

Tutorial 9: Binary Search Trees

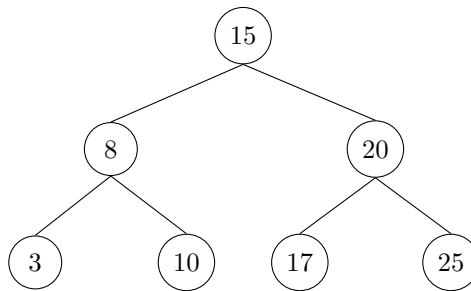
1st April, 2025

Binary Search Trees (BST)

A Binary Search Tree (BST) is a binary tree with the following properties:

- Each node has at most two children (left and right).
- For each node x :
 - All nodes in the left subtree have keys smaller than $x.key$.
 - All nodes in the right subtree have keys greater than $x.key$.

Example of a BST



BST Operations

Search/Query

The search operation recursively checks the tree by comparing the key k to the current node's key and deciding whether to move left or right.

Algorithm 1 BST-Search($root, k$)

```
1: if  $root = NIL$  or  $root.key = k$  then  
2:   return  $root$   
3: else if  $k < root.key$  then  
4:   return BST-Search( $root.left, k$ )  
5: else  
6:   return BST-Search( $root.right, k$ )  
7: end if
```

Minimum and Maximum

The minimum and maximum operations find the smallest and largest keys in the BST by traversing left or right, respectively.

Algorithm 2 BST-Minimum(*root*)

```
1: while root.left  $\neq$  NIL do
2:   root  $\leftarrow$  root.left
3: end while
4: return root
```

Algorithm 3 BST-Maximum(*root*)

```
1: while root.right  $\neq$  NIL do
2:   root  $\leftarrow$  root.right
3: end while
4: return root
```

Successor

The successor of a node x is the next larger node in the BST. If x has a right subtree, it is the minimum node in that subtree; otherwise, move up until finding a node that is the left child of its parent.

Algorithm 4 BST-Successor(x)

```
1: if x.right  $\neq$  NIL then
2:   return BST-Minimum(x.right)
3: end if
4: y  $\leftarrow$  x.parent
5: while y  $\neq$  NIL and x = y.right do
6:   x  $\leftarrow$  y
7:   y  $\leftarrow$  y.parent
8: end while
9: return y
```

Insertion

Insertion involves finding the correct leaf position according to BST ordering and then attaching the new node z .

Algorithm 5 BST-Insert(*root*, z)

```
1: y  $\leftarrow$  NIL, x  $\leftarrow$  root
2: while x  $\neq$  NIL do
3:   y  $\leftarrow$  x
4:   if z.key < x.key then
5:     x  $\leftarrow$  x.left
6:   else
7:     x  $\leftarrow$  x.right
8:   end if
9: end while
10: z.parent  $\leftarrow$  y
11: if y = NIL then
12:   root  $\leftarrow$  z
13: else if z.key < y.key then
14:   y.left  $\leftarrow$  z
15: else
16:   y.right  $\leftarrow$  z
17: end if
```

Deletion

Deletion has three scenarios:

1. **No child:** Directly remove node z .
2. **One child:** Replace node z with its child.
3. **Two children:** Replace z 's key with successor's key, then recursively delete successor.

Algorithm 6 BST-Delete($root, z$)

```
1: if  $z.left = NIL$  then
2:   Transplant( $root, z, z.right$ )
3: else if  $z.right = NIL$  then
4:   Transplant( $root, z, z.left$ )
5: else
6:    $y \leftarrow \text{BST-Minimum}(z.right)$ 
7:   if  $y.parent \neq z$  then
8:     Transplant( $root, y, y.right$ )
9:      $y.right \leftarrow z.right$ 
10:     $y.right.parent \leftarrow y$ 
11:   end if
12:   Transplant( $root, z, y$ )
13:    $y.left \leftarrow z.left$ 
14:    $y.left.parent \leftarrow y$ 
15: end if
```

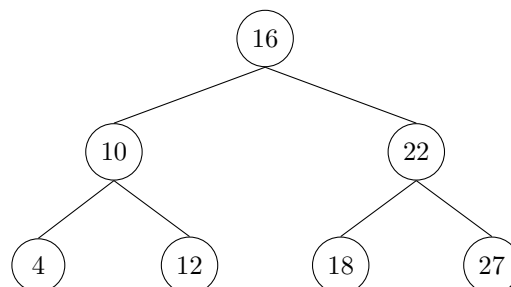
Helper method for deletion The helper method **Transplant** replaces one subtree (node u) with another subtree (node v) as the child of u 's parent.

Algorithm 7 Transplant($root, u, v$)

```
1: if  $u.parent = NIL$  then
2:    $root \leftarrow v$ 
3: else if  $u = u.parent.left$  then
4:    $u.parent.left \leftarrow v$ 
5: else
6:    $u.parent.right \leftarrow v$ 
7: end if
8: if  $v \neq NIL$  then
9:    $v.parent \leftarrow u.parent$ 
10: end if
```

Solved Example

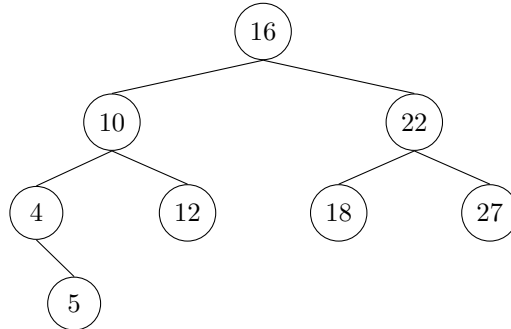
Consider the initial BST below:



Insertion of keys: 5, 15

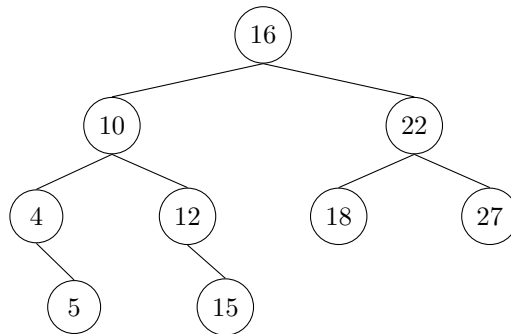
Step 1: Insert key 5

Key 5 is smaller than 16 (move left), smaller than 10 (move left), larger than 4 (move right). We insert 5 as the right child of node 4.



Step 2: Insert key 15

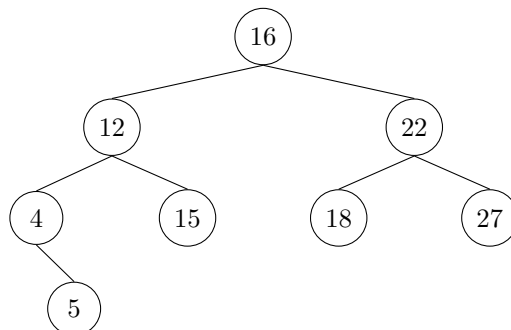
Key 15 is smaller than 16 (move left), larger than 10 (move right), larger than 12 (move right). Insert 15 as the right child of node 12.



Deletion of keys: 10, 16

Step 1: Delete key 10

Node 10 has two children. Find successor (12), replace key 10 with 12, and delete node 12 (which has one child: 15). Node 15 takes place of 12.



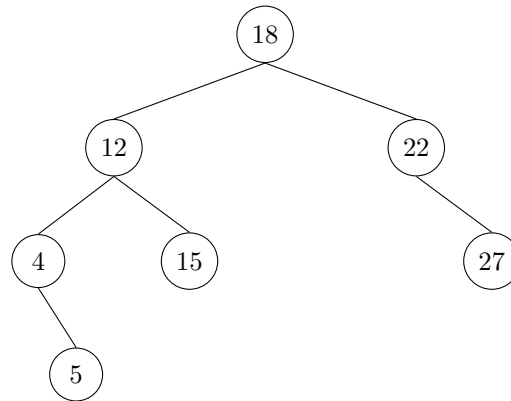
Step 2: Delete key 16

Node 16 has two children. Find its successor, node 18 (minimum of the right subtree rooted at node 22). Since node 18 is not a direct child of node 16, we must perform the following steps clearly:

- First, transplant node 18 with its right subtree (node 18 has no left subtree), replacing node 18 with NIL as its parent's (node 22) left child.

- Then replace node 16 with node 18 at the root position.
- Set node 18's left child to node 12, and node 18's right child to node 22. Update all parent pointers accordingly.

After carefully performing these transplant operations, the BST becomes:



Finding Successors

Successor of node 12: Node 12 has a right subtree. Successor is minimum of right subtree, node 15.

Thus, we have:

$$\text{Successor}(12) = 15$$

Note to students: Always clearly illustrate each step similarly to this example. Include explanations for every insertion, deletion, and successor operation when solving problems.