UNIVERSITY OF CALGARY

CPSC 319
Data Structures, Algorithms, and Their Applications

Winter 2024

Binary Search Trees

- A Binary Search Tree is a hierarchical data structure that organizes elements in a tree-like structure.

- Each node has at most two children – a left child and a right child.

- The key in each node must be greater than all keys in its left subtree and less than all keys in its right subtree.

Basic Structure of a Binary Search Tree

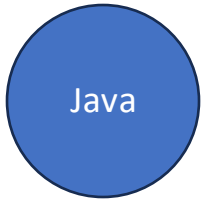
- Nodes: Each element in the tree is represented by a node.
- Root Node: The topmost node in the tree.
- Parent and Child Nodes: Relationship between nodes, where a node can be a parent, left child, or right child.
- Leaf Nodes: Nodes with no children.

Insertion Operation in BST

1. Start at the root.
2. Compare the new key with the current node's key.
3. If smaller, move to the left subtree; if larger, move to the right subtree.
4. Repeat until an empty spot is found, then insert the new node.

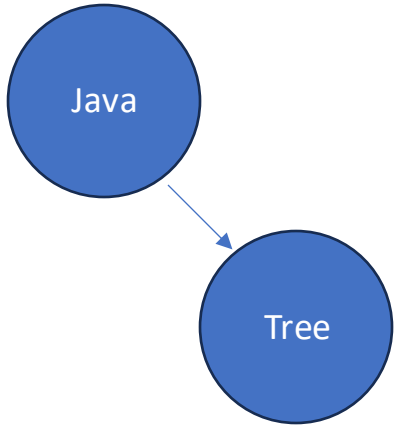
Insertion Operation in BST

- **Word 1 = "Java"**
- Word 2 = "Tree"
- Word 3 = "Binary"
- Word 4 = "Search"
- Word 5 = "CPSC"



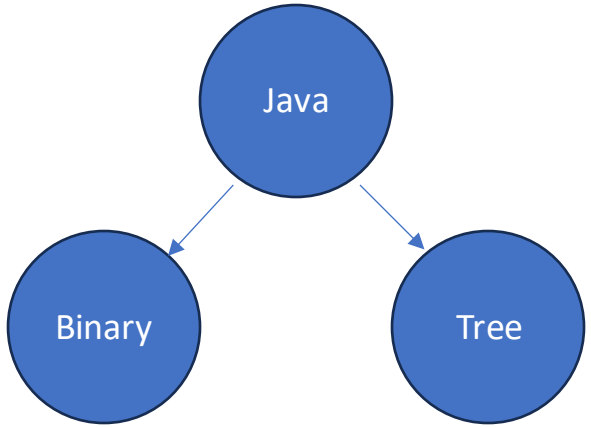
Insertion Operation in BST

- Word 1 = "Java"
- **Word 2 = "Tree"**
- Word 3 = "Binary"
- Word 4 = "Search"
- Word 5 = "CPSC"



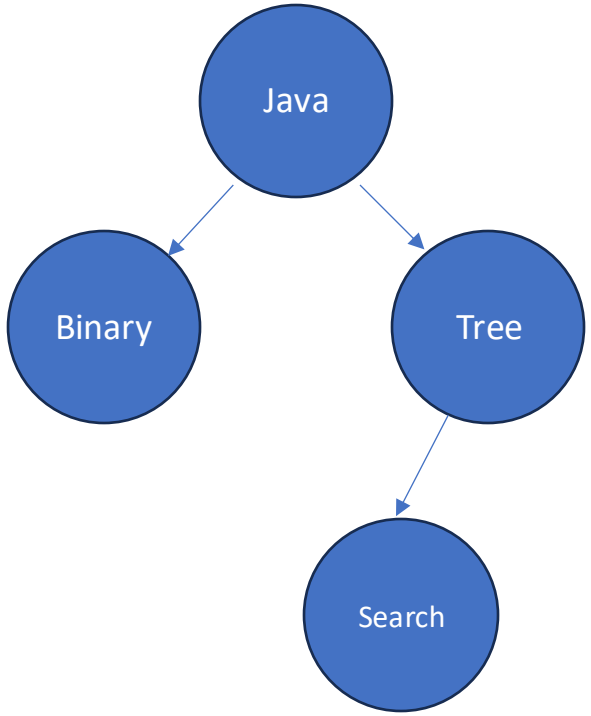
Insertion Operation in BST

- Word 1 = "Java"
- Word 2 = "Tree"
- **Word 3 = "Binary"**
- Word 4 = "Search"
- Word 5 = "CPSC"



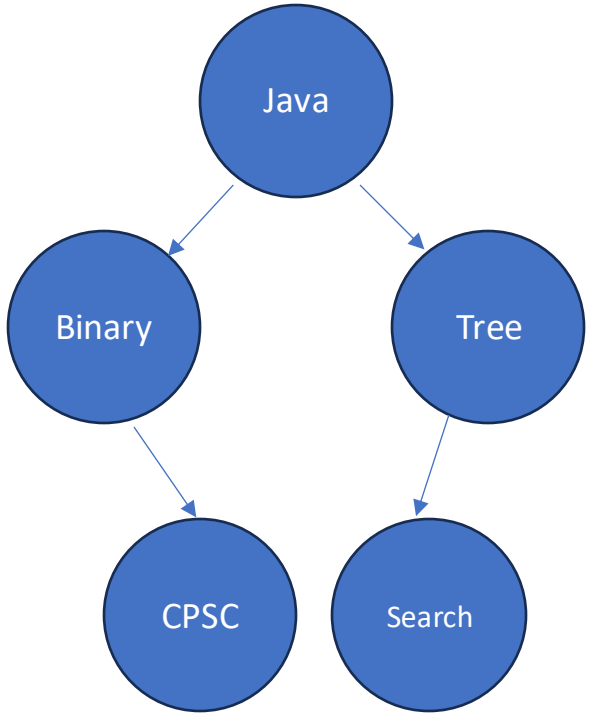
Insertion Operation in BST

- Word 1 = "Java"
- Word 2 = "Tree"
- Word 3 = "Binary"
- **Word 4 = "Search"**
- Word 5 = "CPSC"



Insertion Operation in BST

- Word 1 = "Java"
- Word 2 = "Tree"
- Word 3 = "Binary"
- Word 4 = "Search"
- **Word 5 = "CPSC"**



Searching in BST

1. Start at the root.
2. Compare the target key with the current node's key.
3. If equal, the element is found. If smaller, move to the left subtree; if larger, move to the right subtree.
4. Repeat until the element is found or the end of the tree is reached.

Time Complexity of Operations

- Insertion and Search: O(log n) on average, where n is the number of nodes.
- Best Case: O(1) for operations in a well-balanced tree.
- Worst Case: O(n) for a skewed tree.

Advantages of Binary Search Trees

- Efficient Search: Logarithmic time complexity for balanced trees.
- Ordered Structure: Useful for tasks requiring ordered data.
- Space Efficiency: Compact representation compared to other data structures.