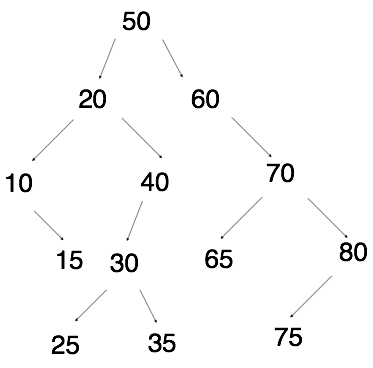
After insertion

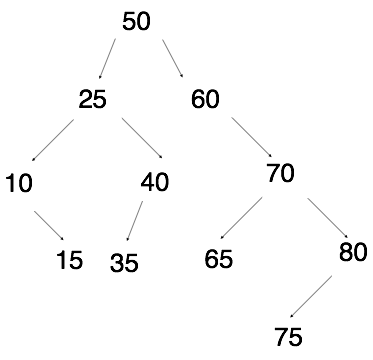


pre-order: 50- 20- 10- 15- 40- 30- 25- 35- 60- 70- 65- 80- 75

in-order: 10- 15- 20- 25- 30- 35- 40- 50- 60- 65- 70- 75- 80

post-order: 15- 10- 25- 35- 30- 40- 20- 65- 75- 80- 70- 60- 50

after deletion



2.

struct Node

{

int data;

Node \*right, \*left, \*parent;

};

pseudocode for insertion:

*input: a value V to insert*

*if the tree is empty*

*allocate a new node and put V into it*

*Point the root pointer to our new node*

*Start at the root of the tree*

*While we’re not done…*

*If V is equal to current node’s value, Done*

*If V is less than current node’s value*

*If there is a left child, then go left*

*Else, allocate a new node and put V into it, and set current*

*node’s left pointer to new node, set the new node’s*

*parent pointer to the current node, Done*

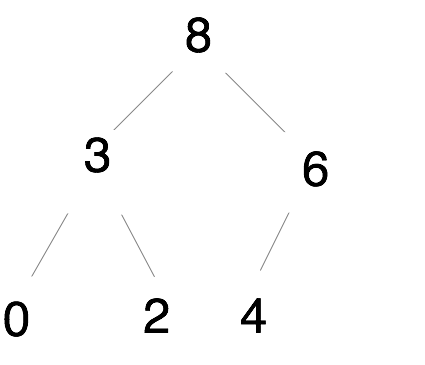
*if V is greater than current node’s value*

*if there is a right child, then go right*

*else, allocate a new node and put V into it, set current*

*node’s right pointer to new node, set the new node’s*

*parent pointer to the current node, Done*

3. 

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

0 1 2 3 4 5 …

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

0 1 2 3 4 …

4.

a. O (C+S)

b. O (log C + S)

c. O (log C + log S)

d. O (1 log S)

e. O (1)

f. O (log C + S)

g. O (S \* log S)

h. O (C log S)