Preethika Kiruveedula

Carey Nachenberg Lecture 1

204912895

Computer Science 32 Homework 5

1. Original tree

30

70

40

10

60

20

50

1. Inserting nodes

25

36

30

15

74

65

80

70

40

60

20

50

10



In-order traversal:

10, 15, 20, 25, 30, 36, 40, 50, 60, 65, 70, 74, 80

Post-order traversal:

15, 10, 25, 36, 30, 40, 20, 65, 74, 80, 70, 60, 50

Pre-order traversal:

50, 20, 10, 15, 40, 30, 25, 36, 60, 70, 65, 80, 74

1. After Deleting node 30

74

80

65

70

36

40

15

10

60

20

50

25

After deleting node 20 and 30

74

80

65

70

36

40

15

10

60

25

50

1. struct Node

{

int value;

Node\* parent;

Node\* leftChild;

Node\* rightChild;

};



void insert(Node\* ptr, Node\* rootptr)

{

if rootptr equals nullptr or tree is empty

set rootptr to node, and set node’s parent to rootptr

else if node is equal to rootptr

return

else if node’s value is less than rootptr’s value

if root has left child

recursively pass in left child as rootptr to insert function

else

set root’s left child to node, and node’s parent to root

else if node’s value is greater than rootptr’s value

if root has a right child

recursively pass in right child as rootptr to insert function

else

set root’s right child to node and node’s parent to root

}



0

6

3

8

4

2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 8 | 3 | 6 | 0 | 2 | 4 |



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 6 | 3 | 4 | 0 | 2 |

2. O(C + S)
3. O(logC + S)
4. O(logC + logS)
5. O(logS)
6. O(1)
7. O(logC + S)
8. O(S\*logS)
9. O(C\*logS)