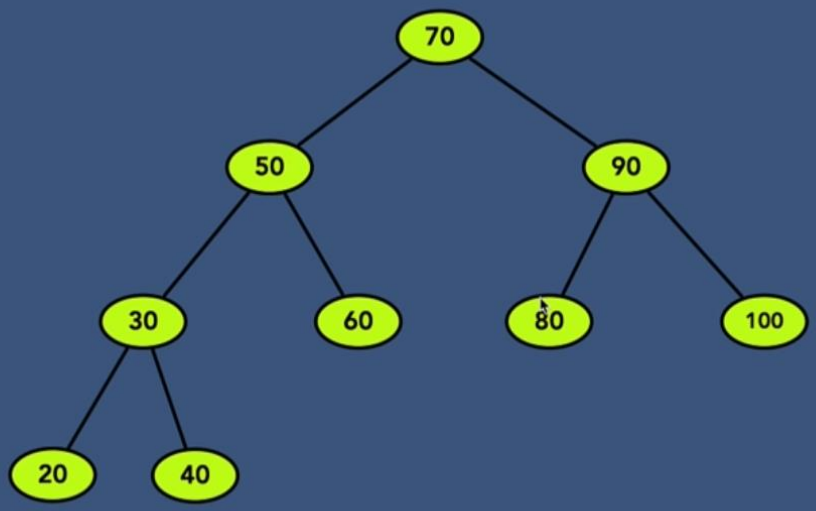
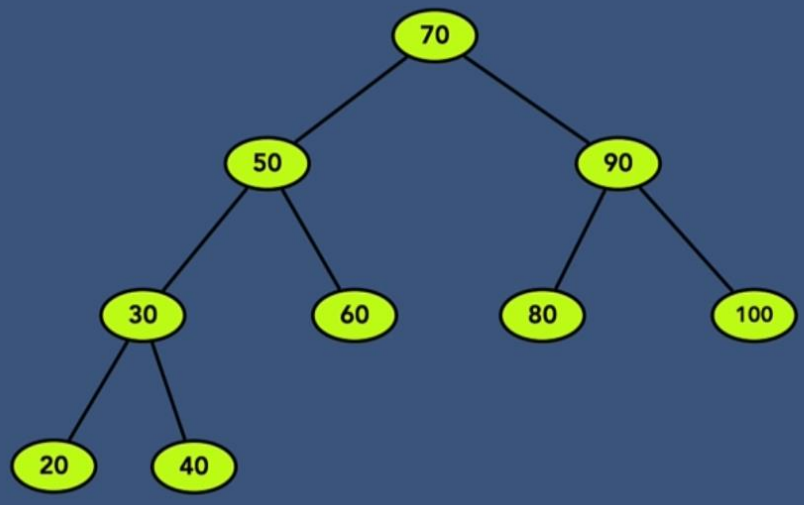
What is a Binary Search Tree?

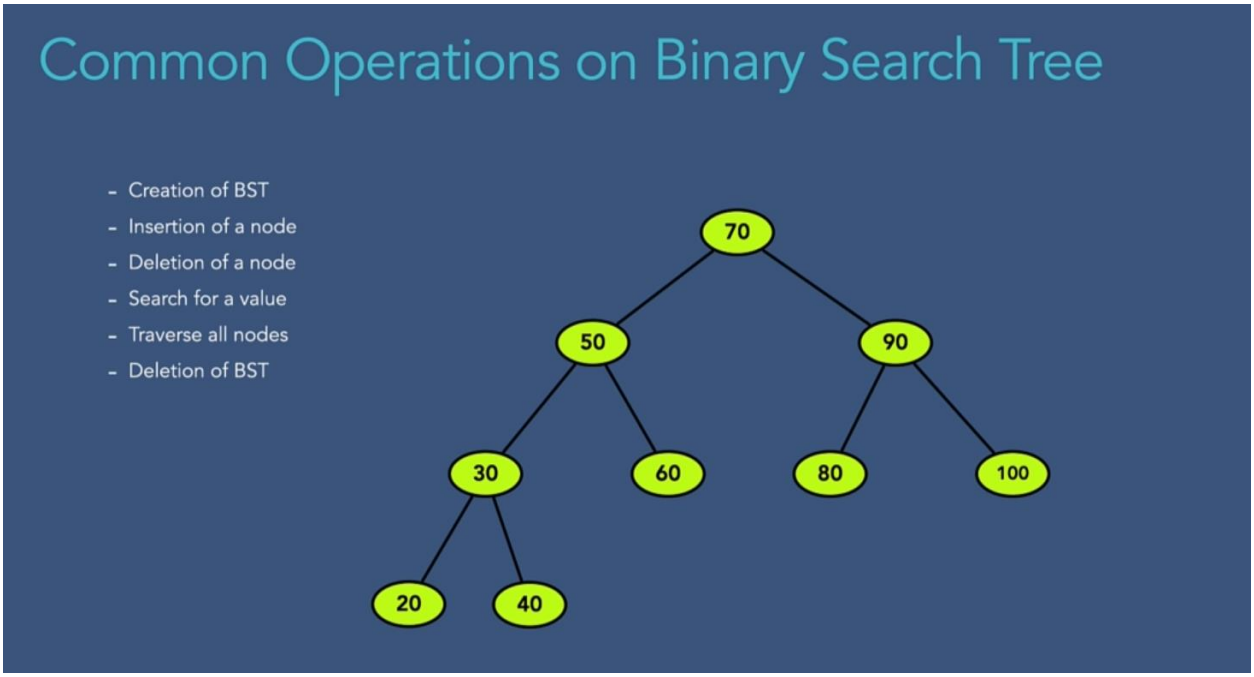
- In the left subtree the value of a node is less than or equal to its parent node's value.
- In the right subtree the value of a node is greater than its parent node's value



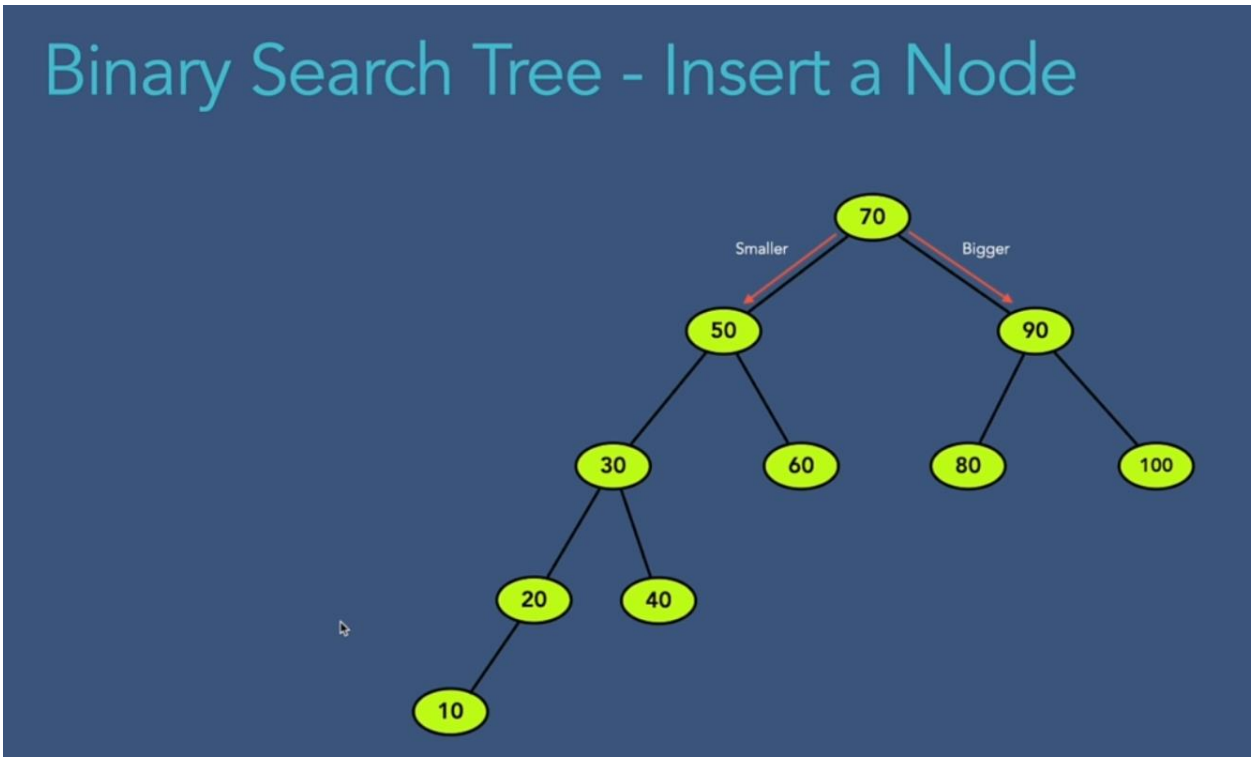
Why Binary Search Tree?

- It performs faster than Binary Tree when inserting and deleting nodes

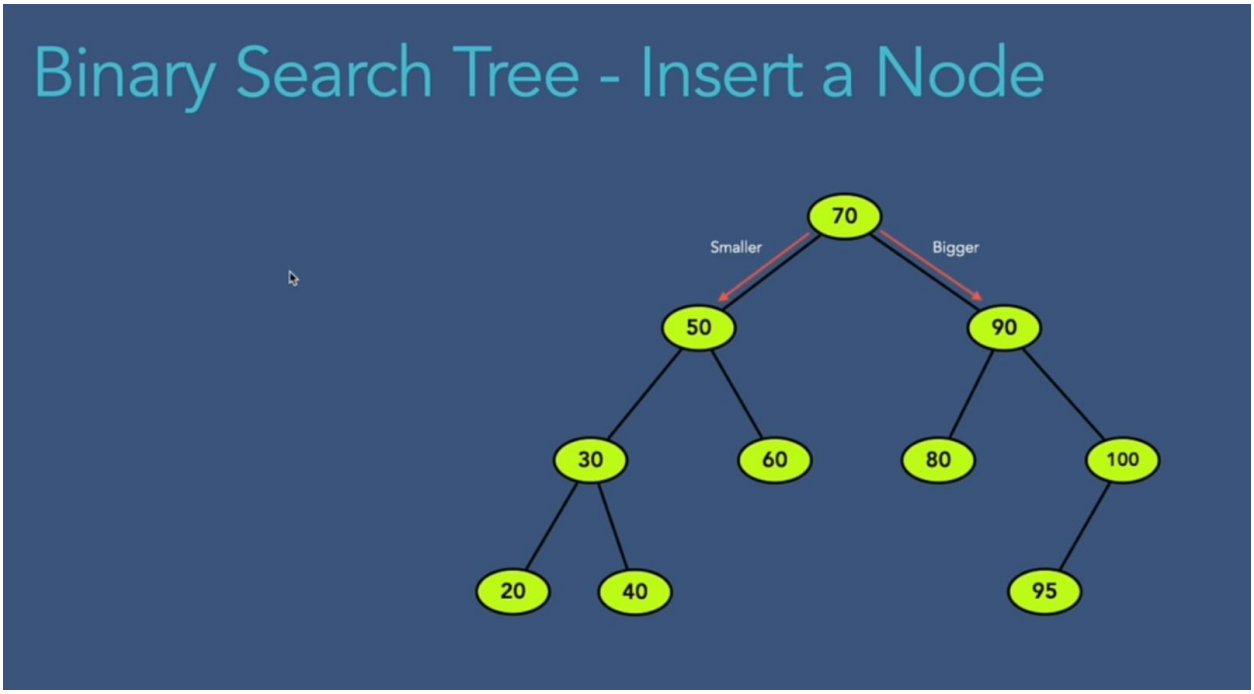


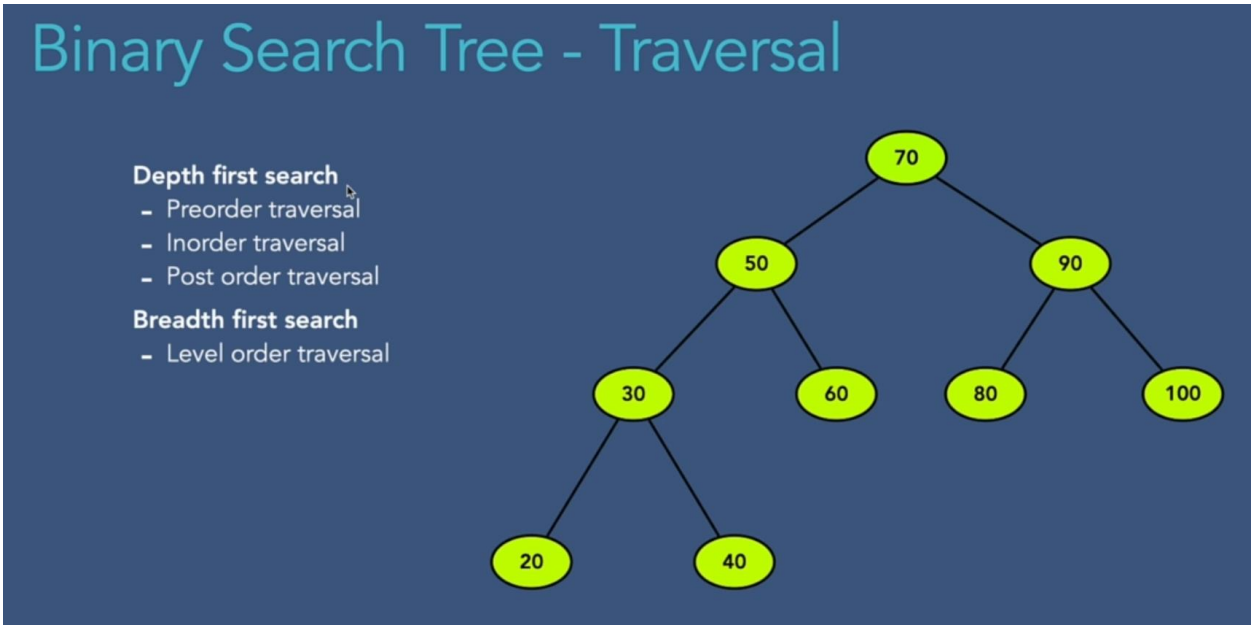


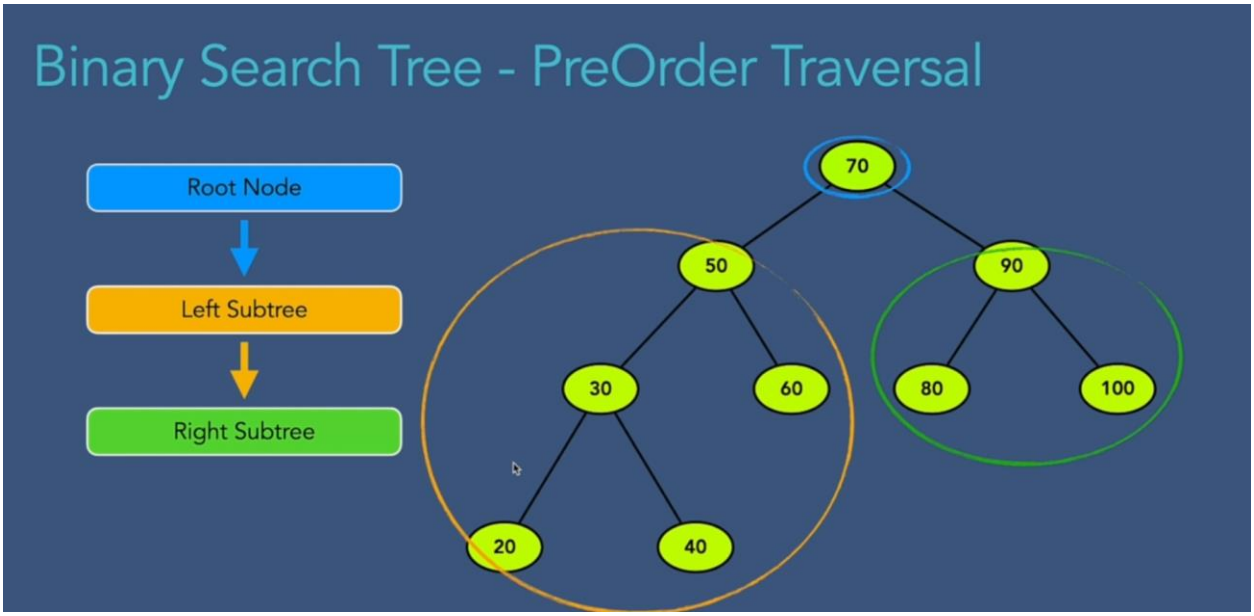
Adding node 10 to the binary search tree

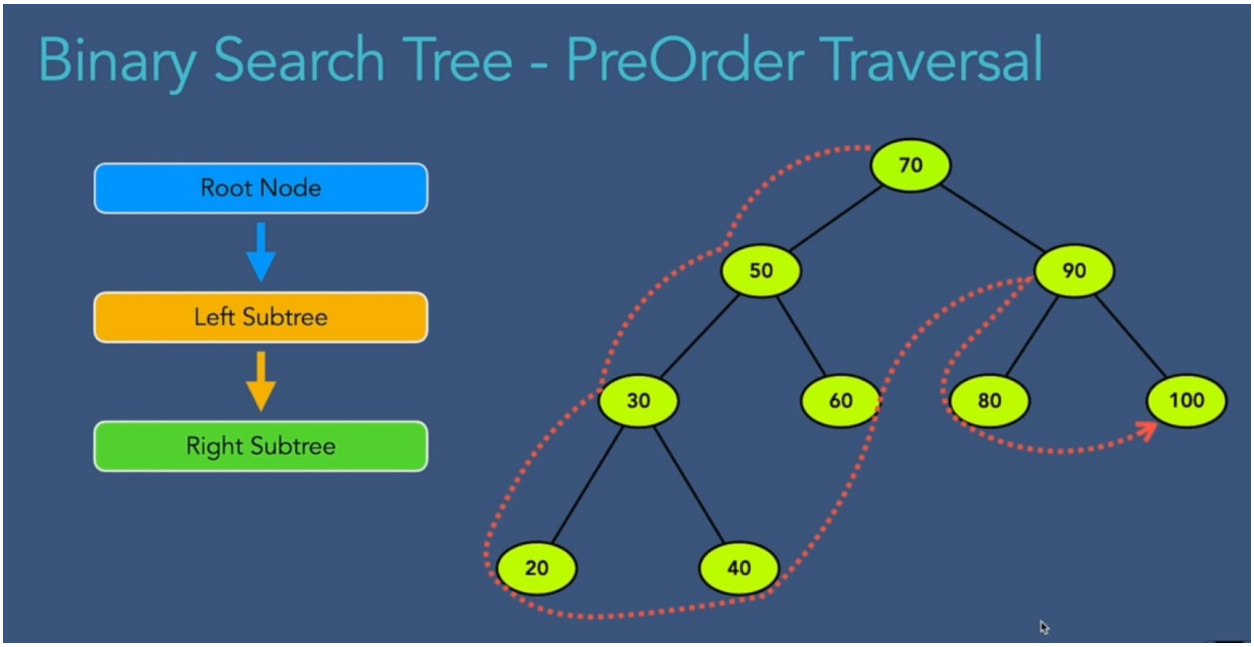


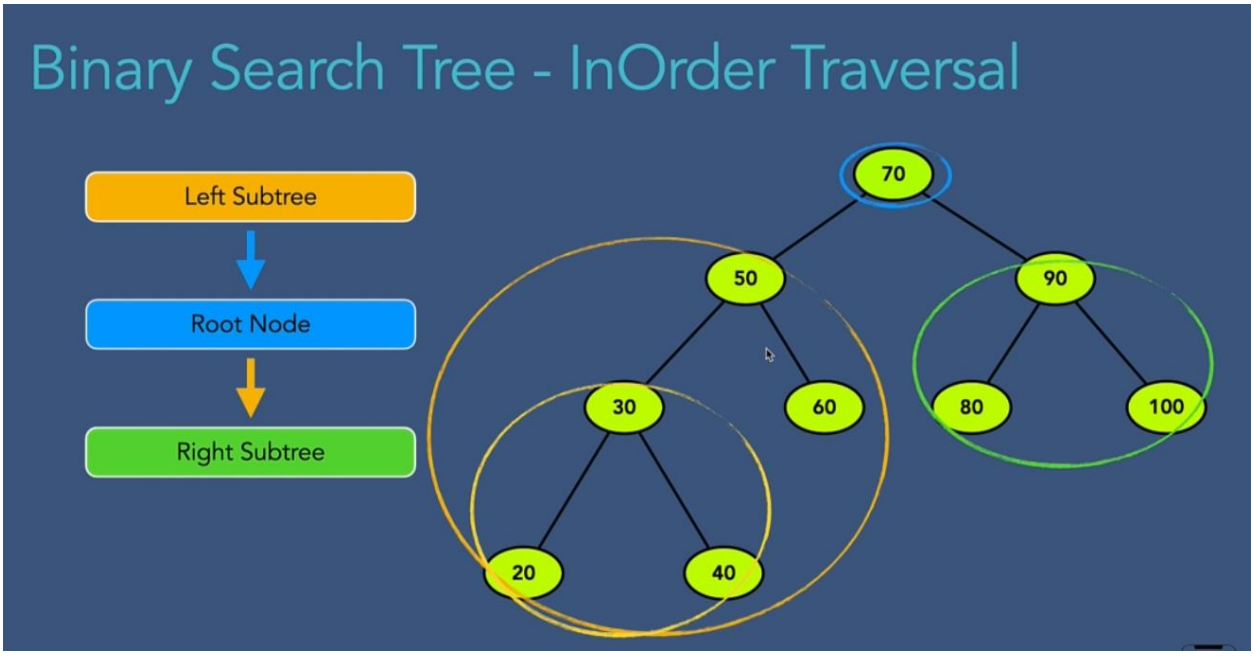
Inserting Node 95 to the binary search tree

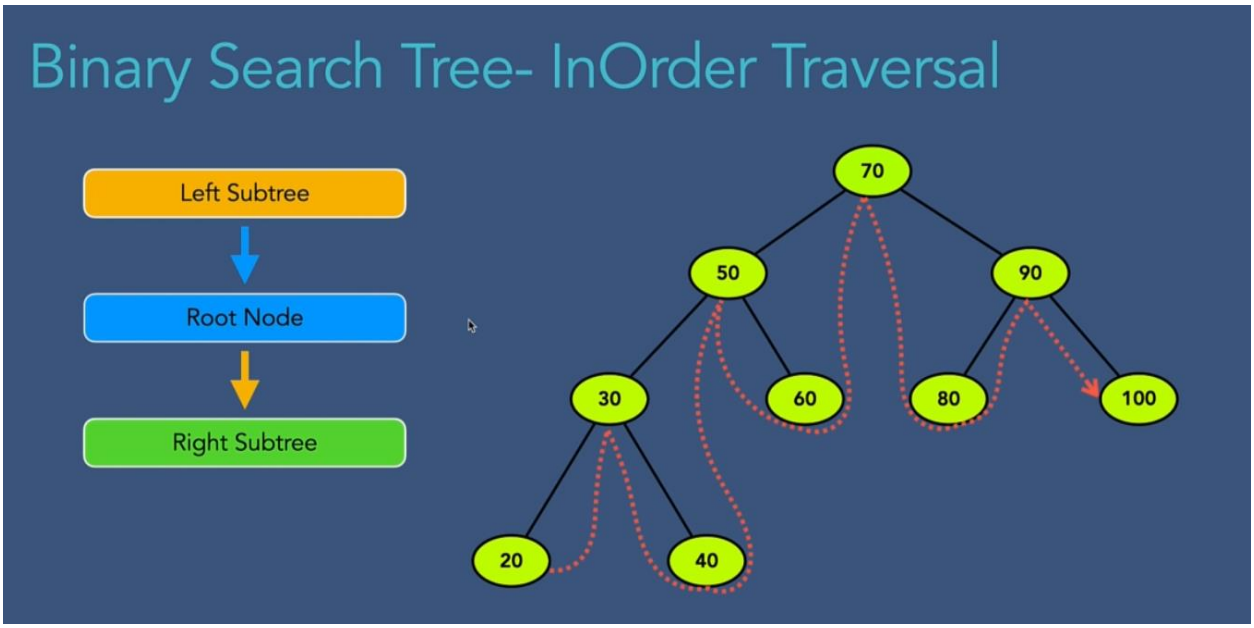


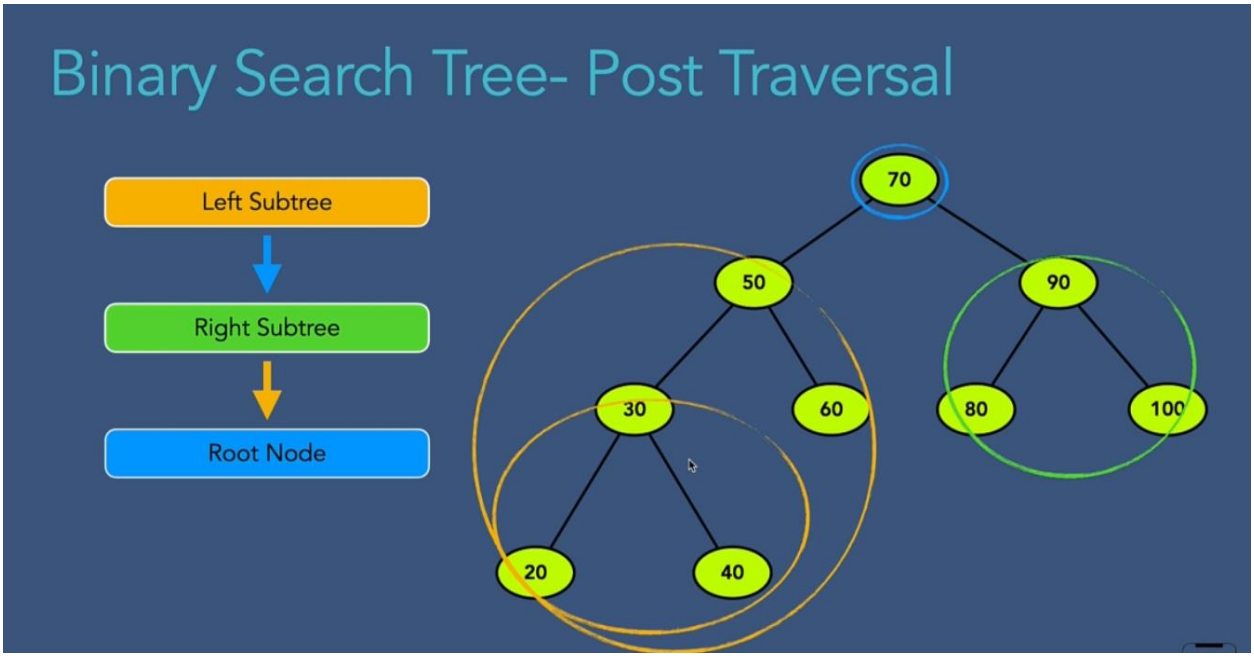


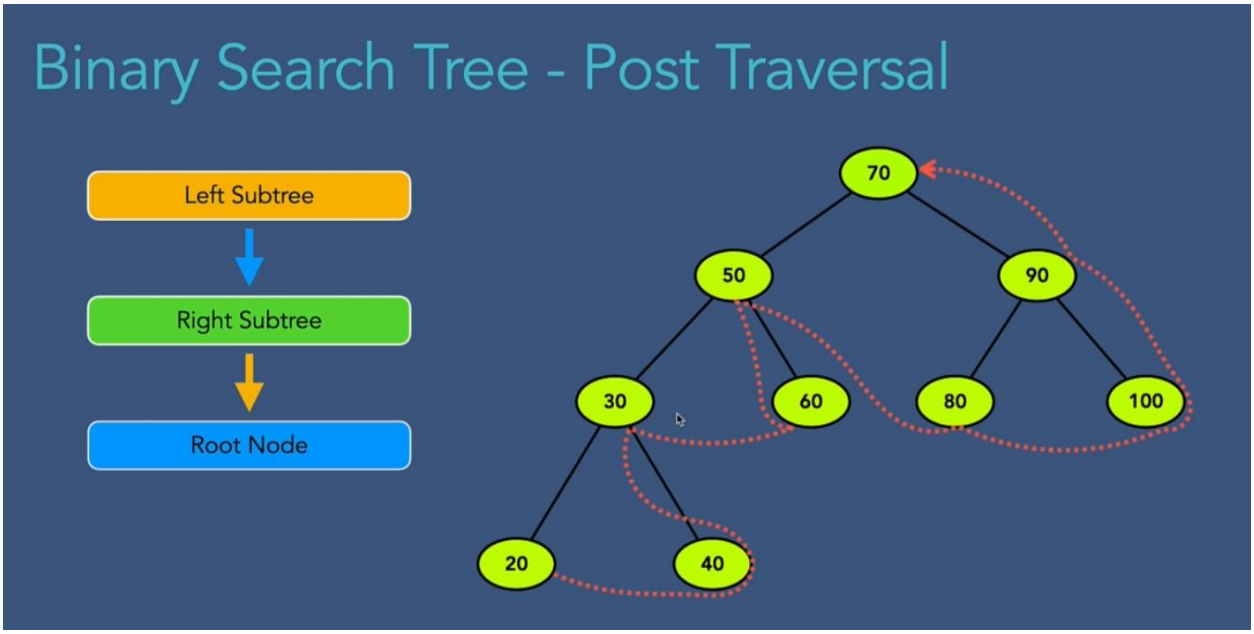

Binary Search Tree - PreOrder Traversal
Root Node
Left Subtree
Right Subtree
70
50
90
30
60
80
100
20
40

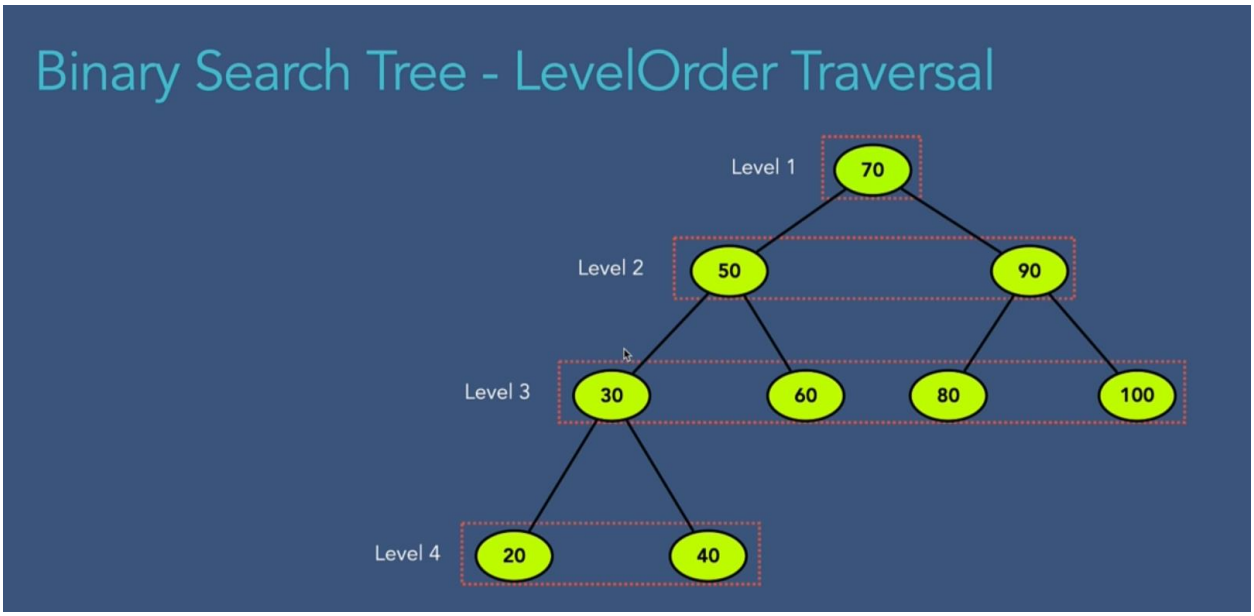

Binary Search Tree - PreOrder Traversal
Root Node
Left Subtree
Right Subtree
70
50
90
30
60
80
100
20
40

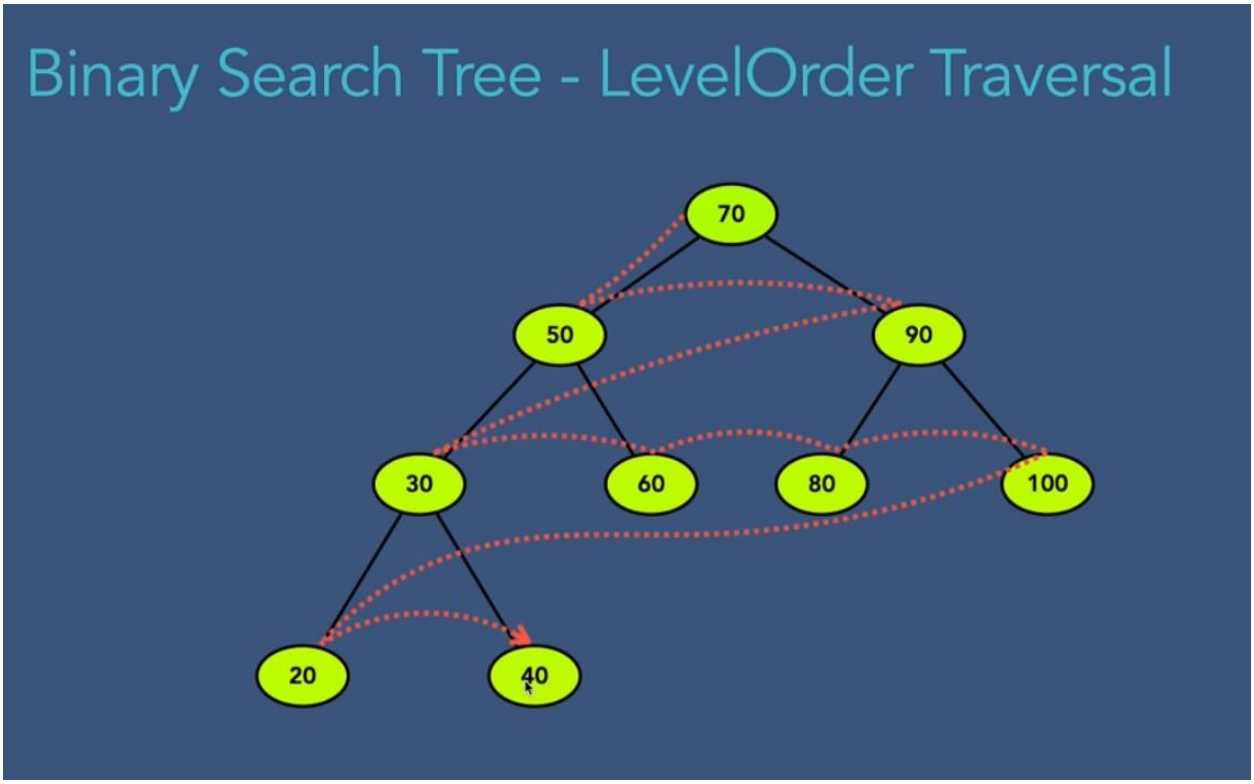

Binary Search Tree - InOrder Traversal
Left Subtree
Root Node
Right Subtree
70
50
90
30
60
80
100
20
40


Binary Search Tree- InOrder Traversal
Left Subtree
Root Node
Right Subtree
70
50
90
30
60
80
100
20
40

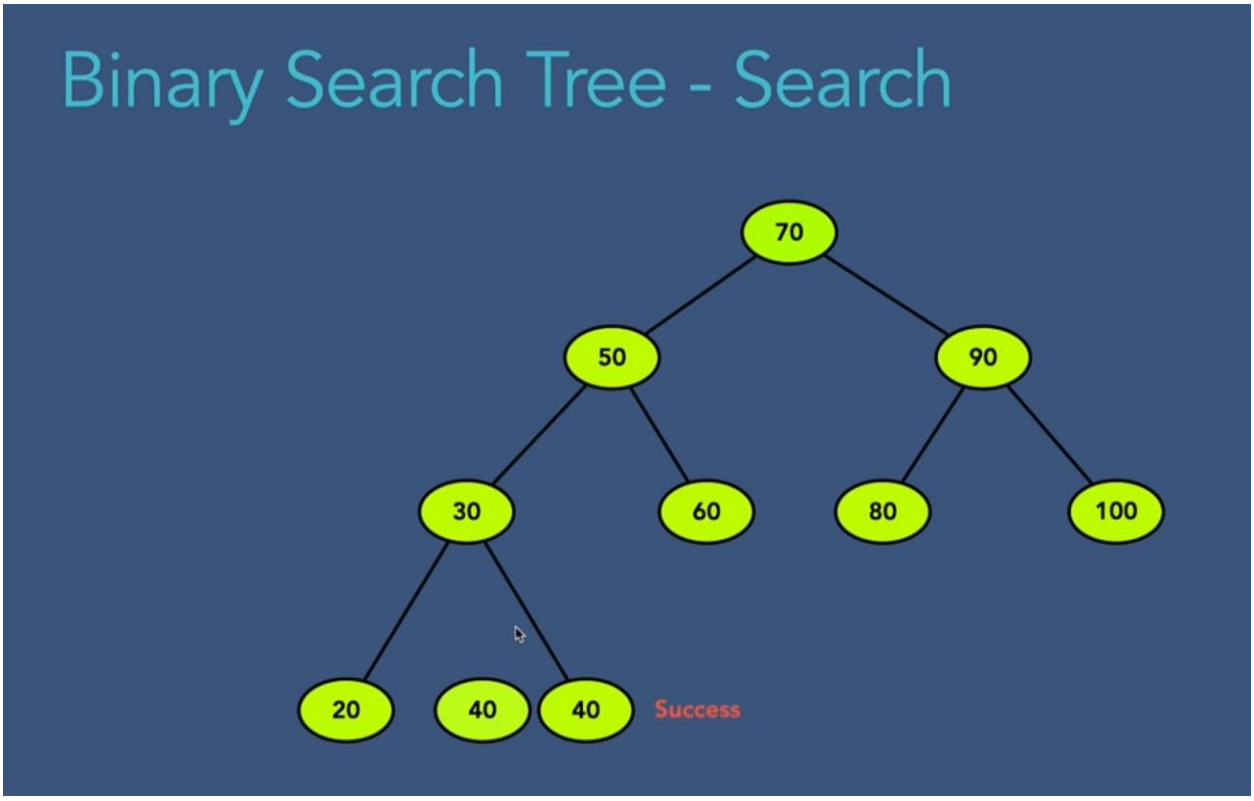

Binary Search Tree- Post Traversal
Left Subtree
Right Subtree
Root Node
70
50
90
30
60
80
100
20
40

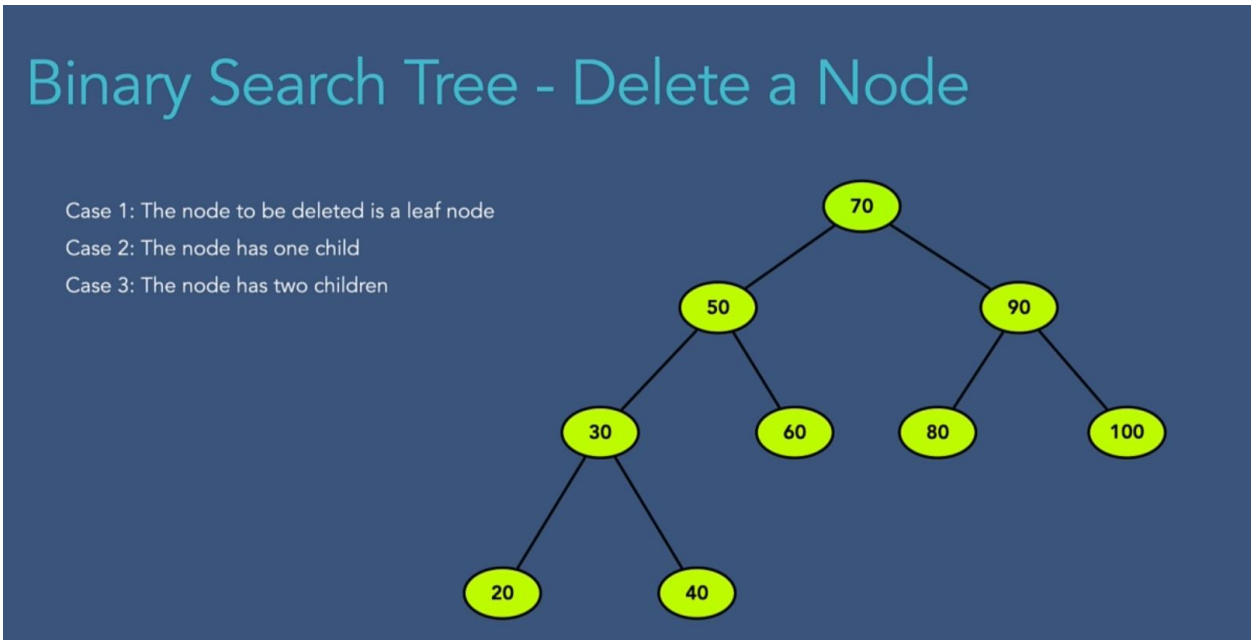

Binary Search Tree - Post Traversal
Left Subtree
Right Subtree
Root Node
70
50
90
30
60
80
100
20
40


Binary Search Tree - LevelOrder Traversal
Level 1
70
Level 2
50
90
Level 3
30
60
80
100
Level 4
20
40

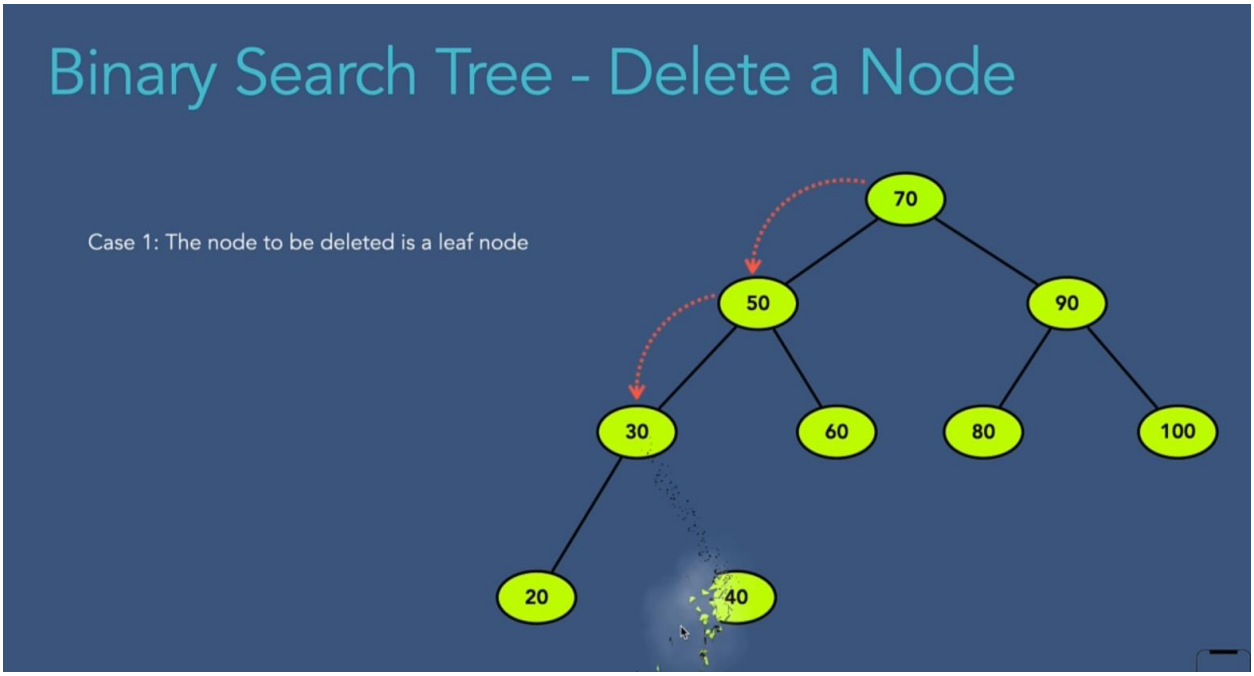

Binary Search Tree - LevelOrder Traversal
70
50
90
30
60
80
100
20
40

Search Node 40





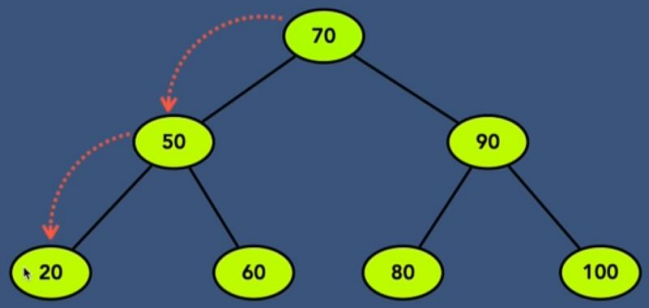
Delete Node 40 (Leaf Node) from the Binary Search Tree



Delete Node 30 (Node with one child) from the Binary Search Tree

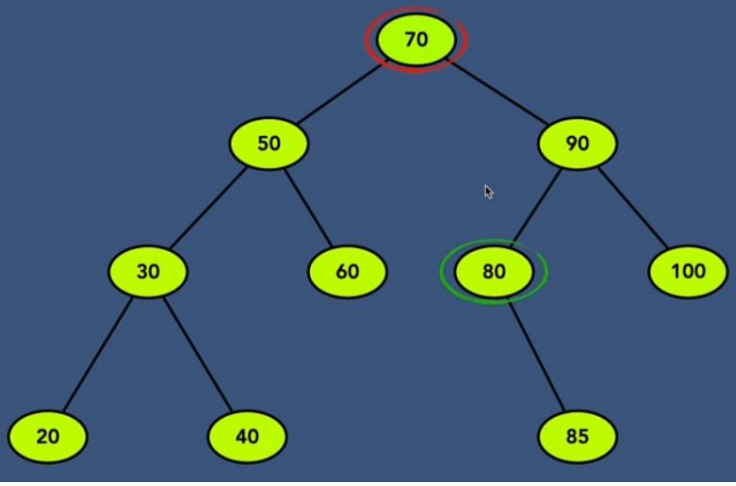
Binary Search Tree - Delete a Node

Case 2: The node has one child



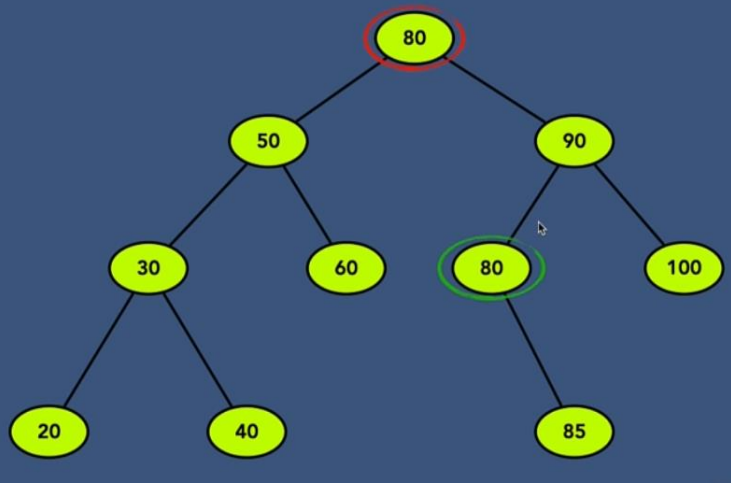
Binary Search Tree - Delete a Node

Case 3: The node has two children



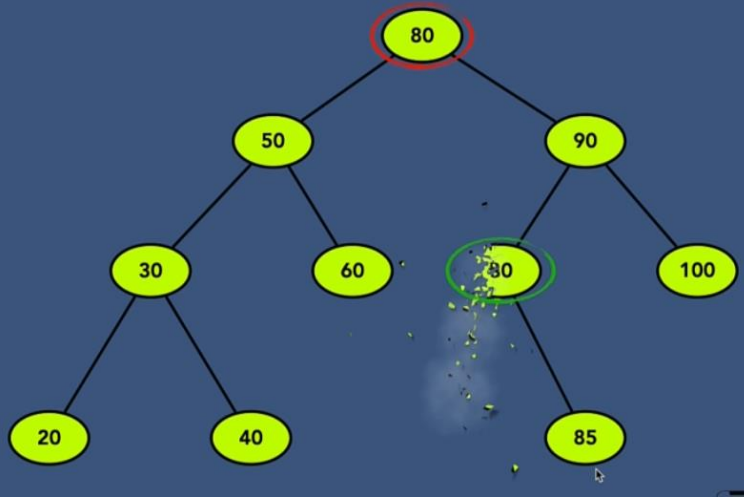
Binary Search Tree - Delete a Node

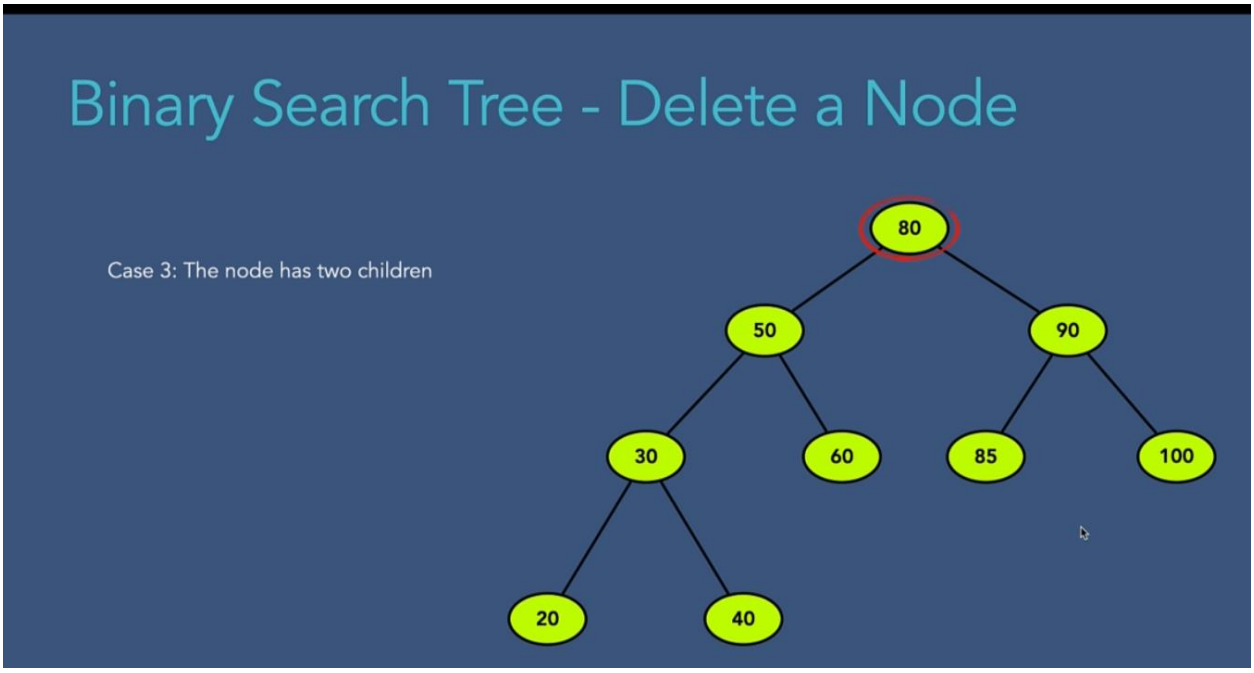
Case 3: The node has two children



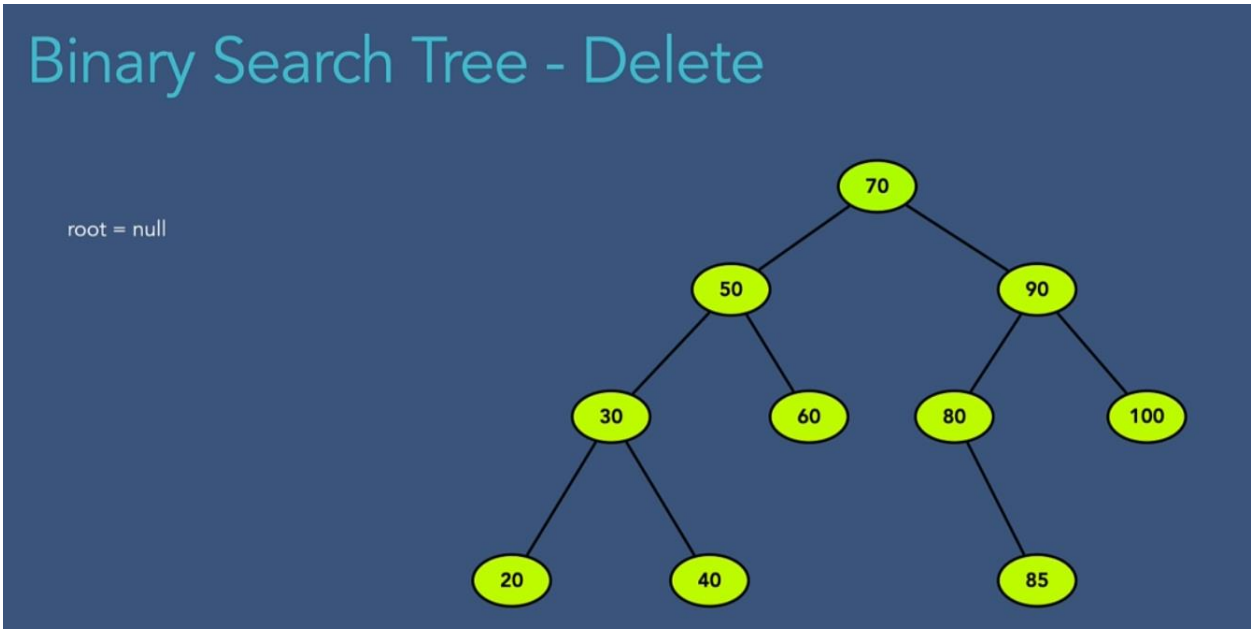
Binary Search Tree - Delete a Node

Case 3: The node has two children



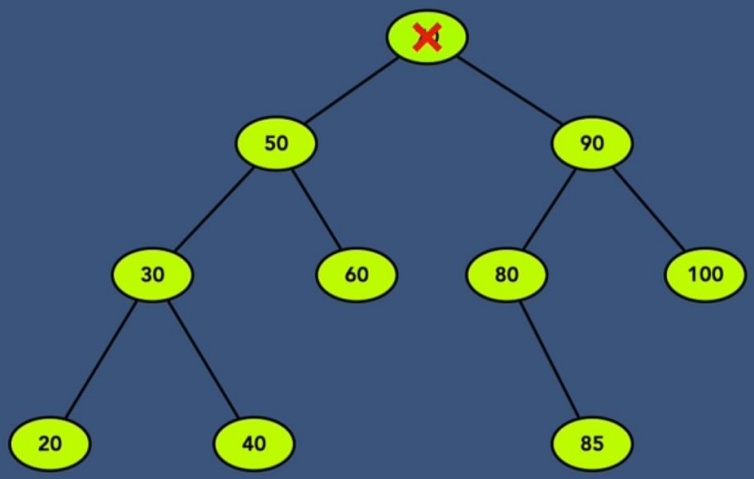


To find the successor of the root node, is to find the smallest node in the right sub tree.



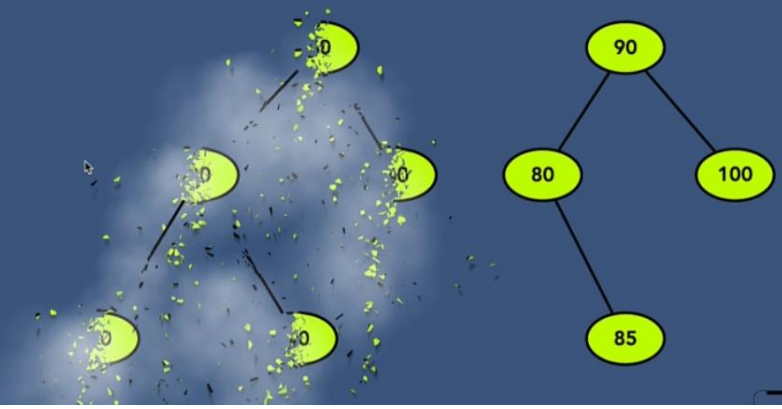
Binary Search Tree - Delete

root = null



Binary Search Tree - Delete

root = null



Binary Search Tree - Time and Space Complexity

	Time complexity	Space complexity
Create BST	O(1)	O(1)
Insert a node BST	O(logN)	O(logN)
Traverse BST	O(N)	O(N)
Search for a node BST	O(logN)	O(logN)
Delete node from BST	O(logN)	O(logN)
Delete Entire BST	O(1)	O(1)