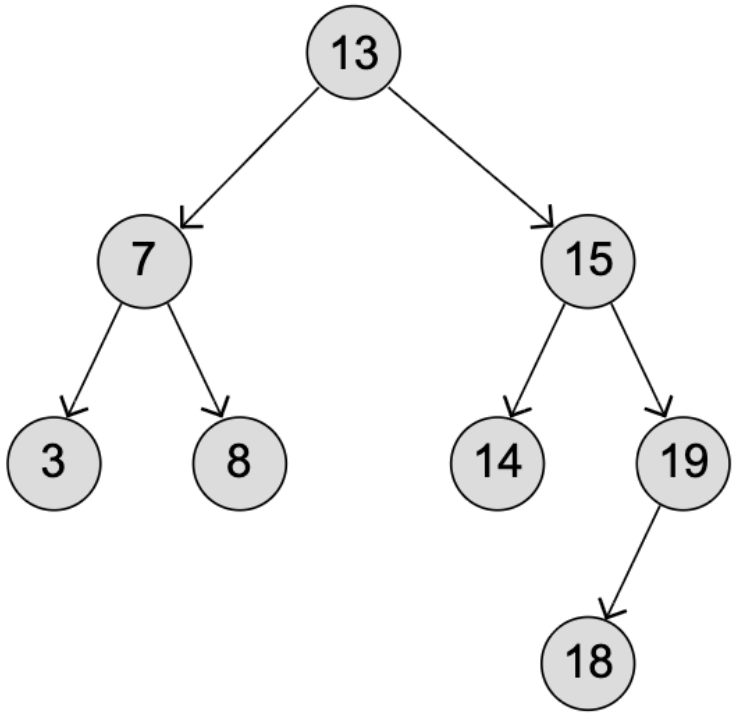
Operations in BST

- Traversal
- Search
- Insert
- Delete

Traversal in BST

- Traversal in BST is same as Binary Tree.
  - Preorder
  - ***Inorder (Provides a sorted list)***
  - Postorder
  - Level-order (BFS)

- For the given tree:
  - Preorder: 13, 7, 3, 8, 15, 14, 19, 18
  - Inorder:3, 7, 8, 13, 14, 15, 18, 19
  - Postorder: 3, 8, 7, 14, 18, 19, 15, 13
  - Level-order: 13, 7, 15, 3, 8, 14, 19, 18



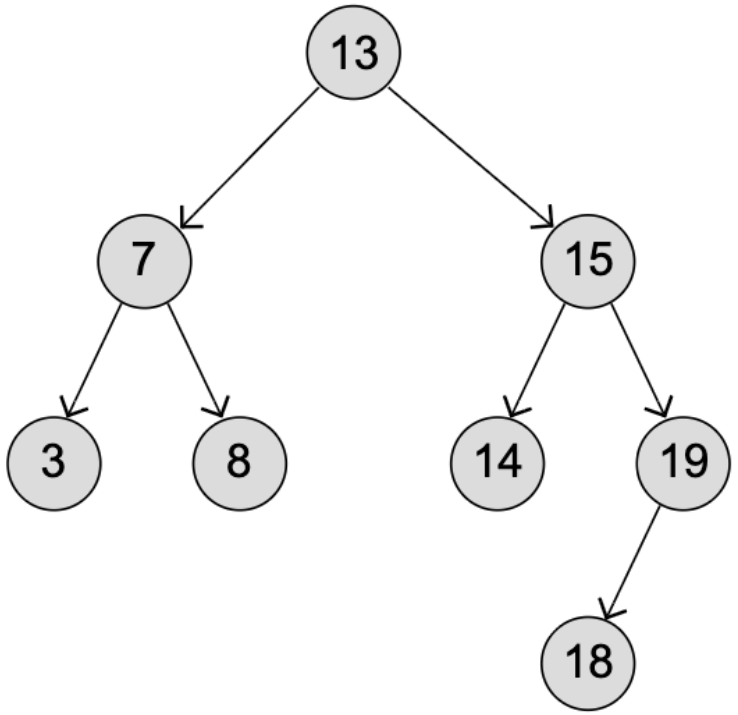
Search in BST

- **Searching** for a value in a **BST** is very ***similar*** to how we found a value using ***Binary Search on an array.***
- **How it Works:**
  1. Start at the root.
  2. If ***key == root's data*** → found.
  3. If ***key < root's data*** → go to left subtree.
  4. If ***key > root's data*** → go to right subtree.
  5. Continue until found or reach NULL.
- *So, it behaves exactly like binary search.*

Insert a Node in BST

- ***Inserting*** a node in a BST is ***similar to searching*** for a value.
- **Process**:
  1. Start at the root.
  2. Compare new value with current node:
     If ***new value < node*** → go left
     If ***new value > node*** → go right
  3. Repeat until you reach an empty spot.
  4. Insert the new value there.

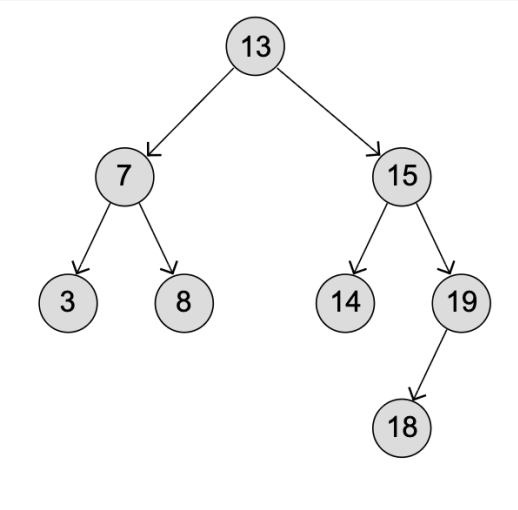
- ***<u>Insert: 10, 16</u>***

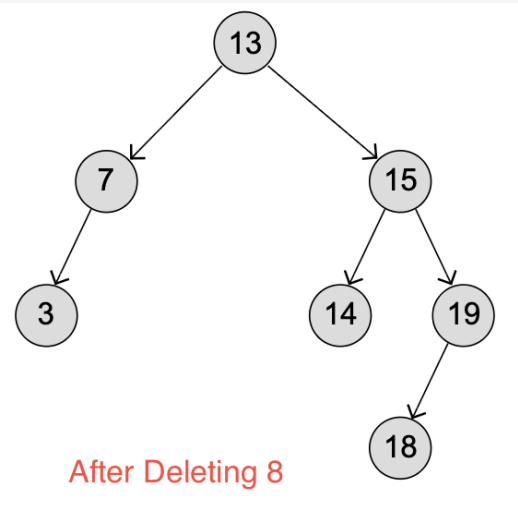


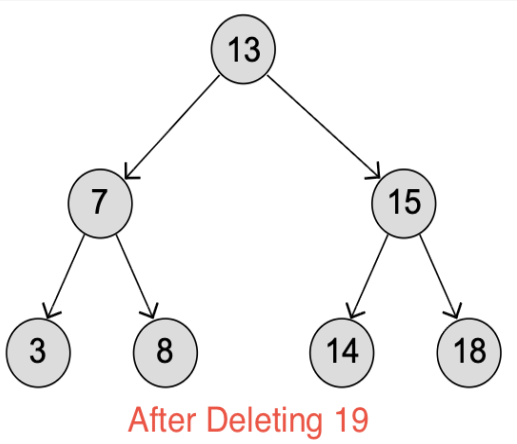
Delete a Node in BST

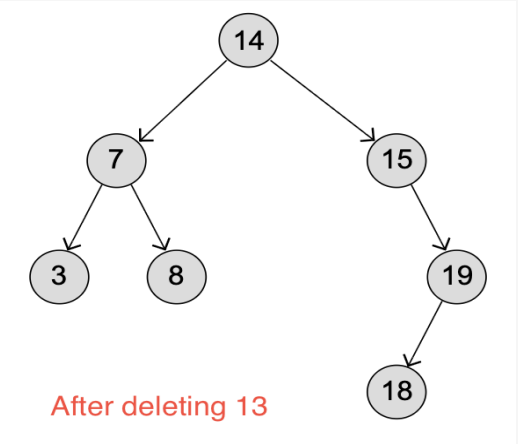
- To delete a node, our function must first ***search*** the BST to find it.
- After the node is found there are ***three different cases*** where deleting a node must be done differently.
  1. If the node is a ***leaf node***, ***remove it*** by removing the link to it.
  2. If the node only has one child node, connect the ***parent node*** of the node you want to remove ***to that child node.***
  3. If the node has both right and left child nodes:
     a. Find **inorder successor** (smallest value in right subtree) or **inorder predecessor** (largest value in left subtree).
     b. Replace the node's value with successor's value and Delete the successor or predecessor.

Delete Nodes [**8, 19, 13, 15**]









Construct a Binary Tree

- ***Similar to Binary Tree***
- It follows: Preorder, inorder, and postorder
- ***Construting tree from random list: 10, 6, 4, 3, 5, 8, 18, 15, 21, 22***

M-ways Tree

- B Tree
- B+ Tree

Tree→ Binary Tree → Binary Search Tree→ AVL Tree

References

- **Chapter 8: Tree (Data Structures using C** by E. Balagurusamy)

Thank You