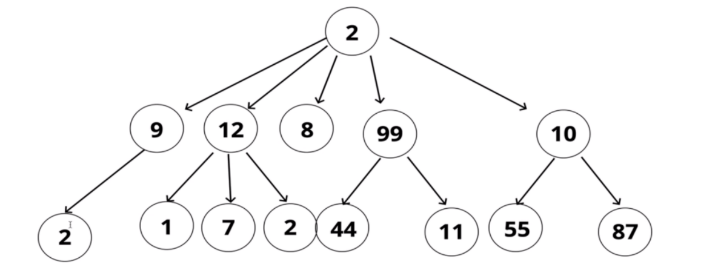
Tree Implementation

1. **class** Node:
2. **def** \_\_init\_\_(self, data):
3. self.data = data
4. self.left = None
5. self.right = None
7. **class** BinarySearchTree:
8. **def** \_\_init\_\_(self):
9. self.root = None
11. **def** insert(self, data):
12. node = Node(data)
13. **if** **not** self.root:
14. self.root = node
15. **return** self
16. current = self.root
17. **while** True:
18. **if** data == current.data:
19. **return** None
20. **if** data < current.data:
21. **if** current.left == None:
22. current.left = node
23. **return** self
24. current = current.left
25. **else**:
26. **if** current.right == None:
27. current.right = node
28. **return** self
29. current = current.right
31. **def** contains(self, data):
32. **if** **not** self.root:
33. **return** False
34. current = self.root
35. **while** current:
36. **if** data < current.data:
37. current = current.left
38. **elif** data > current.data:
39. current = current.right
40. **else**:
41. **return** True
42. **return** False
44. **def** find(self, data):
45. **if** **not** self.root:
46. **return** False
47. current = self.root
48. found = False
49. **while** current **and** **not** found:
50. **if** data < current.data:
51. current = current.left
52. **elif** data > current.data:
53. current = current.right
54. **else**:
55. found = True
56. **if** **not** found:
57. **return** None
58. **return** current

1. **def** BFS(self):
2. node = self.root
3. li, queue = [], []
4. queue.append(node)
5. **while** queue:
6. node = queue.pop(0)
7. li.append(node.data)
8. **if** node.left:
9. queue.append(node.left)
10. **if** node.right:
11. queue.append(node.right)
12. **return** li
14. **def** DFSPreOrder(self):
15. li = []
16. **def** traverse(node):
17. li.append(node.data)
18. **if** node.left:
19. traverse(node.left)
20. **if** node.right:
21. traverse(node.right)
22. traverse(self.root)
23. **return** li
25. **def** DFSPostOrder(self):
26. li = []
27. **def** traverse(node):
28. **if** node.left:
29. traverse(node.left)
30. **if** node.right:
31. traverse(node.right)
32. li.append(node.data)
33. traverse(self.root)
34. **return** li
36. **def** DFSInOrder(self):
37. li = []
38. **def** traverse(node):
39. **if** node.left:
40. traverse(node.left)
41. li.append(node.data)
42. **if** node.right:
43. traverse(node.right)
44. traverse(self.root)
45. **return** li



### Binary Search Tree Big O

* Insertion = O(log n)
* Deletion = O(1)
* Searching = O(log n)
* Access = O(n)