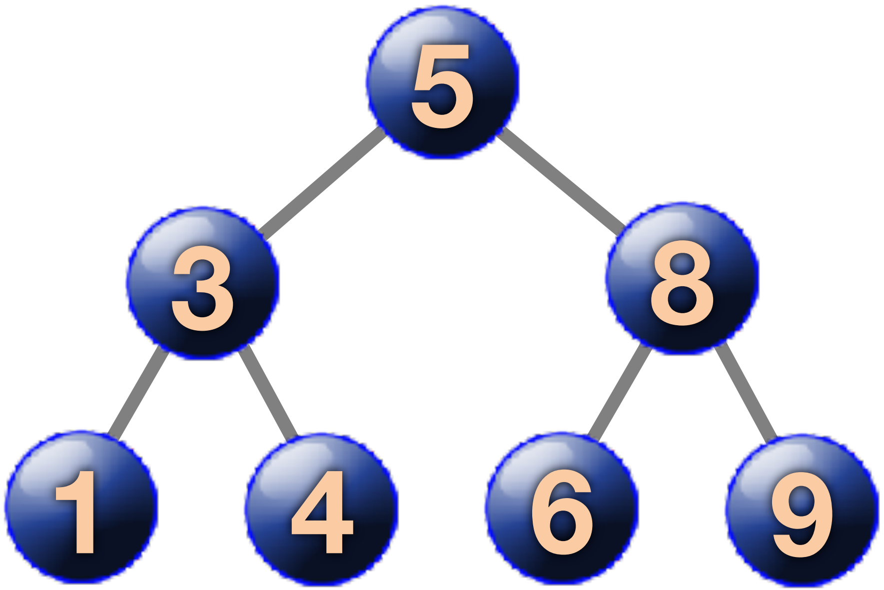
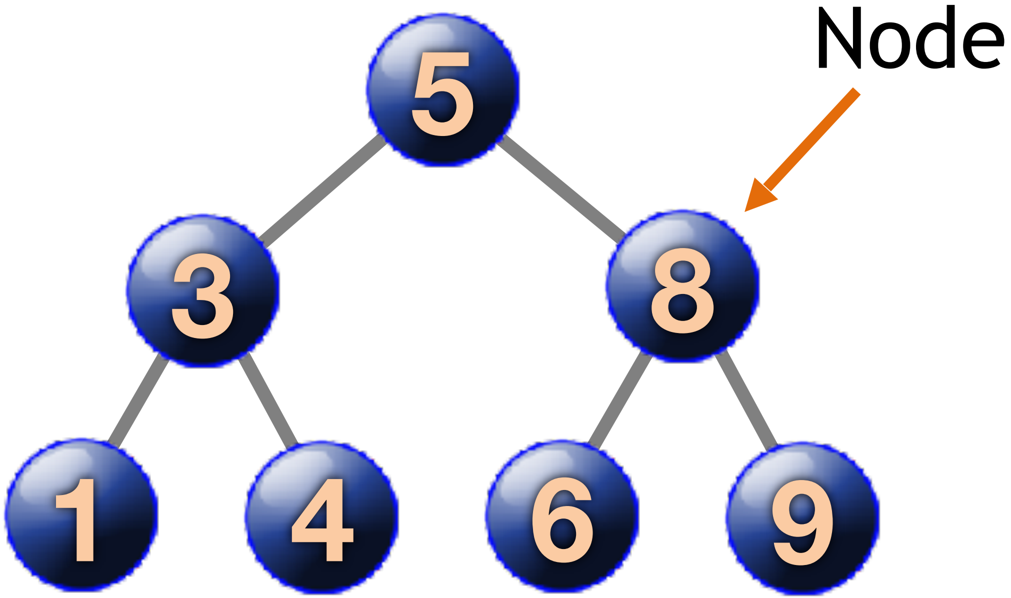
Binary Search Trees

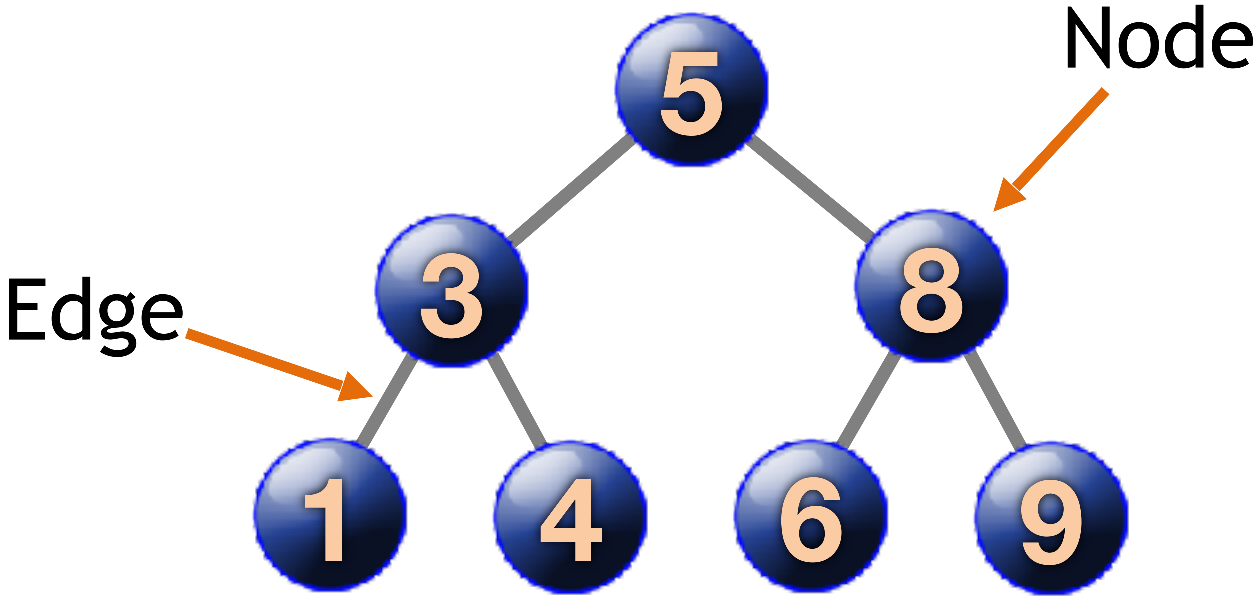
Tree



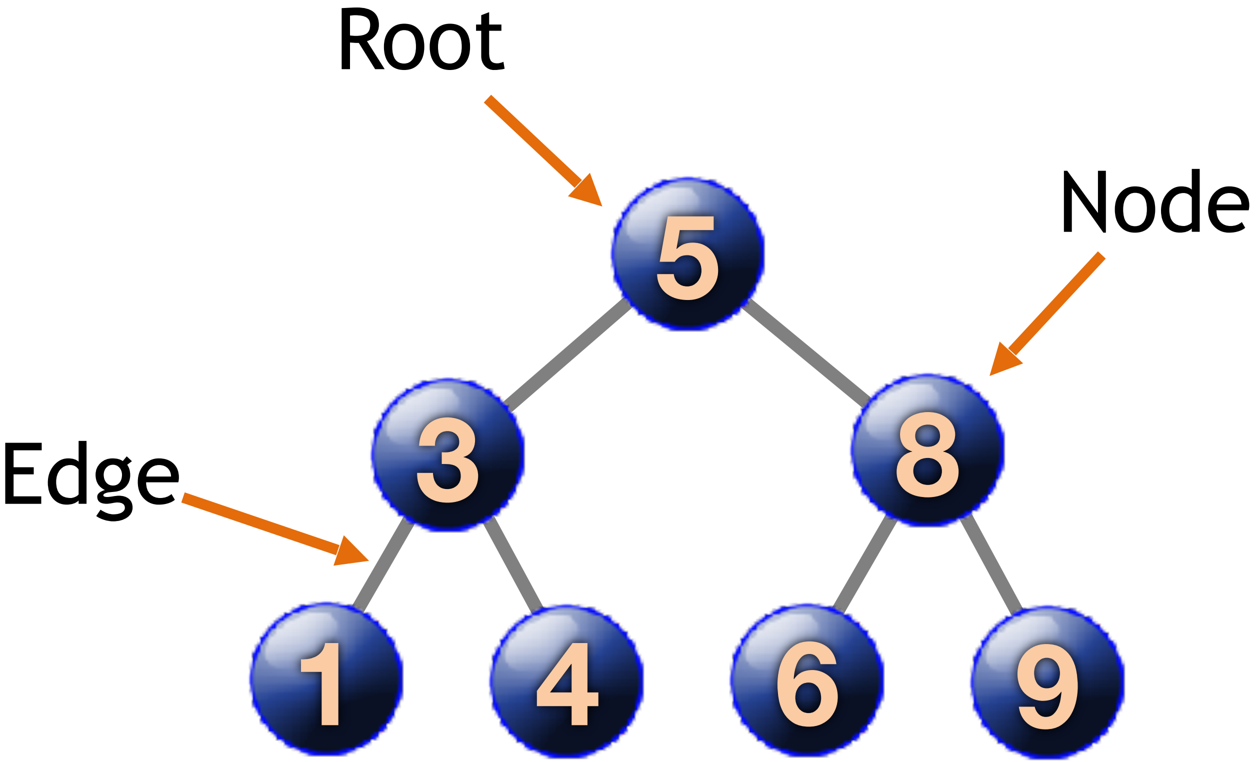
Tree

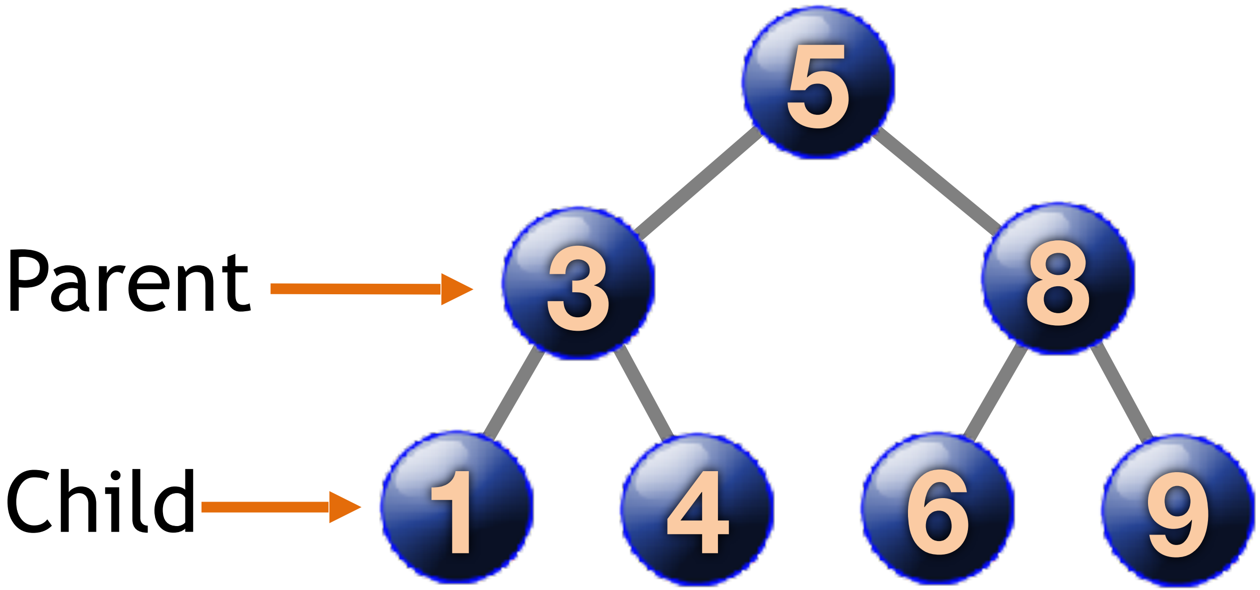


Tree

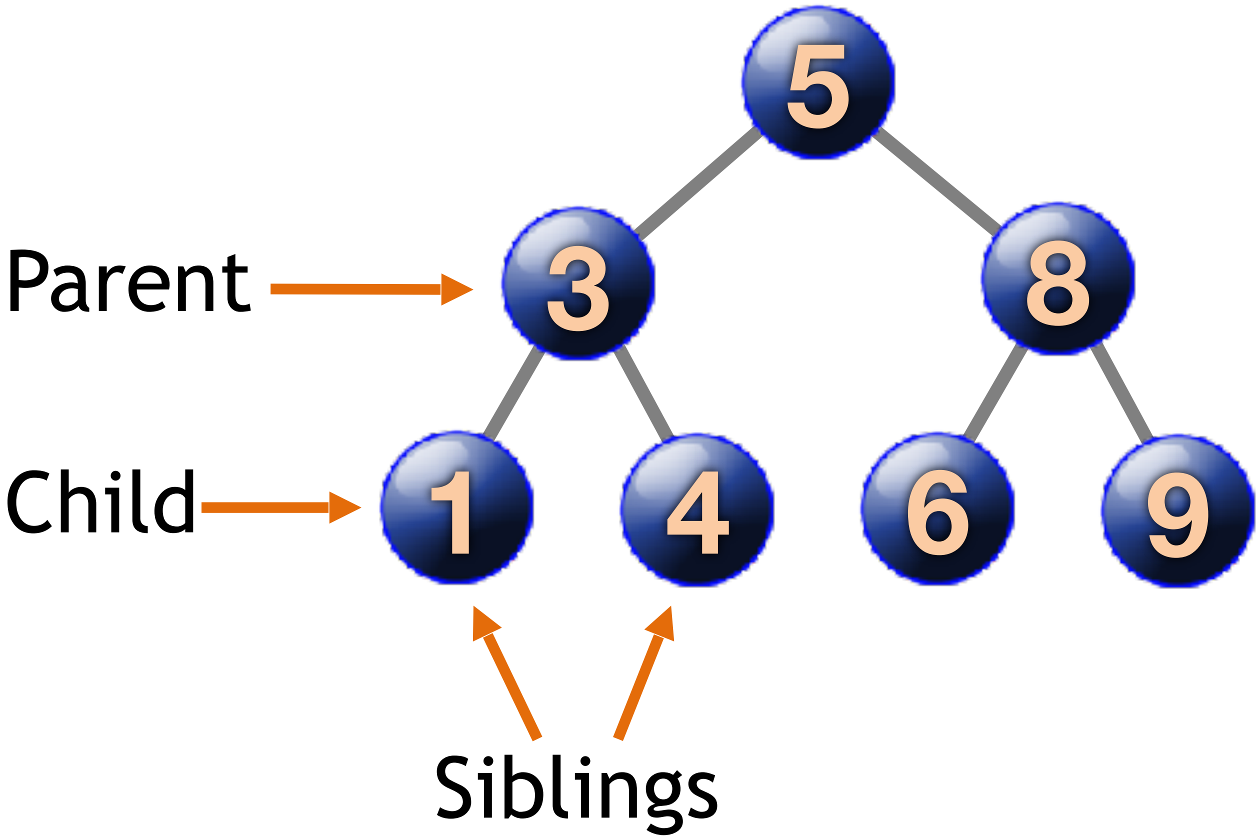


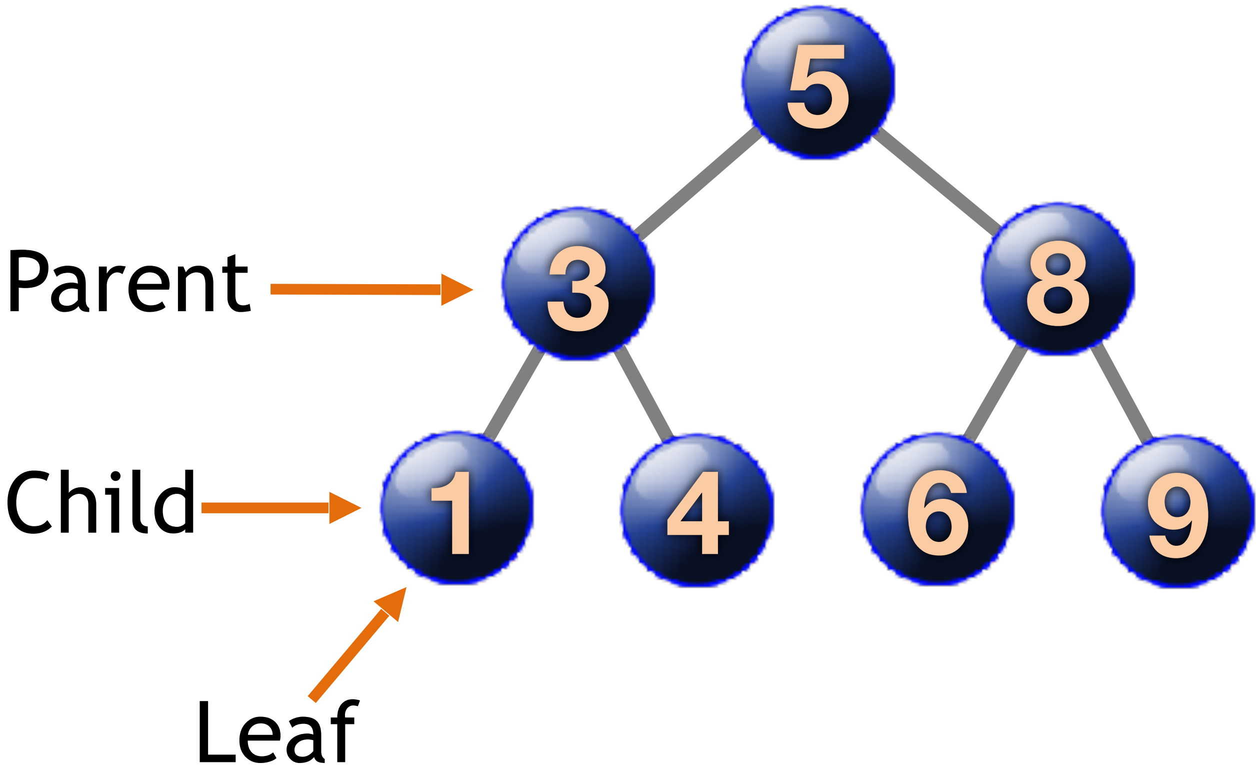
Tree



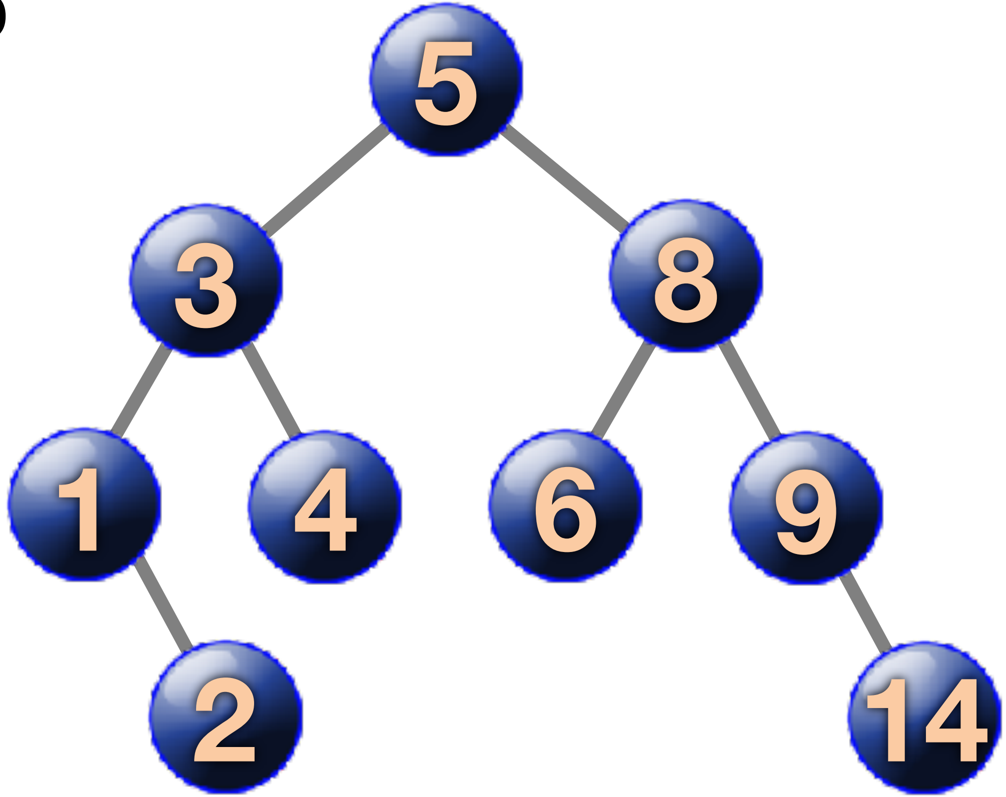


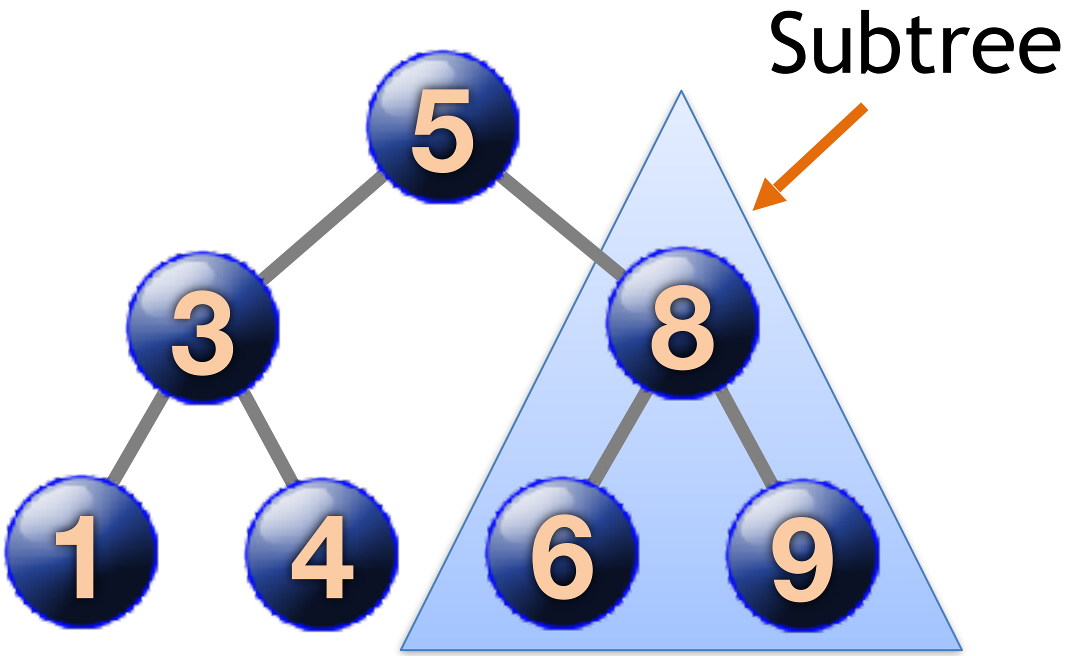
Parent
Child
5
3
8
1
4
6
9

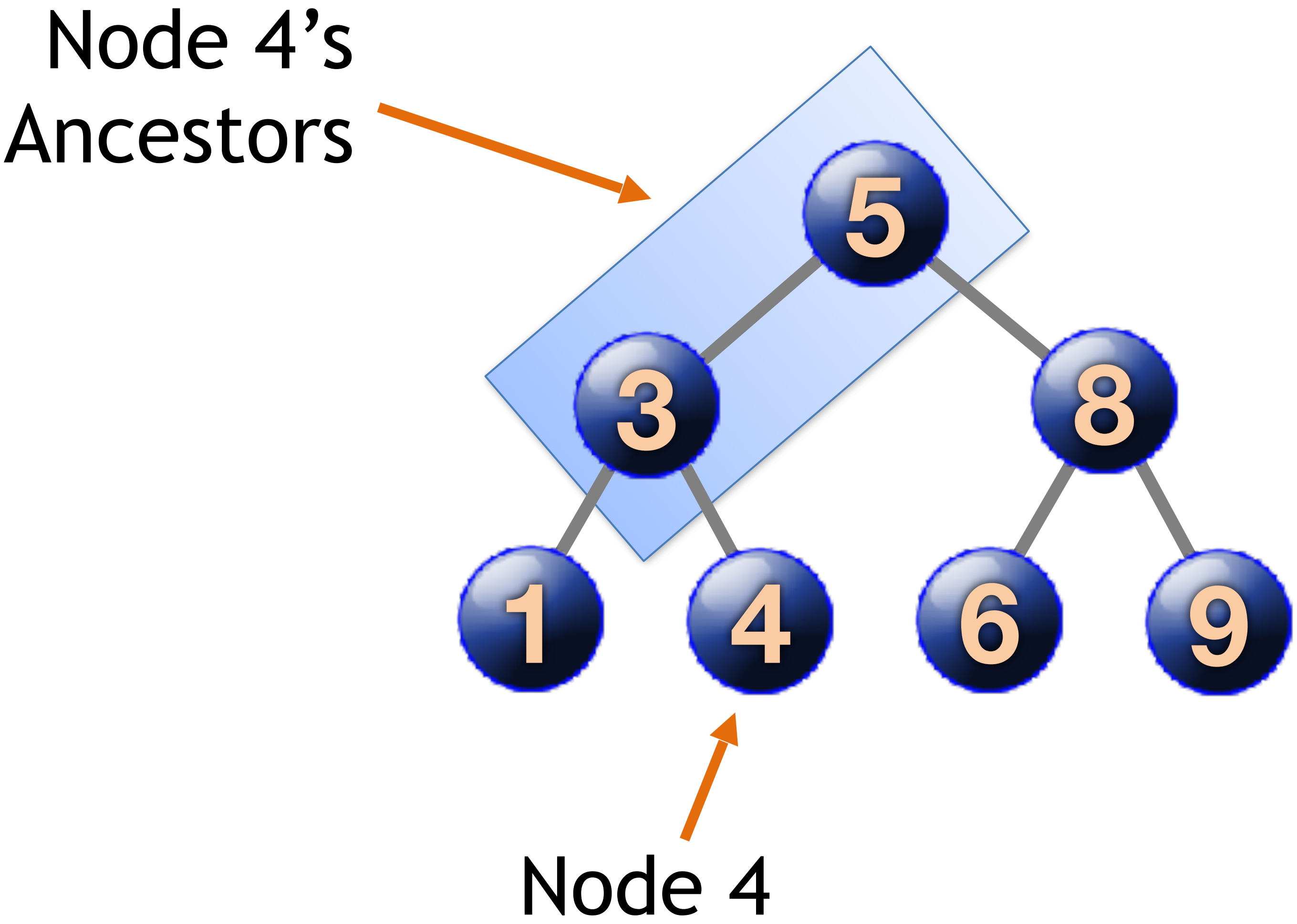

Parent
Child
5
3
8
1
4
6
9
Siblings


Parent
Child
Leaf
5
3
8
1
4
6
9

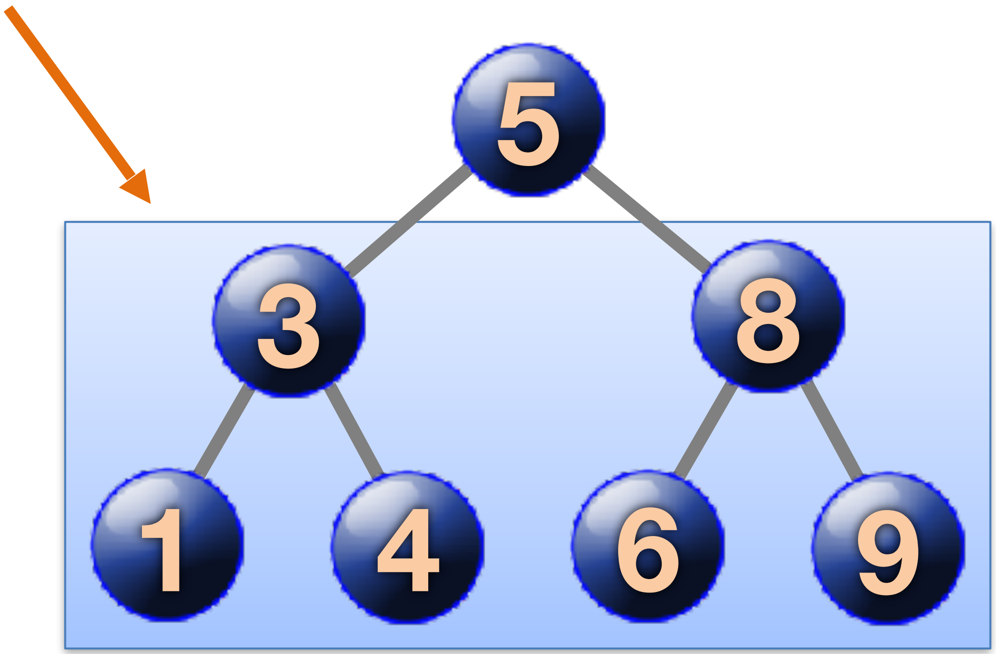
Binary Tree - each node can have up to 2 child nodes.




Subtree
5
3
8
1
4
6
9

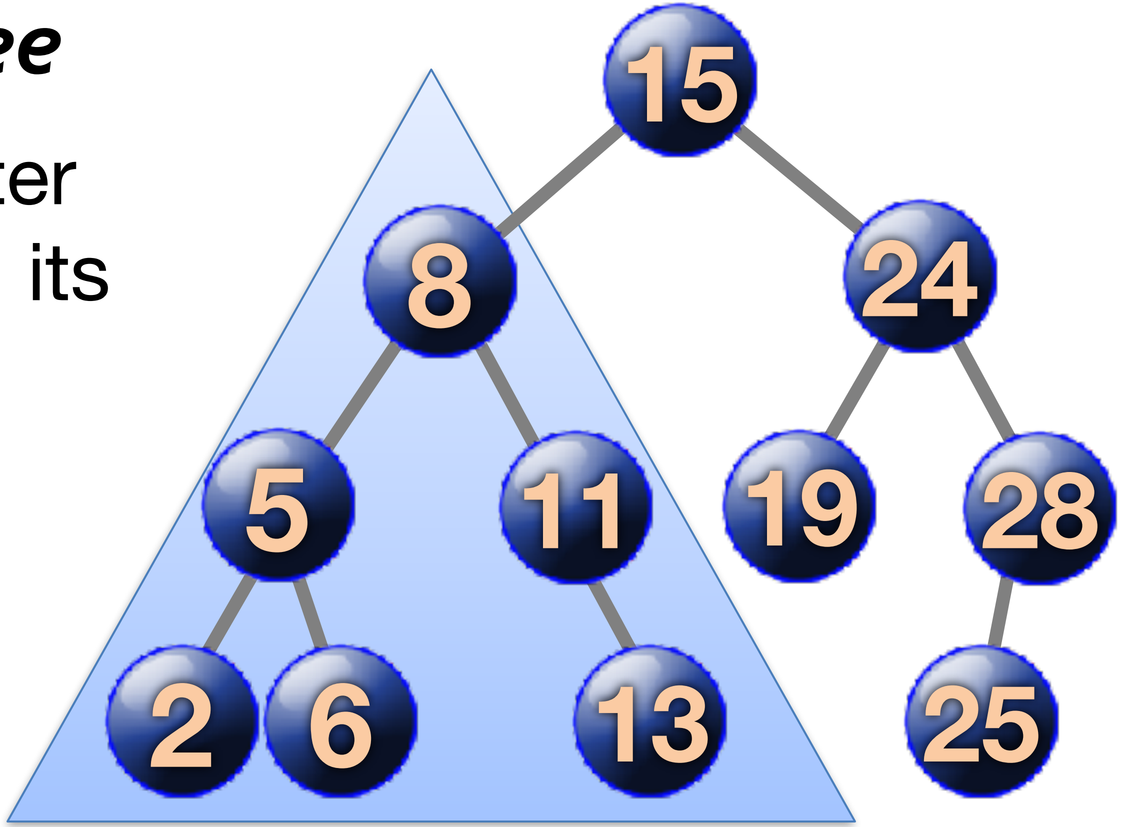

Node 4's
Ancestors
5
3
8
1
4
6
9
Node 4

Node 5's Descendants



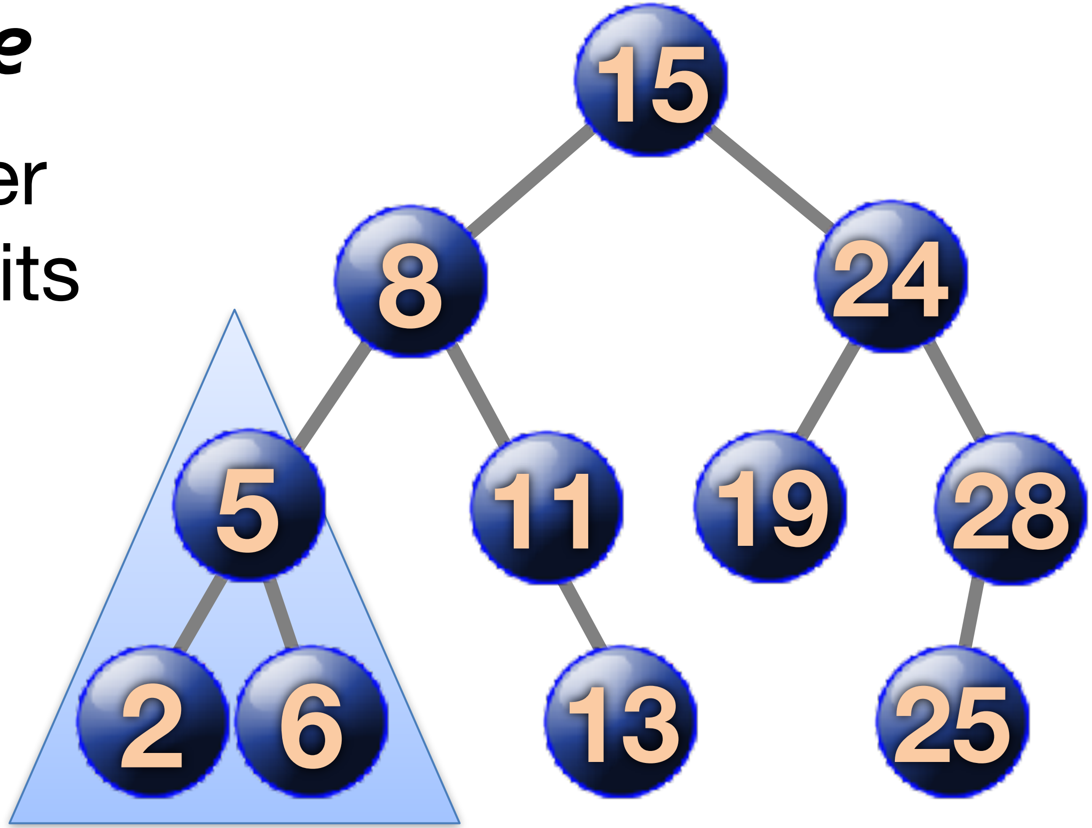
Binary Search Tree

- each node is greater than every node in its left subtree



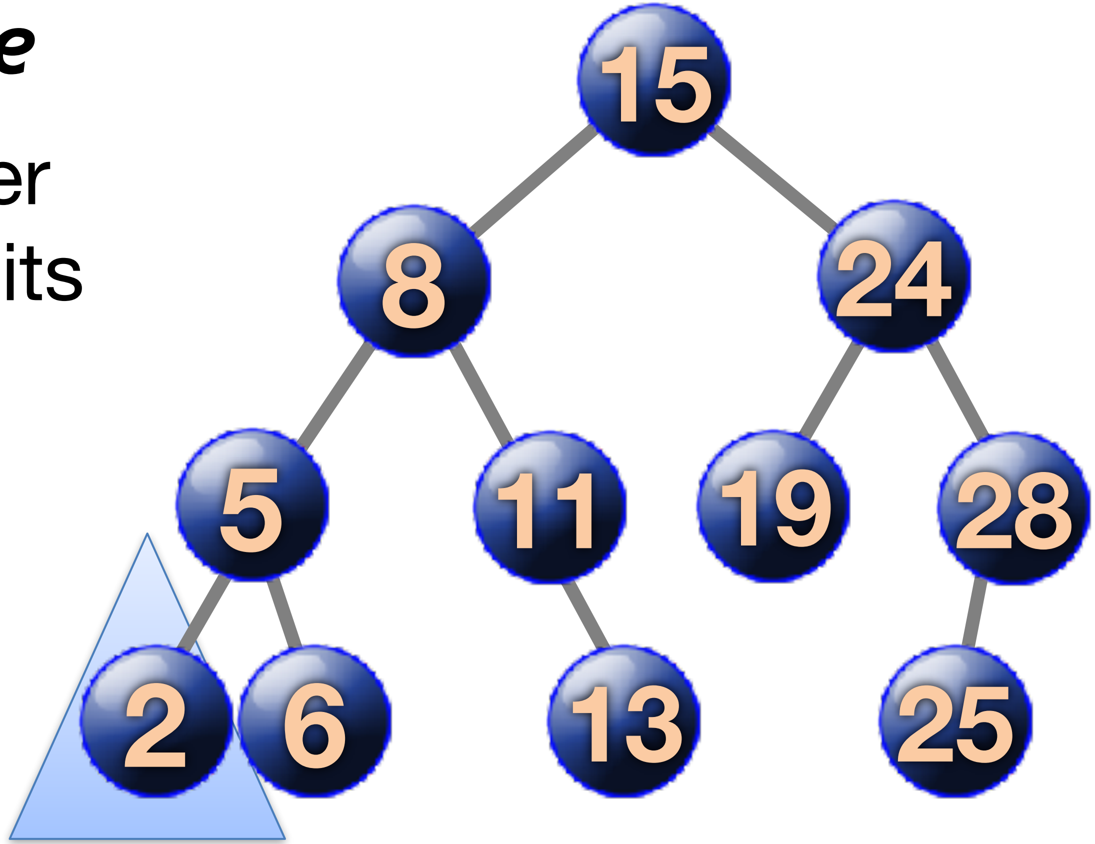
Binary Search Tree

- each node is greater than every node in its left subtree



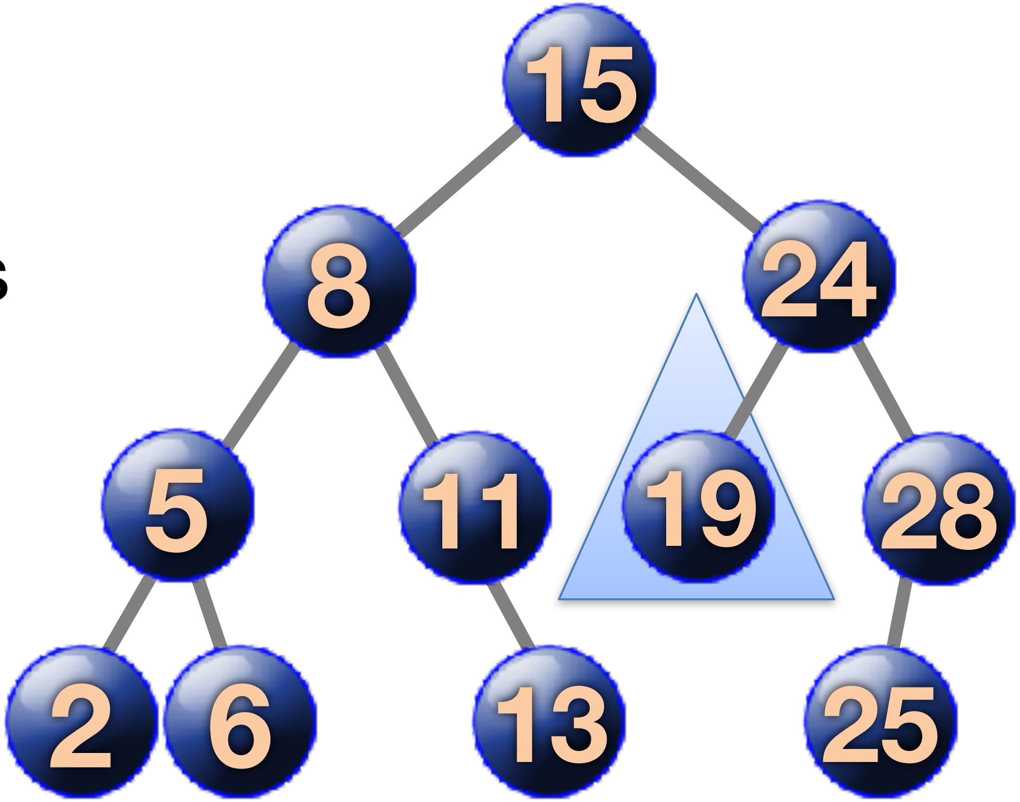
Binary Search Tree

- each node is greater than every node in its left subtree



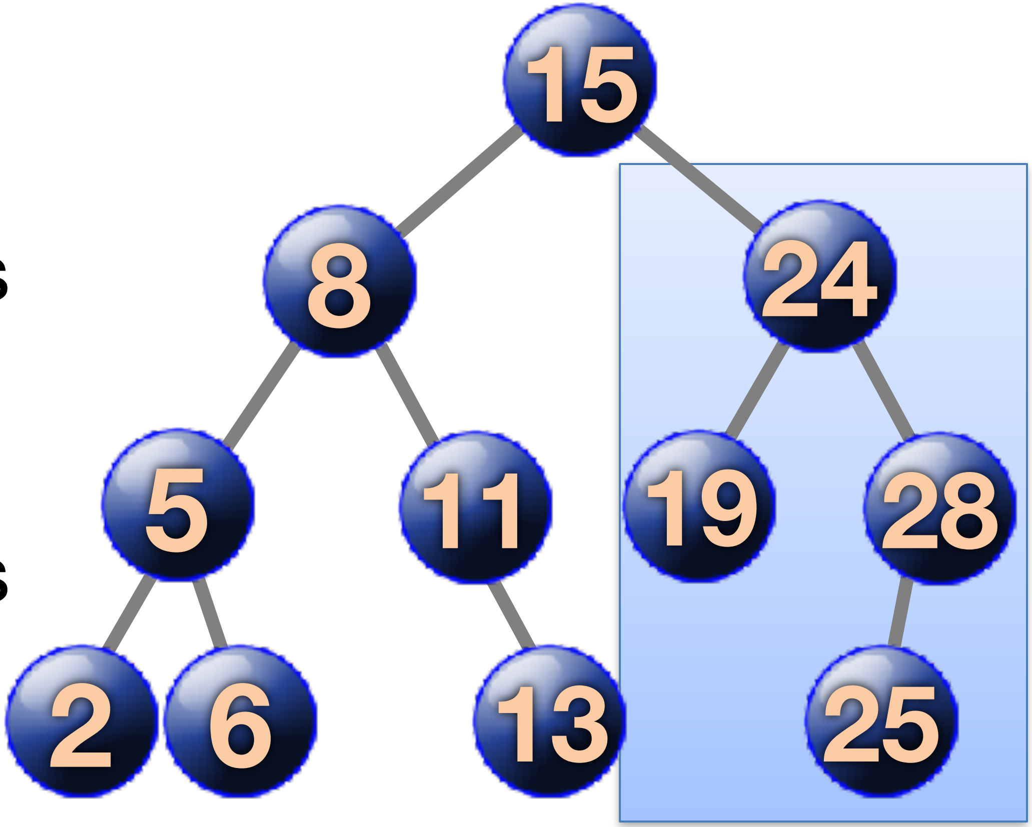
Binary Search Tree

- each node is greater than every node in its left subtree



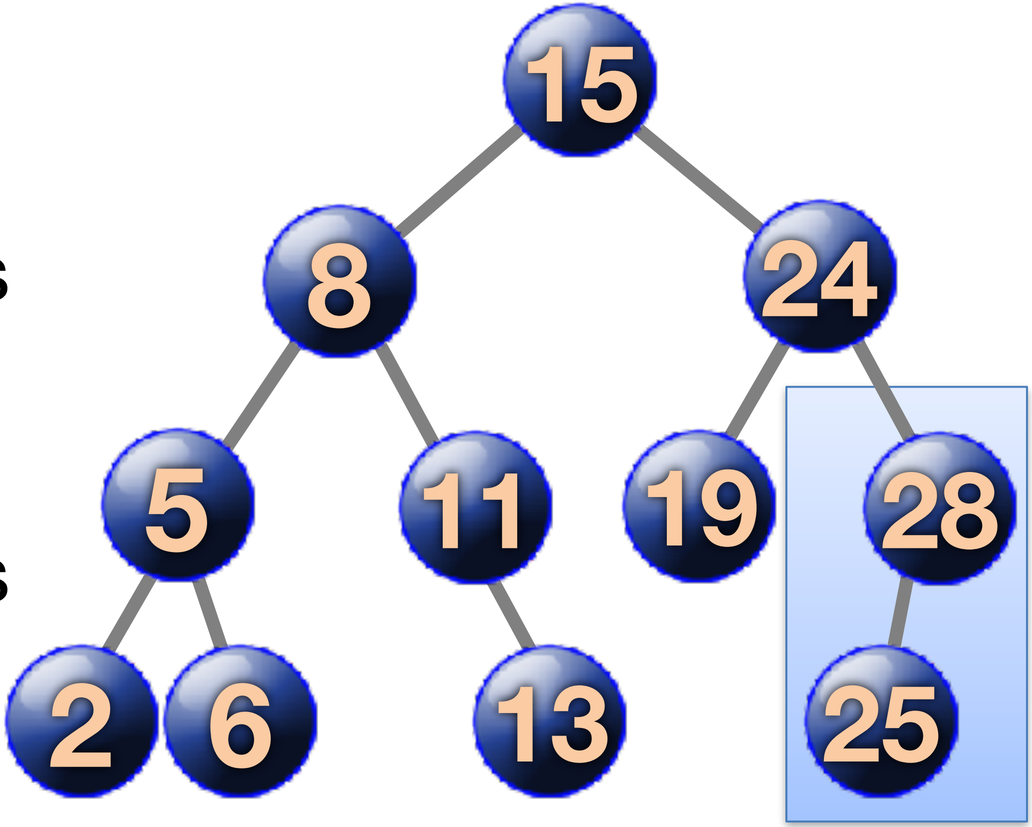
Binary Search Tree

- each node is greater than every node in its left subtree
- each node is less than every node in its right subtree



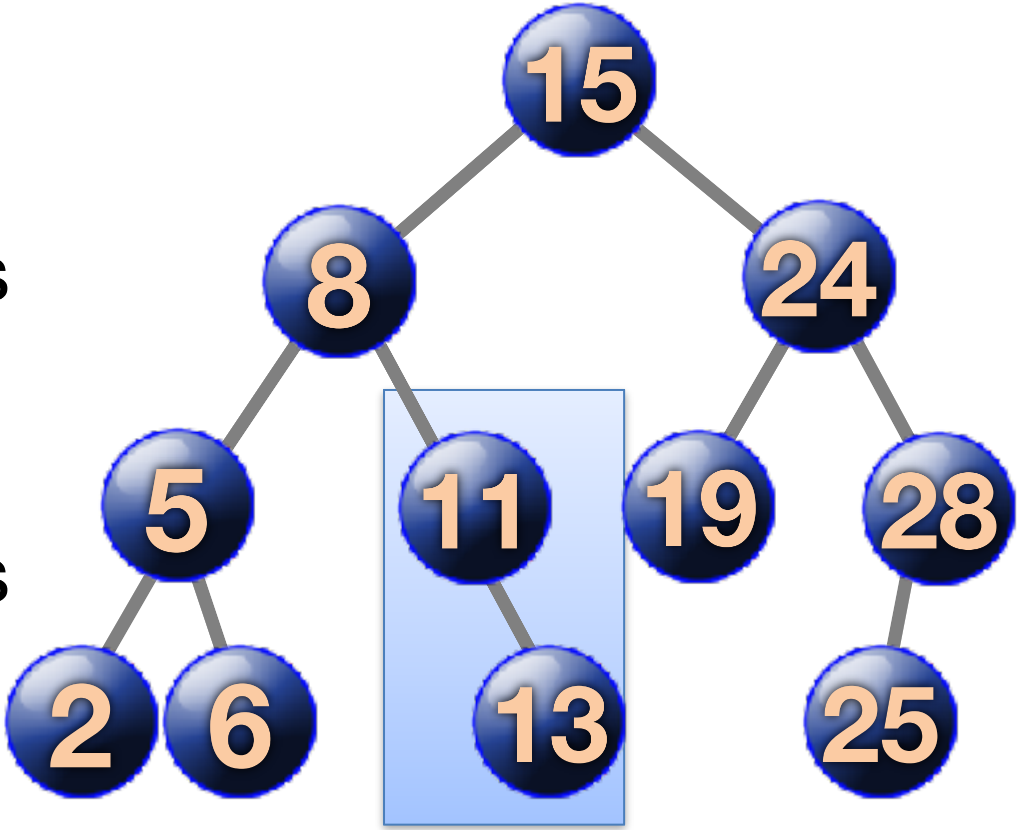
Binary Search Tree

- each node is greater than every node in its left subtree
- each node is less than every node in its right subtree



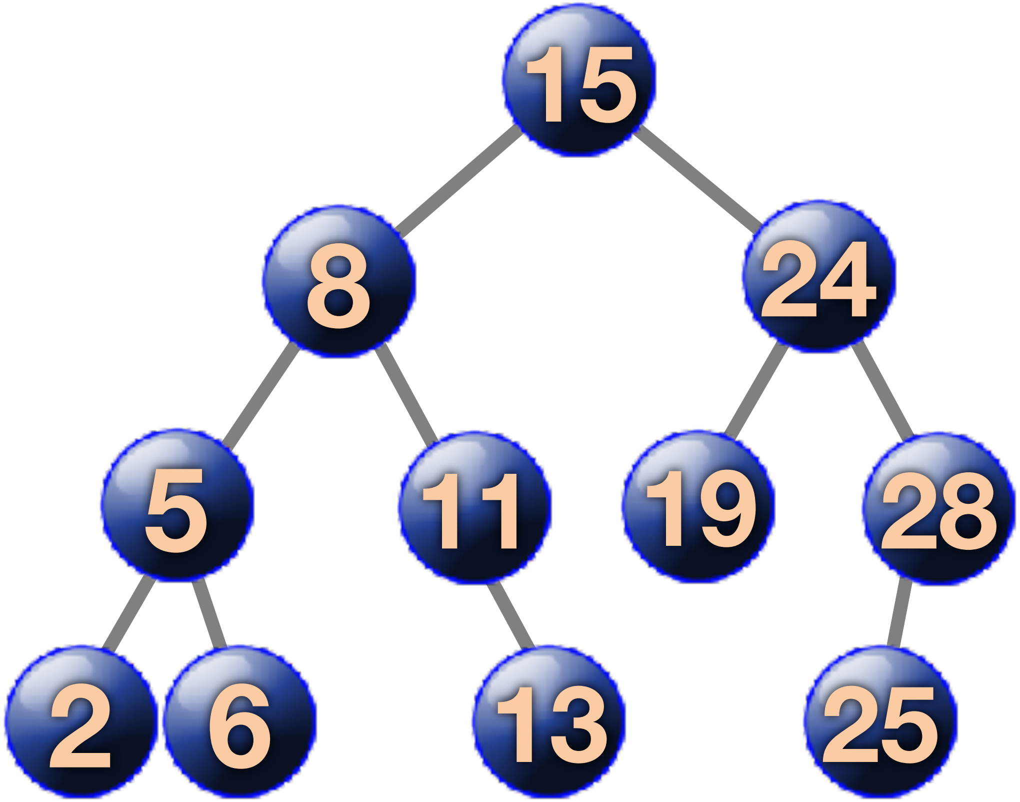
Binary Search Tree

- each node is greater than every node in its left subtree
- each node is less than every node in its right subtree



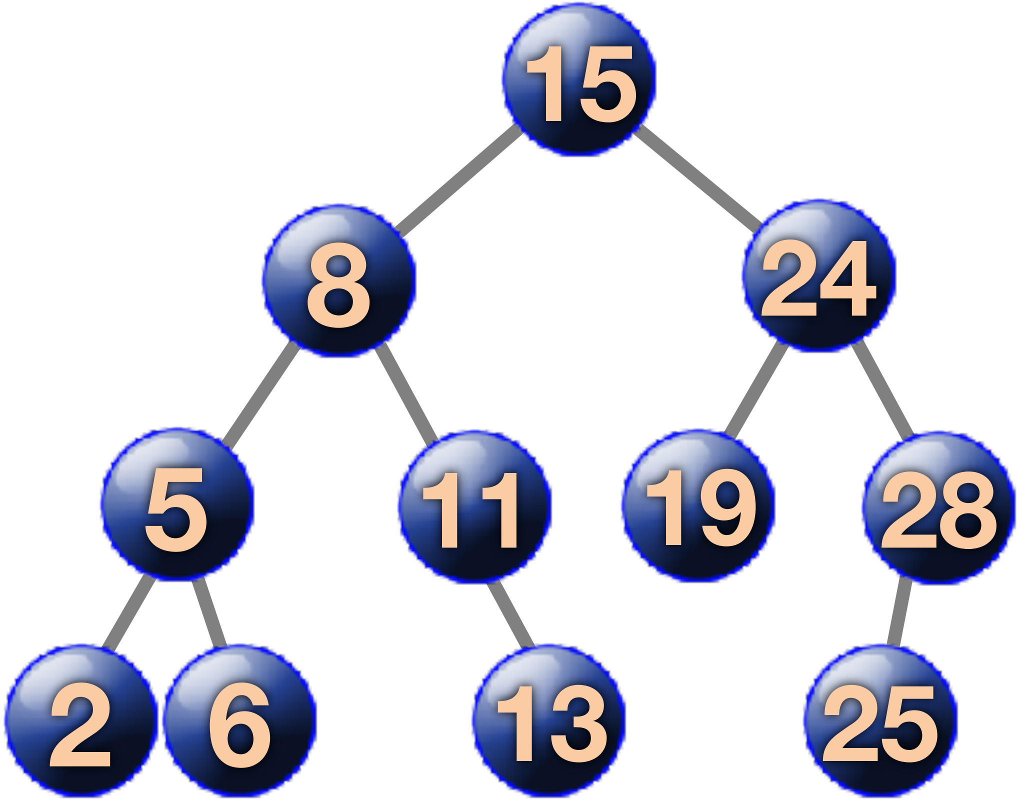
BST Operations

- Insert
- Find
- Delete
- Get_size
- Traversals



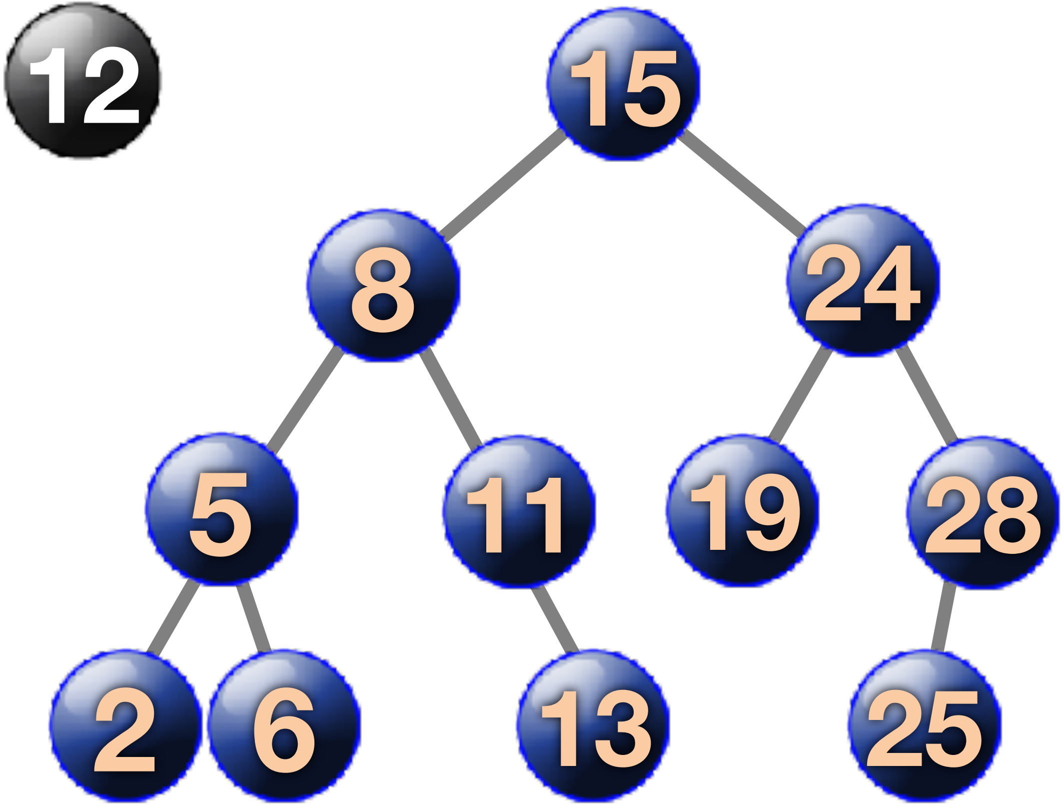
BST Insert

- Start at root
- Always insert as a leaf



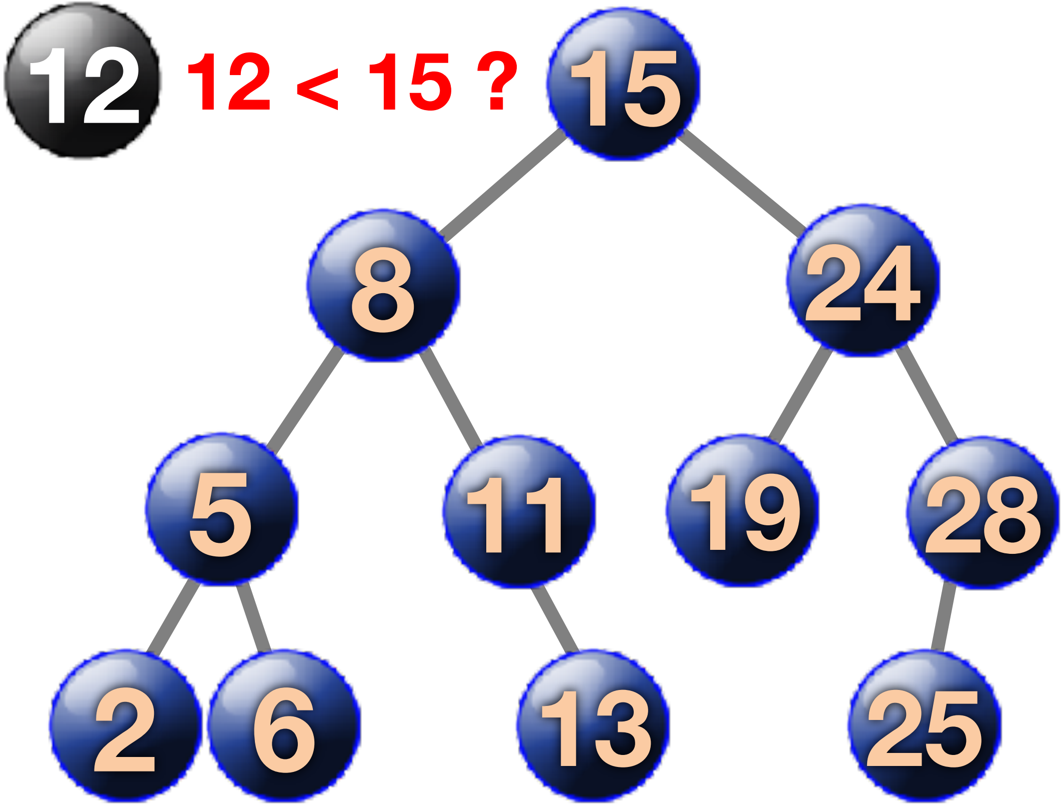
BST Insert

- Start at root
- Always insert as a leaf



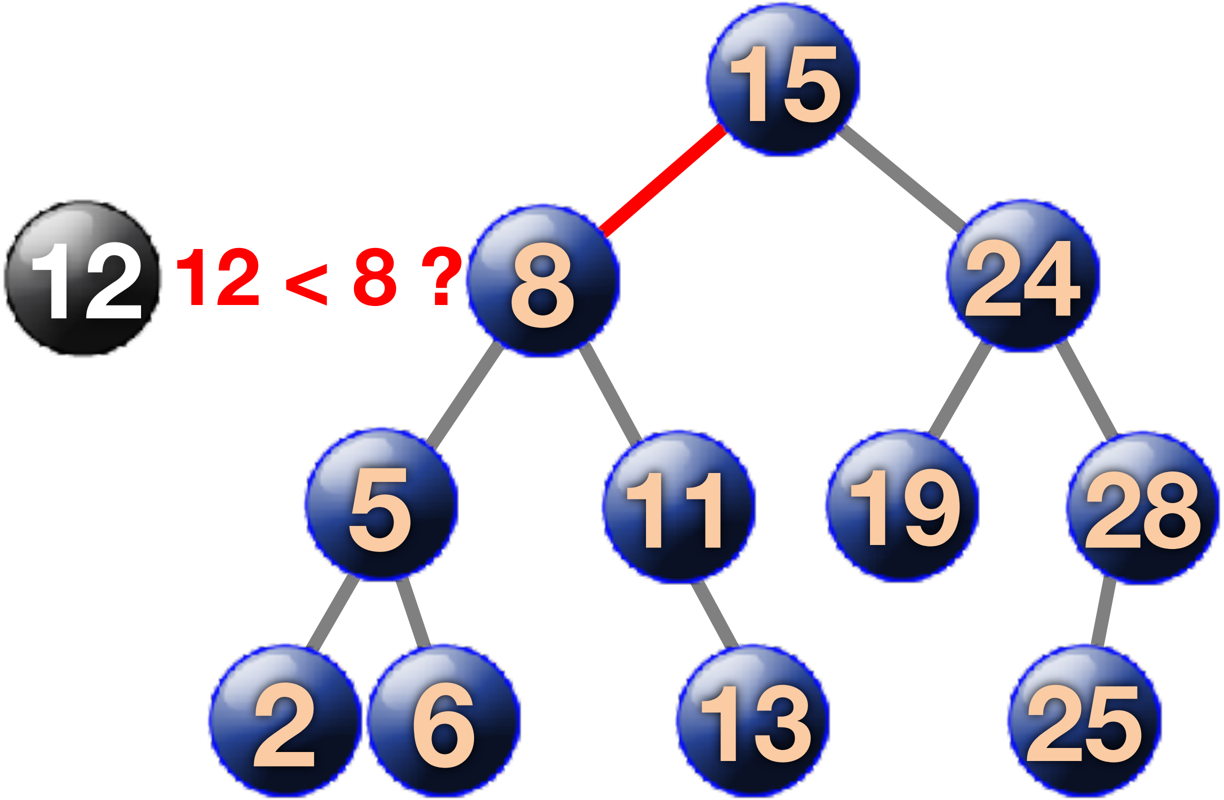
BST Insert

- Start at root
- Always insert as a leaf



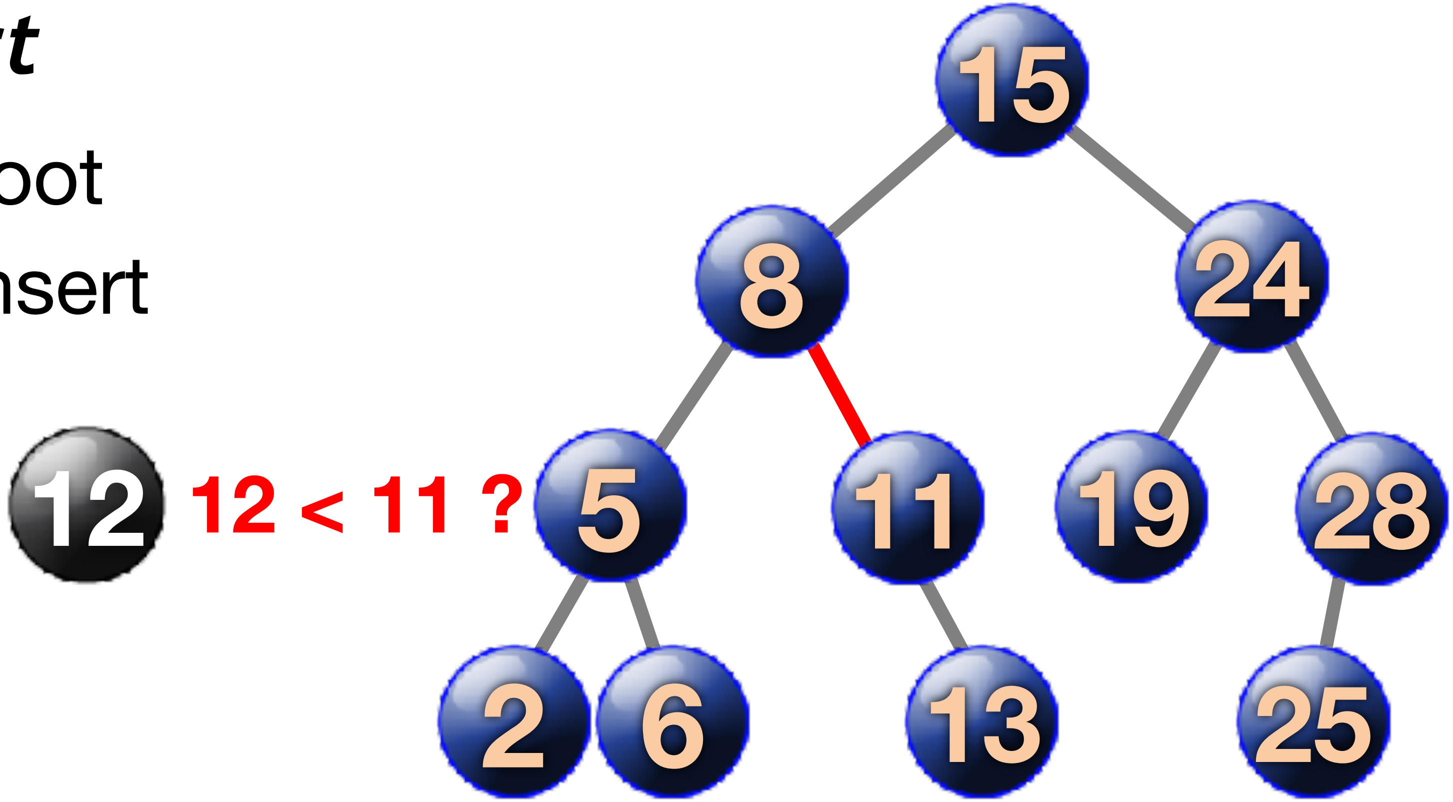
BST Insert

- Start at root
- Always insert as a leaf



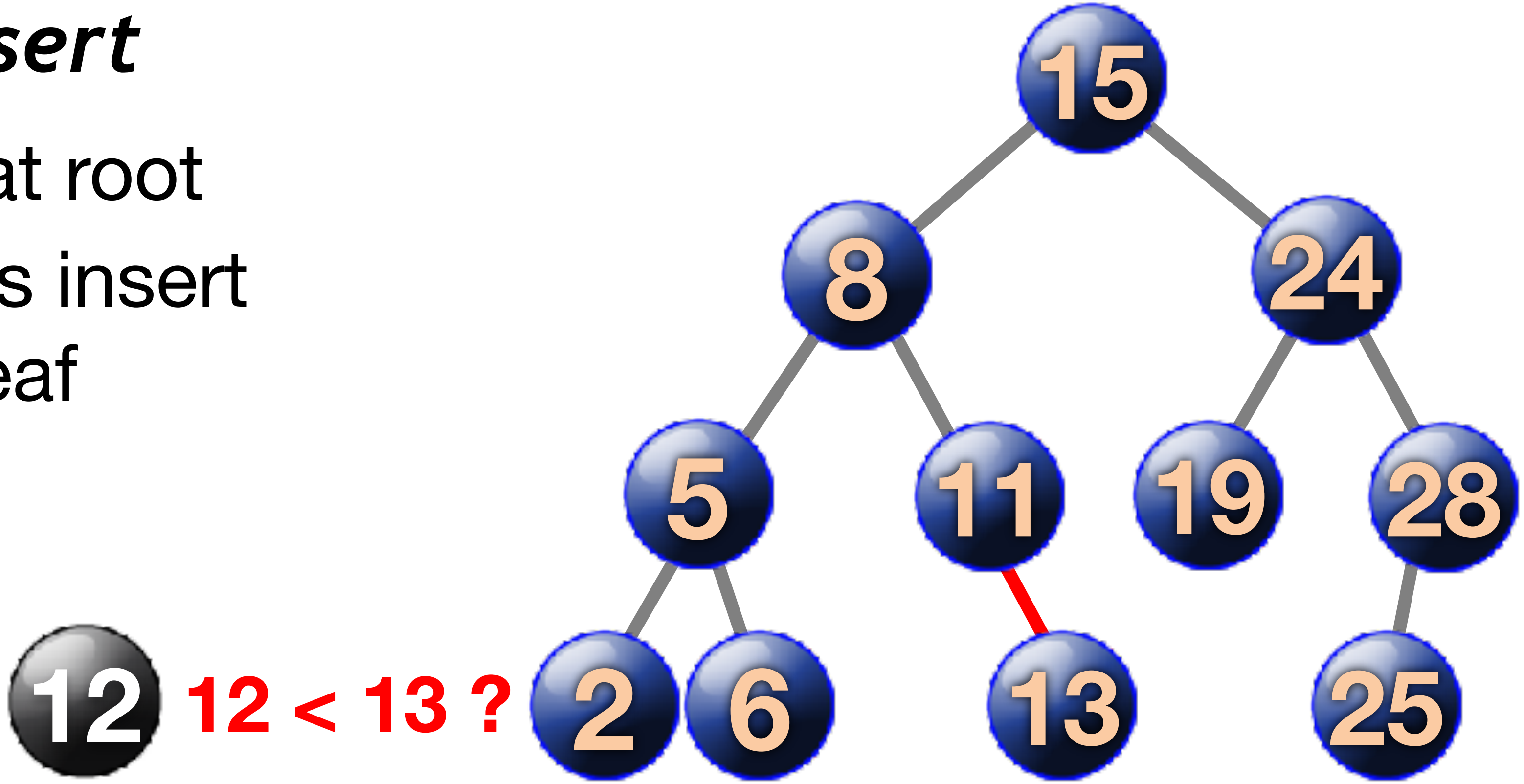
BST Insert

- Start at root
- Always insert as a leaf



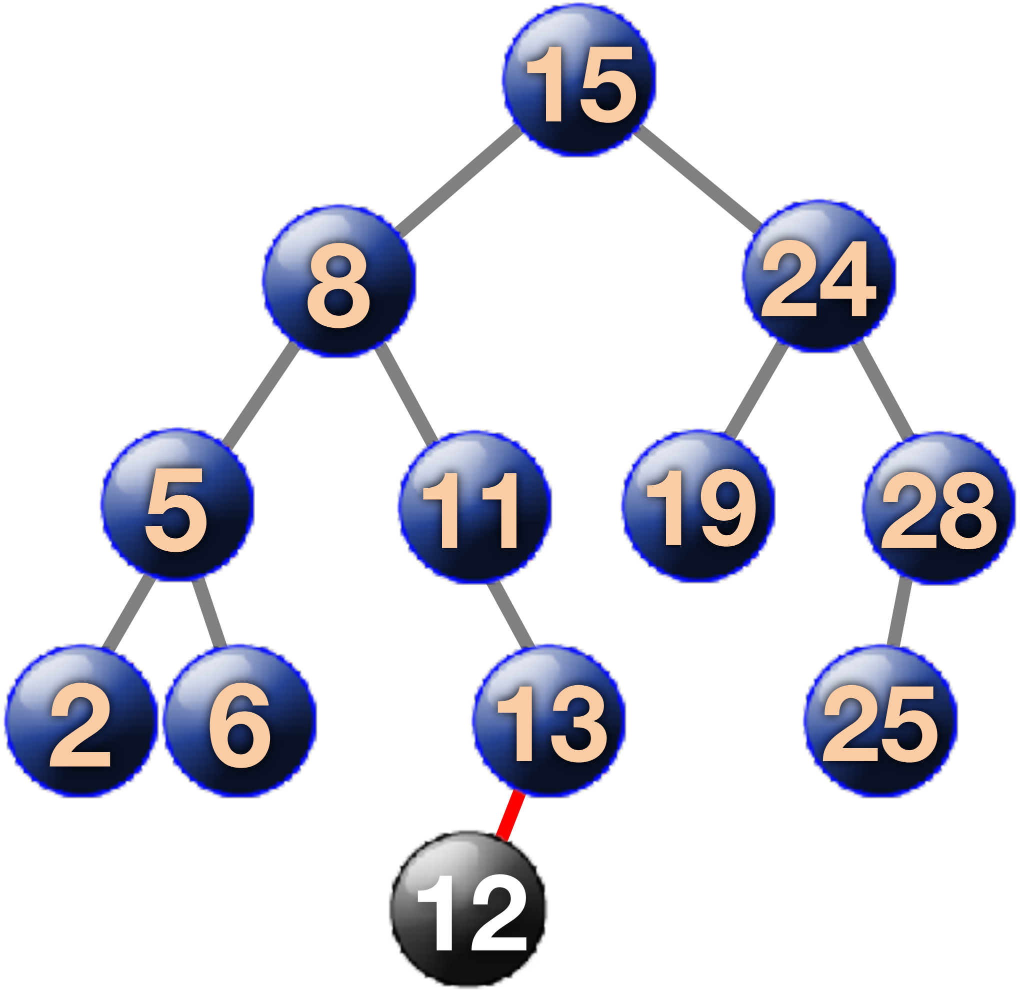
BST Insert

- Start at root
- Always insert as a leaf



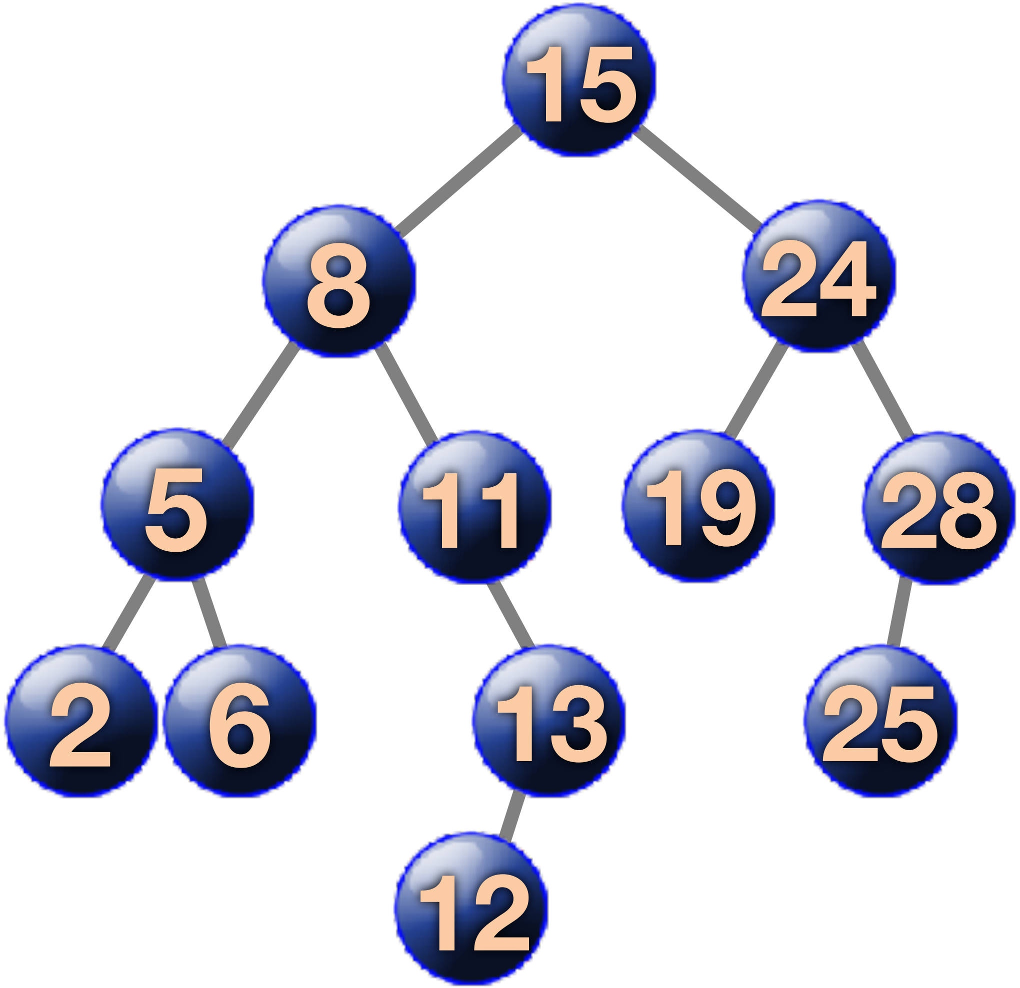
BST Insert

- Start at root
- Always insert as a leaf



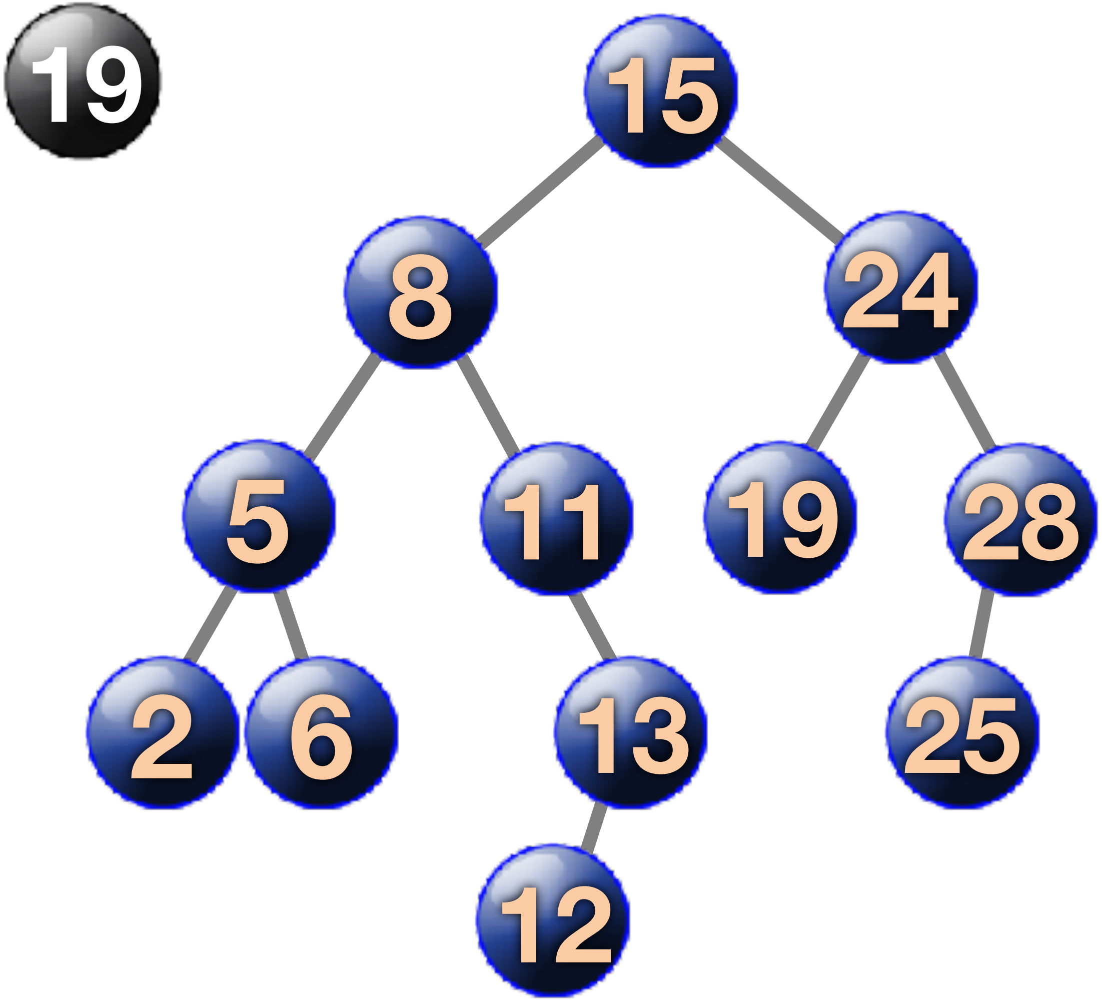
BST Find

- Start at root



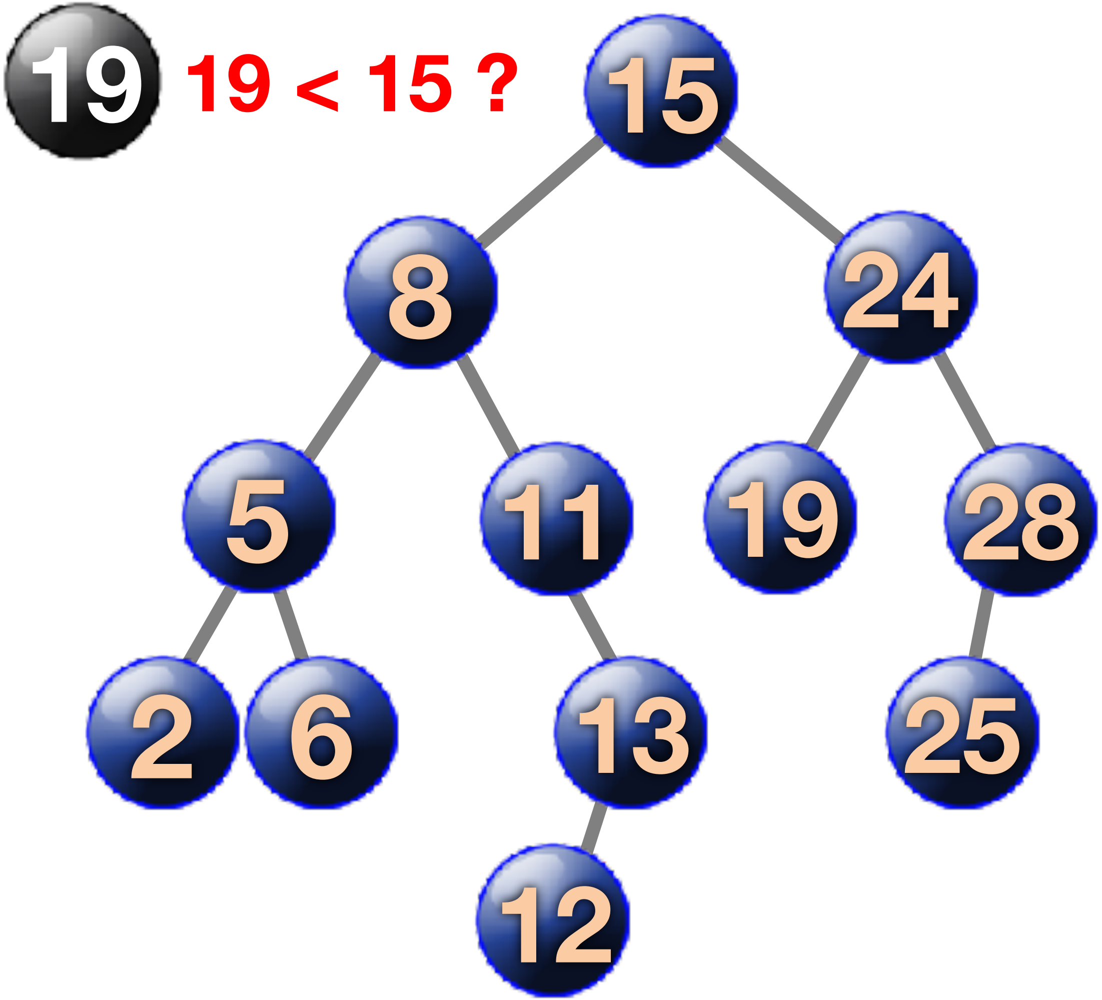
BST Find

- Start at root



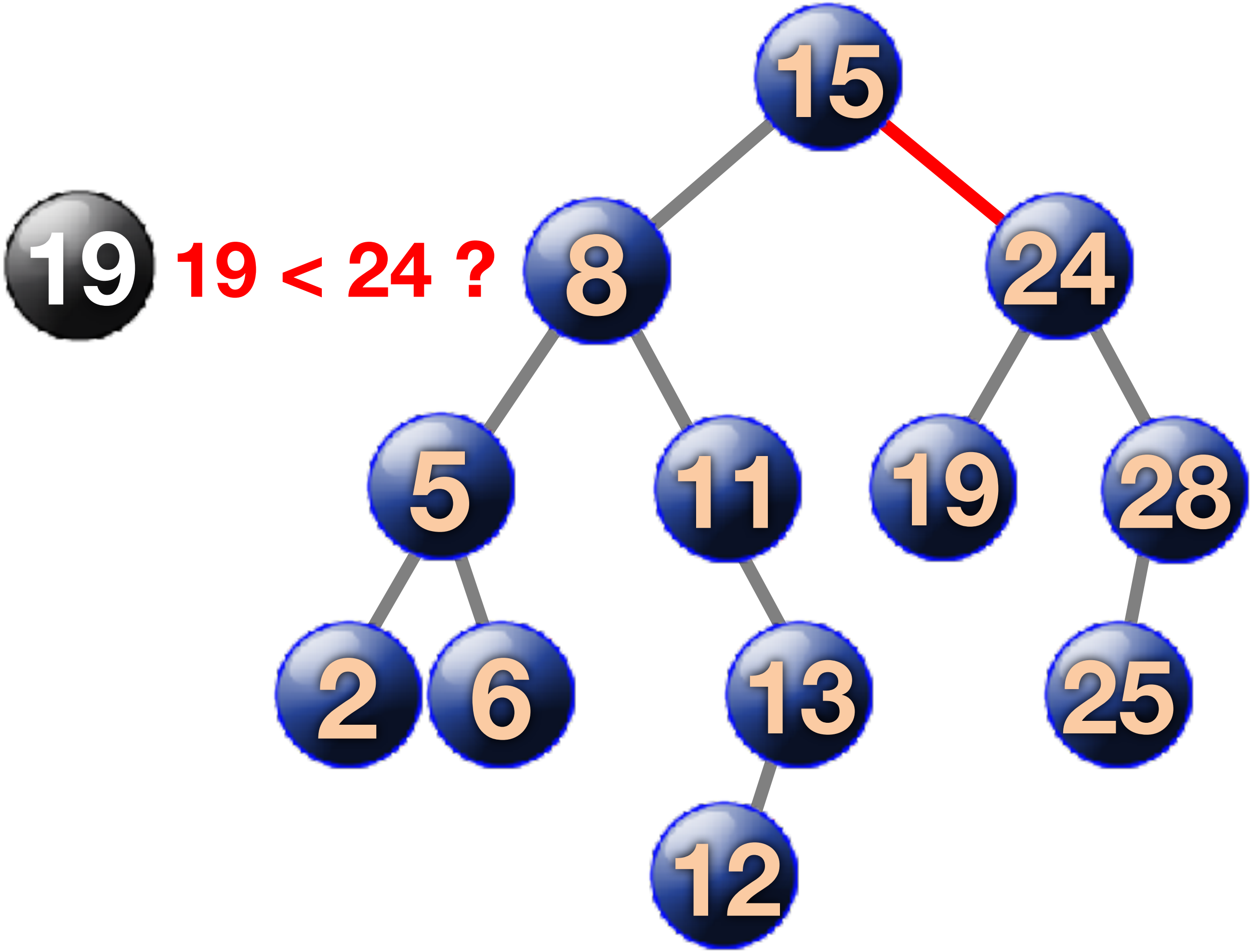
BST Find

- Start at root



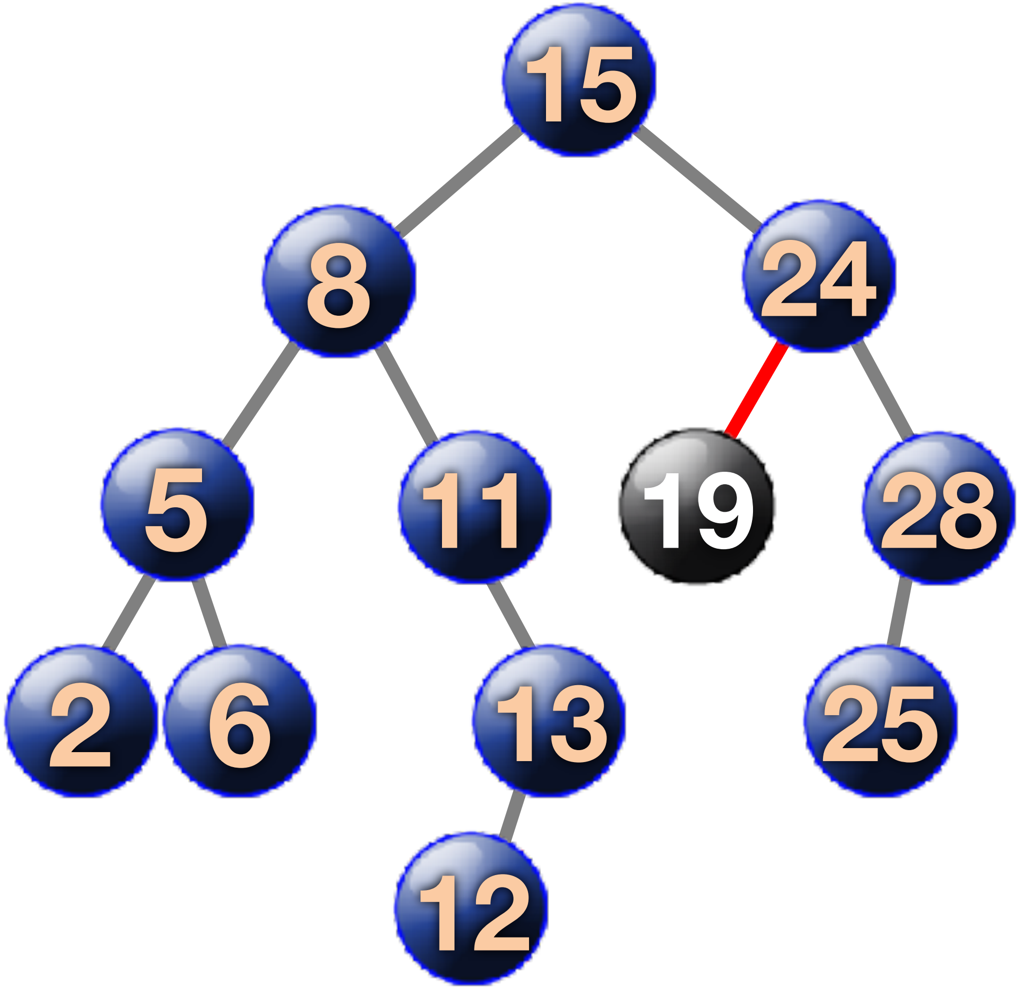
BST Find

- Start at root



BST Find

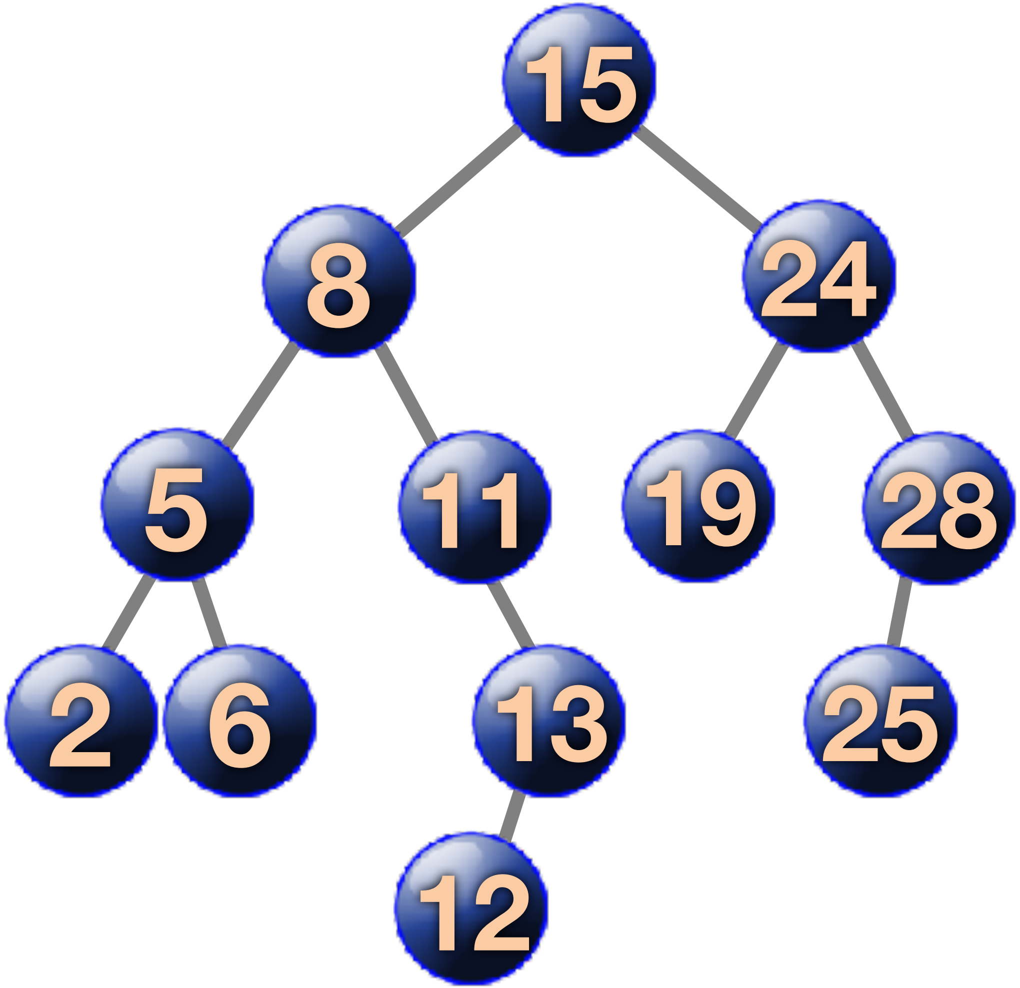
- Start at root
- Return the data if found, or False if not found



BST Delete

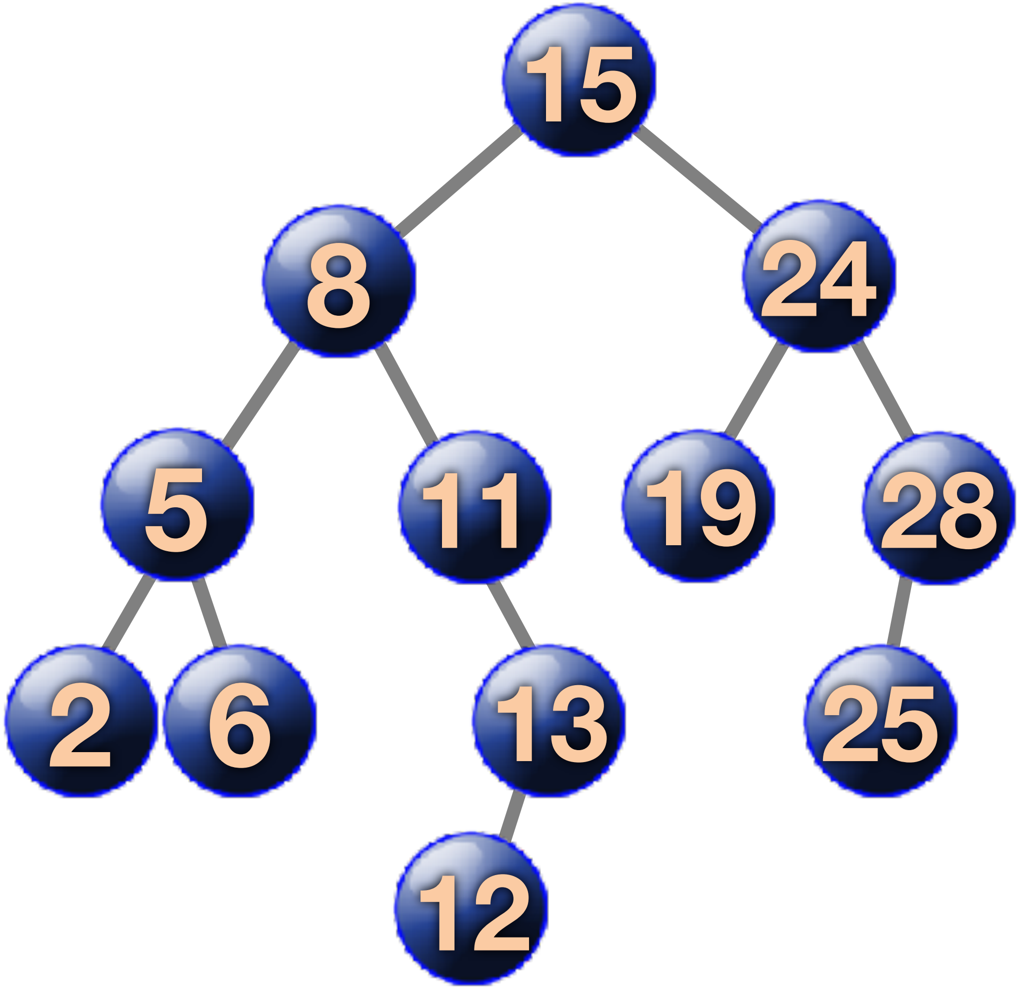
3 possible cases:

- leaf node
- 1 child
- 2 children



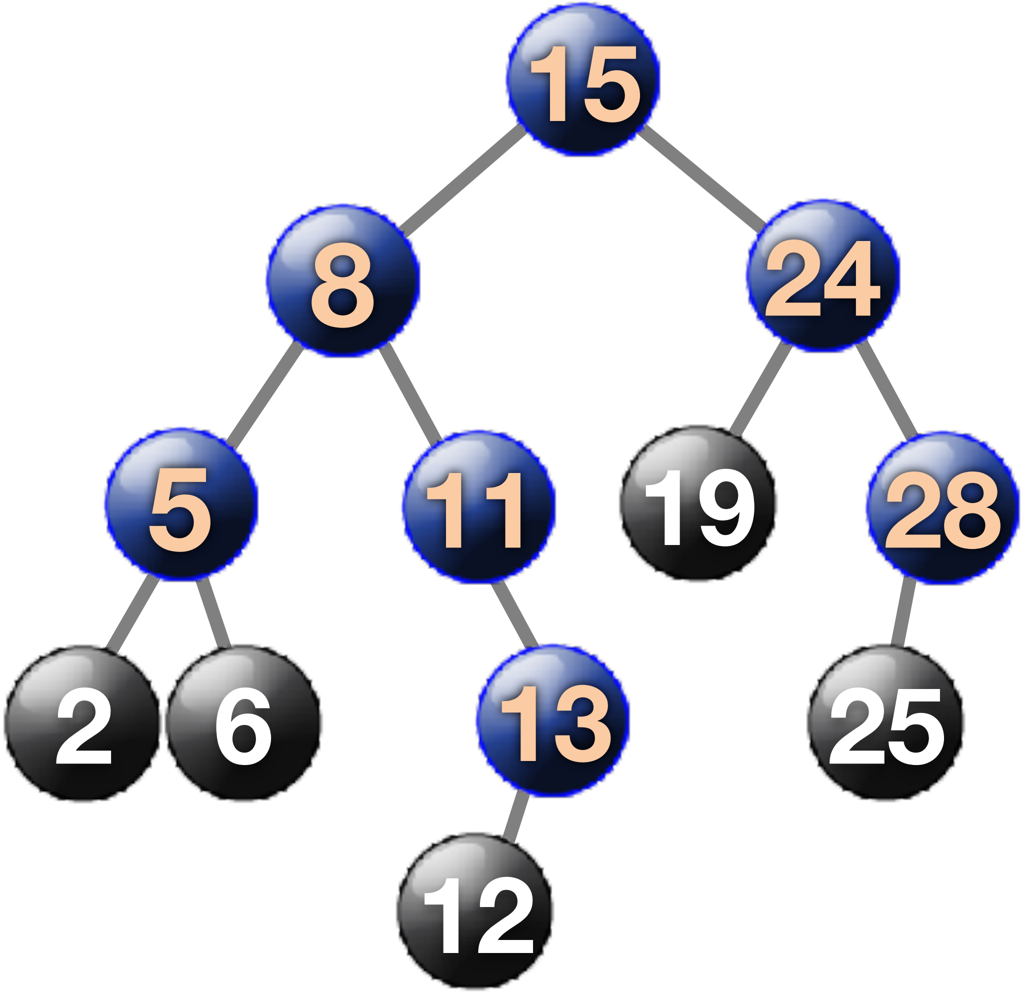
BST Delete

- leaf node



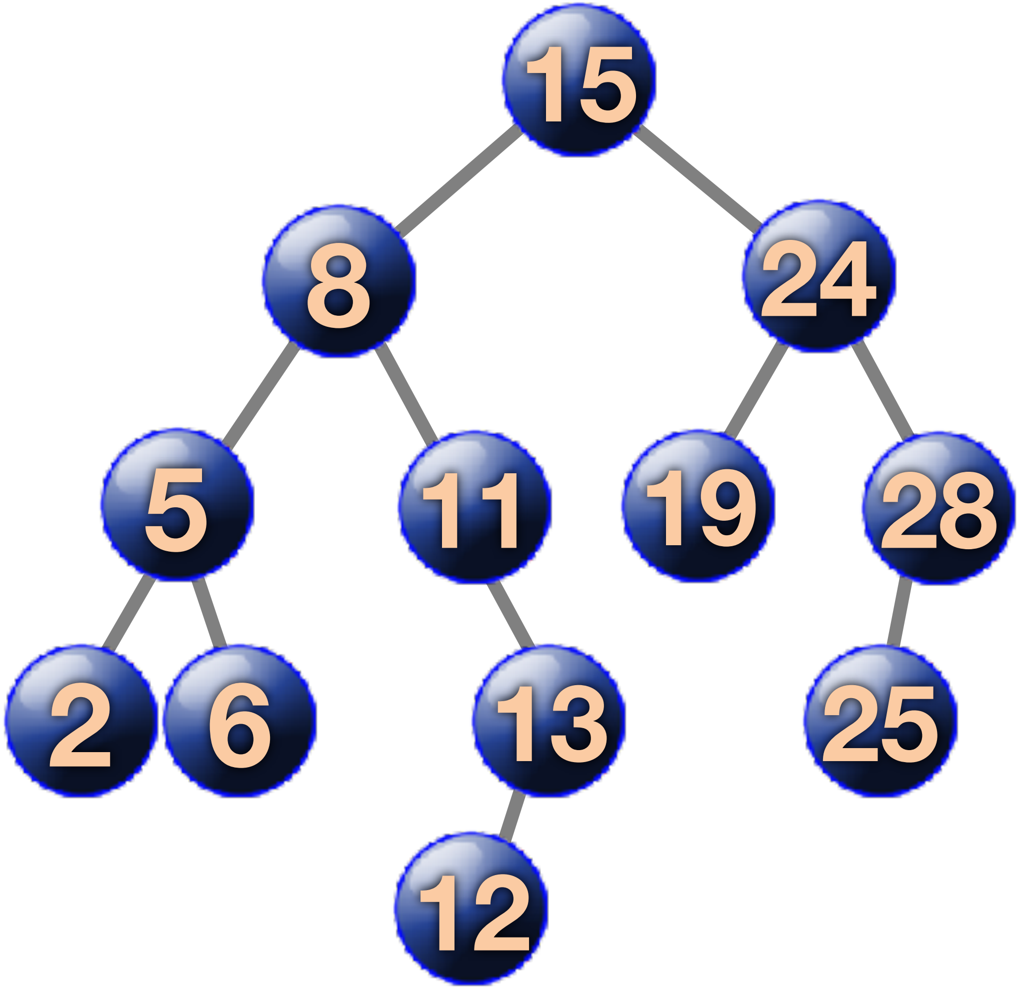
BST Delete

- leaf node
  - just delete the leaf node



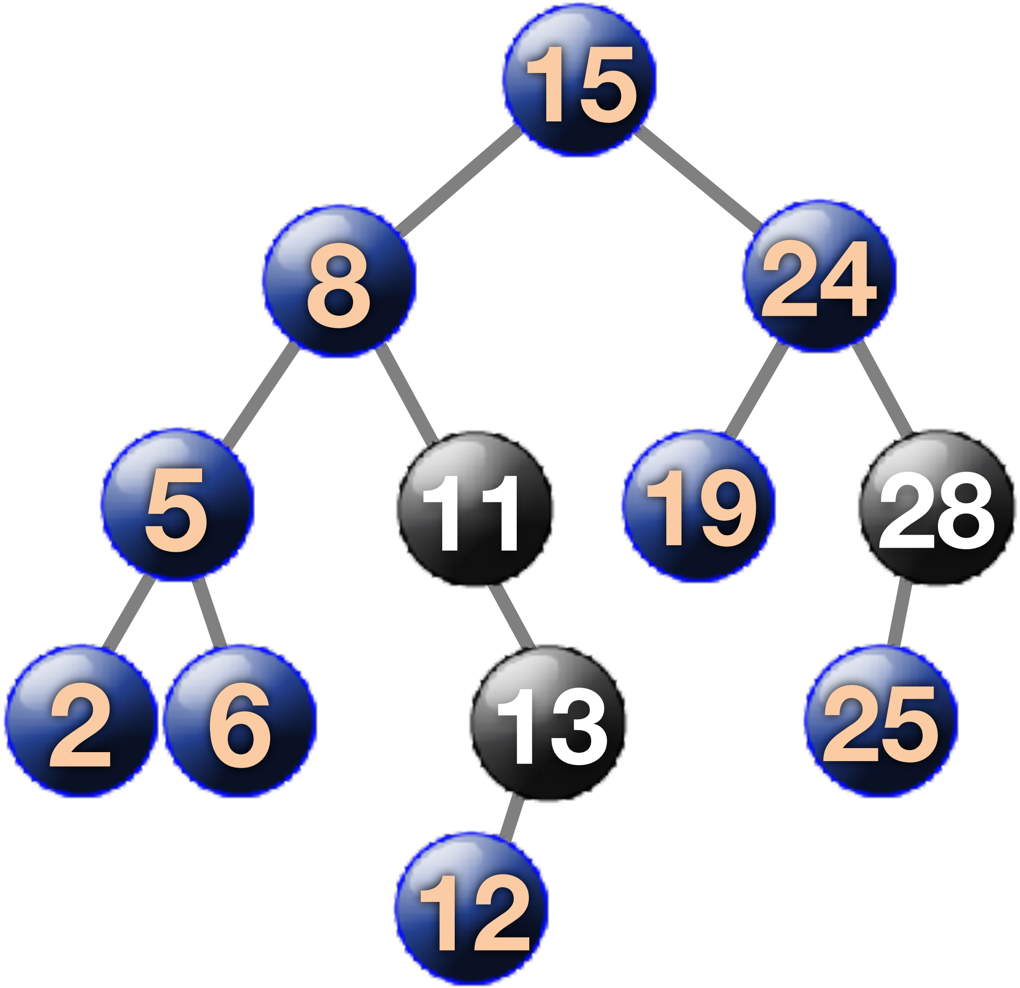
BST Delete

- 1 child



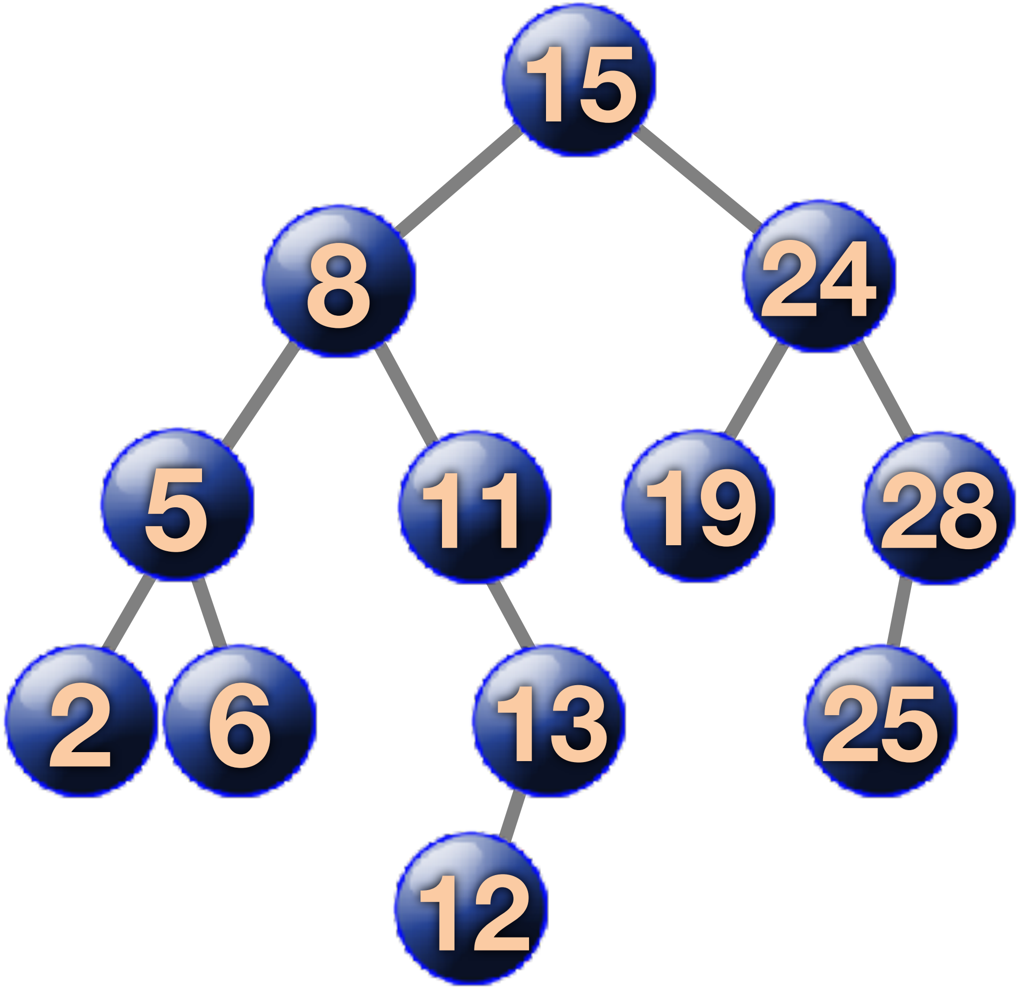
BST Delete

- 1 child
  - promote the child to the target node's position



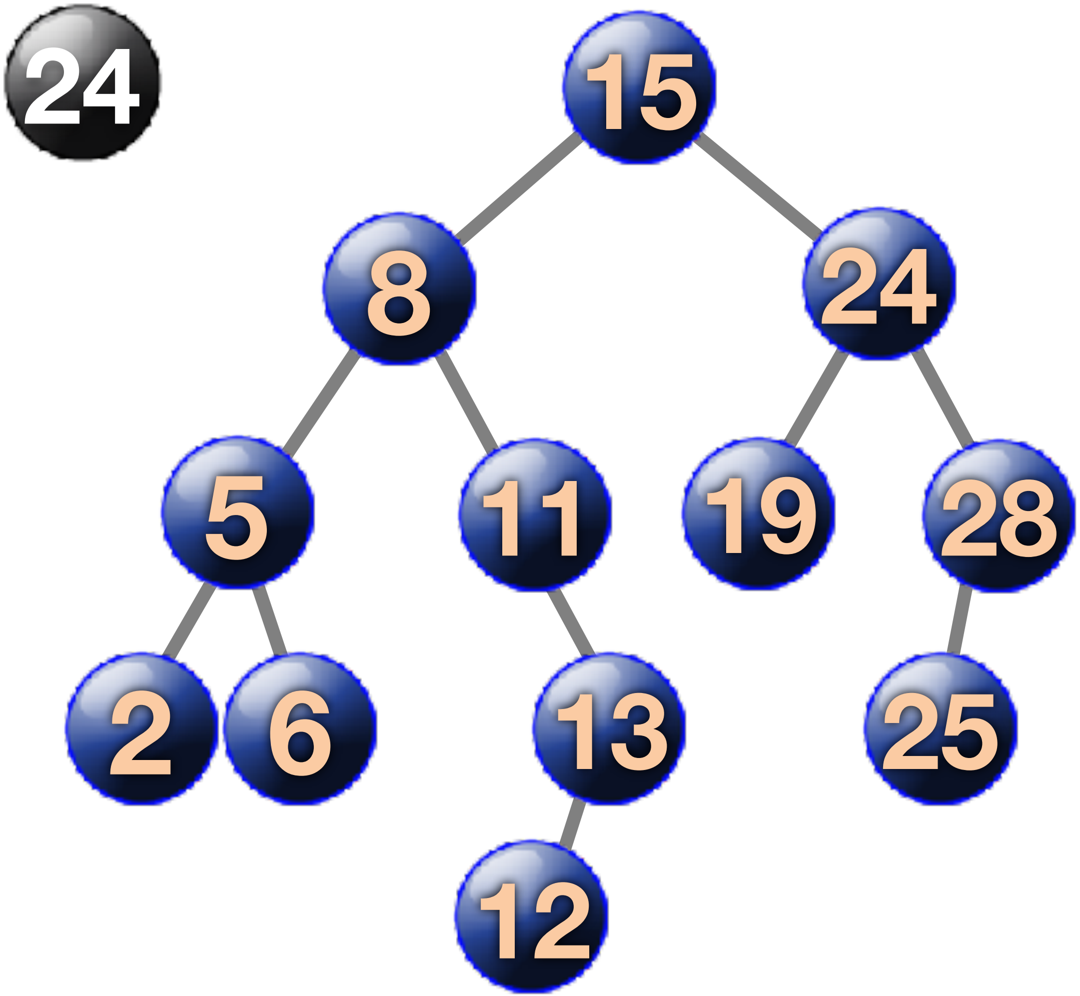
BST Delete

- 2 children



BST Delete

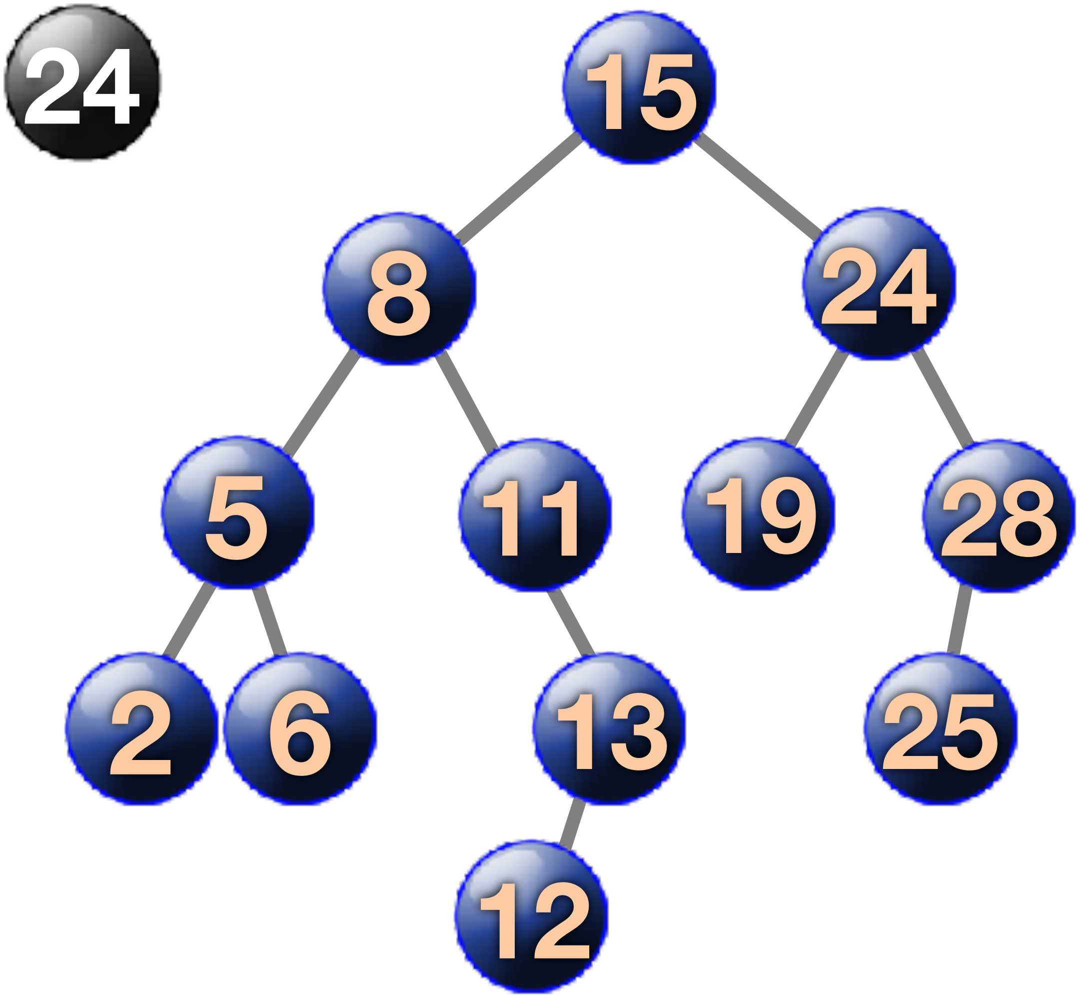
- 2 children



BST Delete

- 2 children

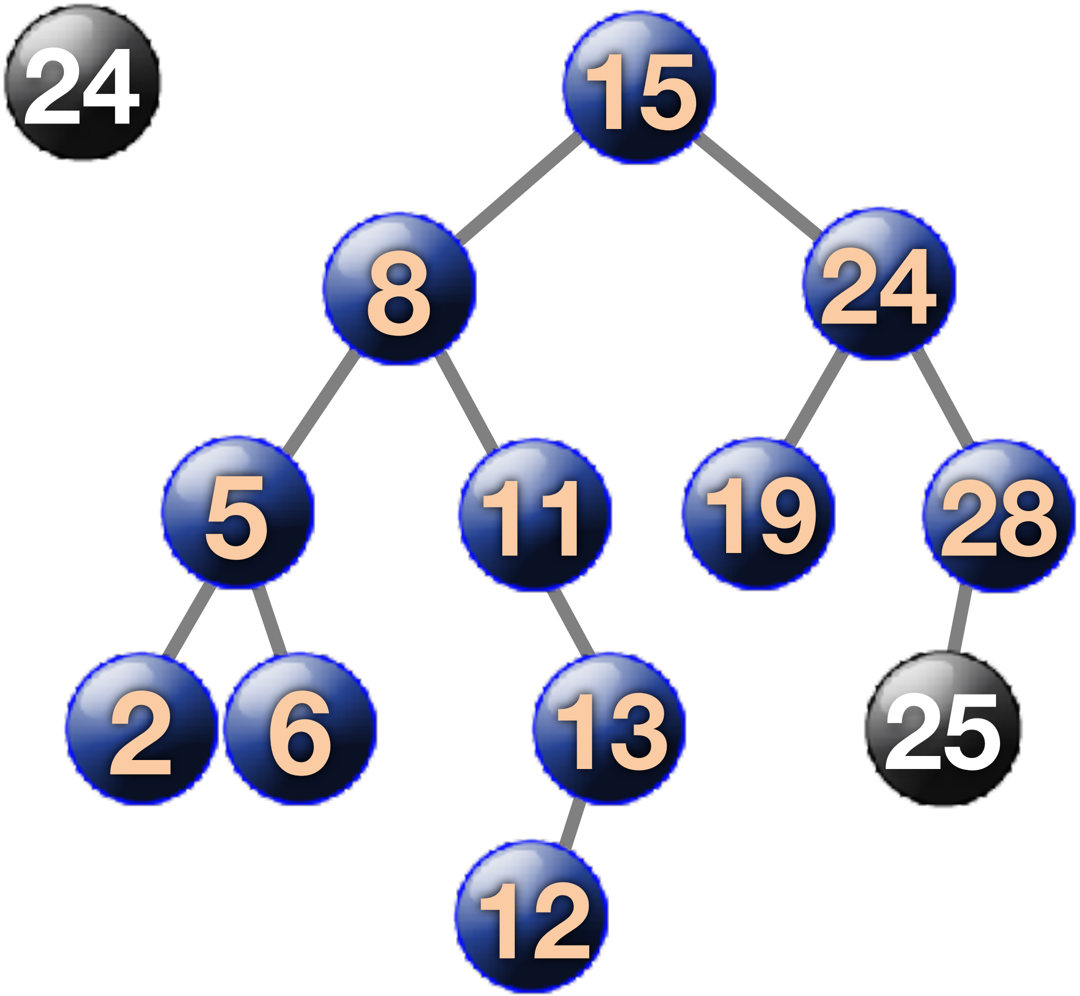
Find the next higher node



BST Delete

- 2 children

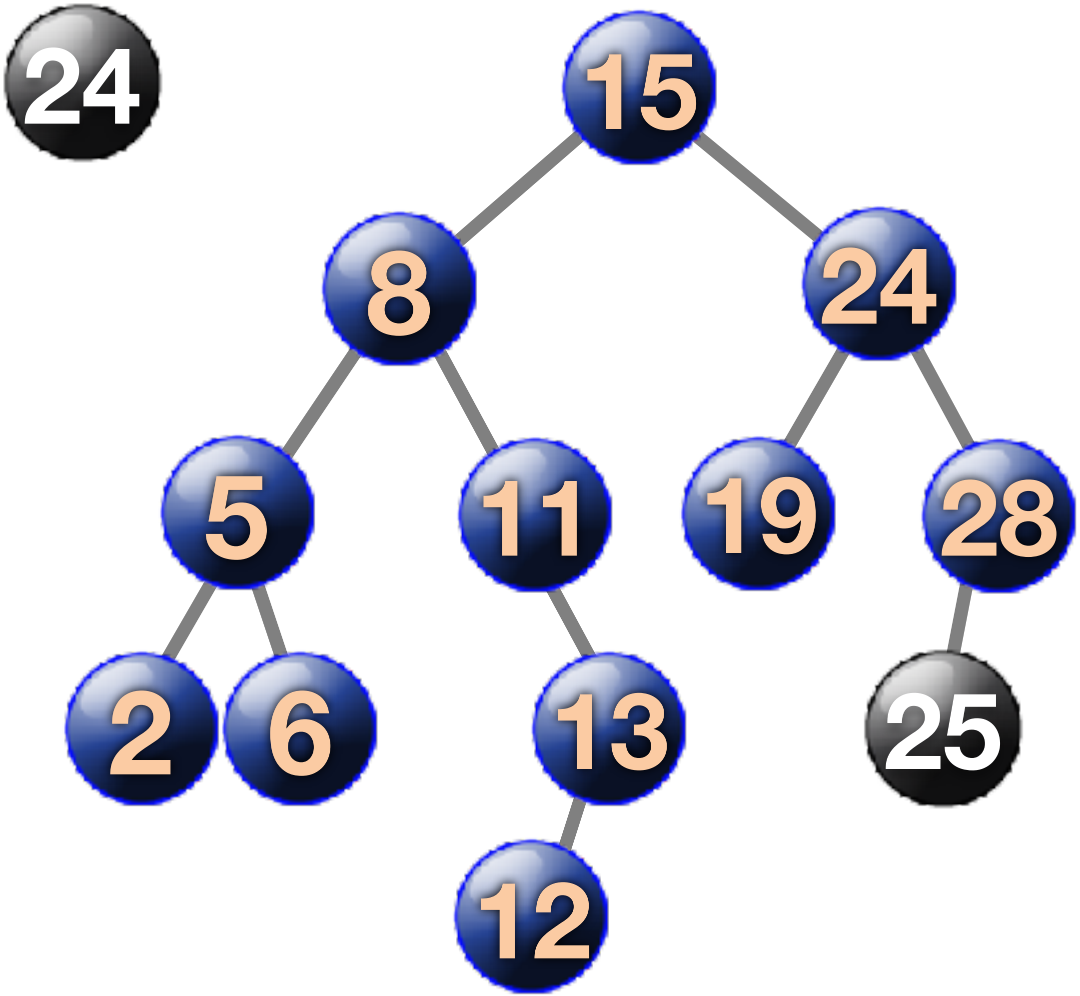
Find the next higher node



BST Delete

- 2 children

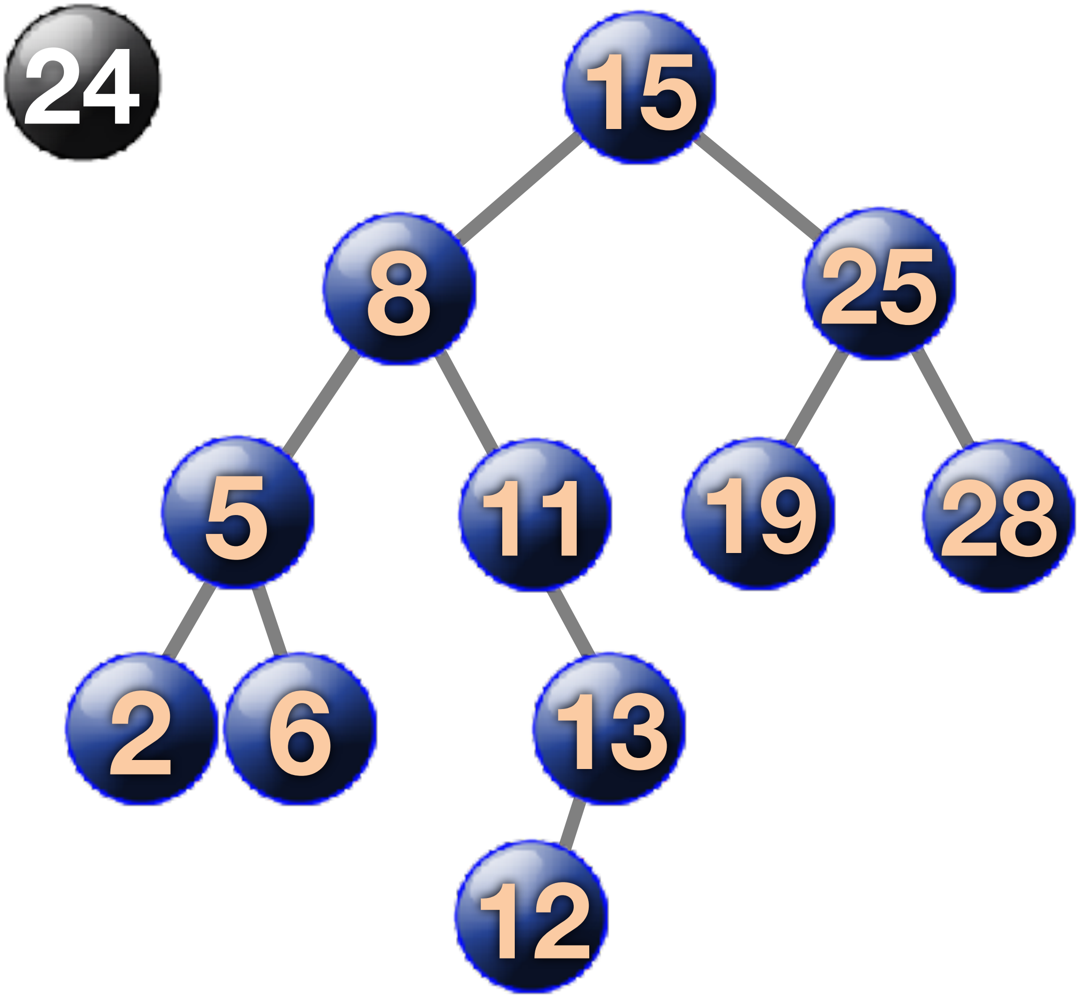
Find the next higher node, change 24 to 25, then delete node 25



BST Delete

- 2 children

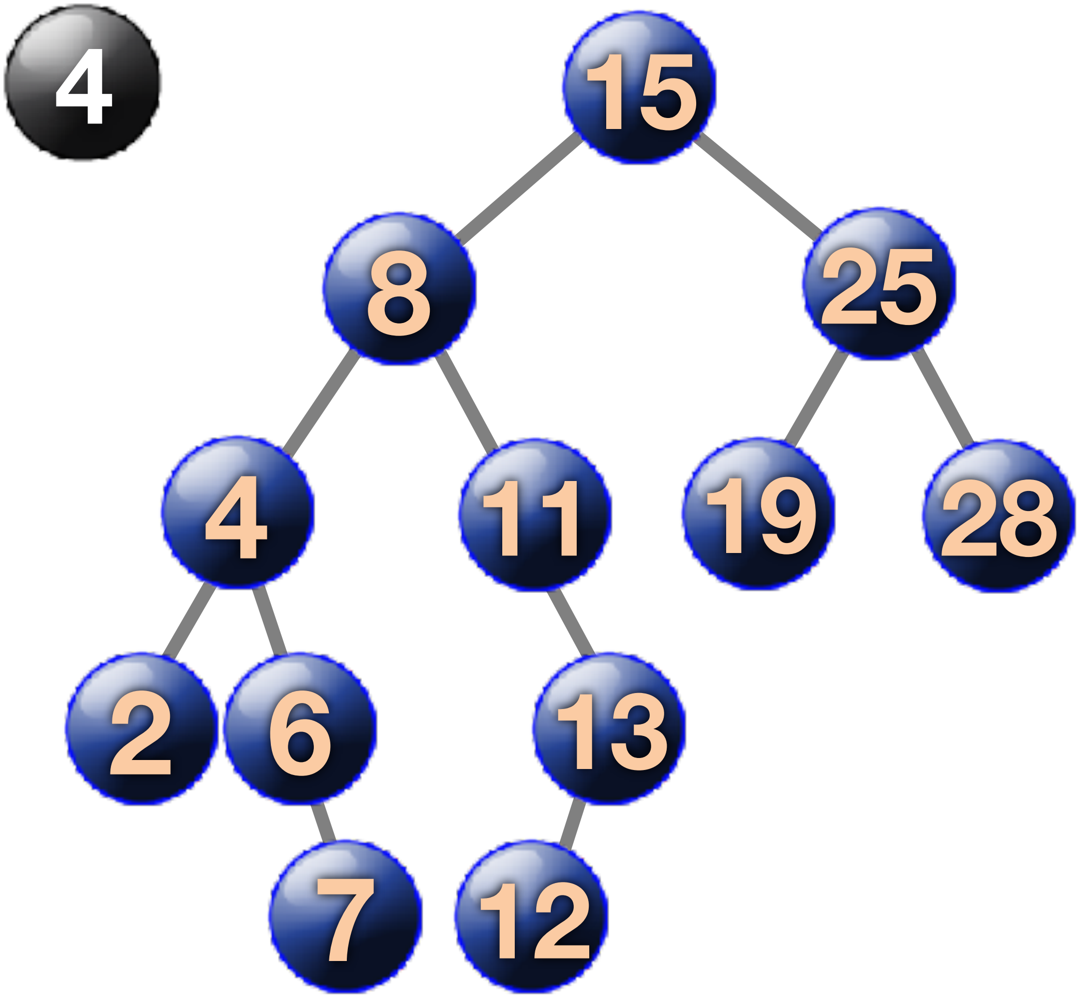
Find the next higher node, change 24 to 25, then delete node 25



BST Delete

- 2 children

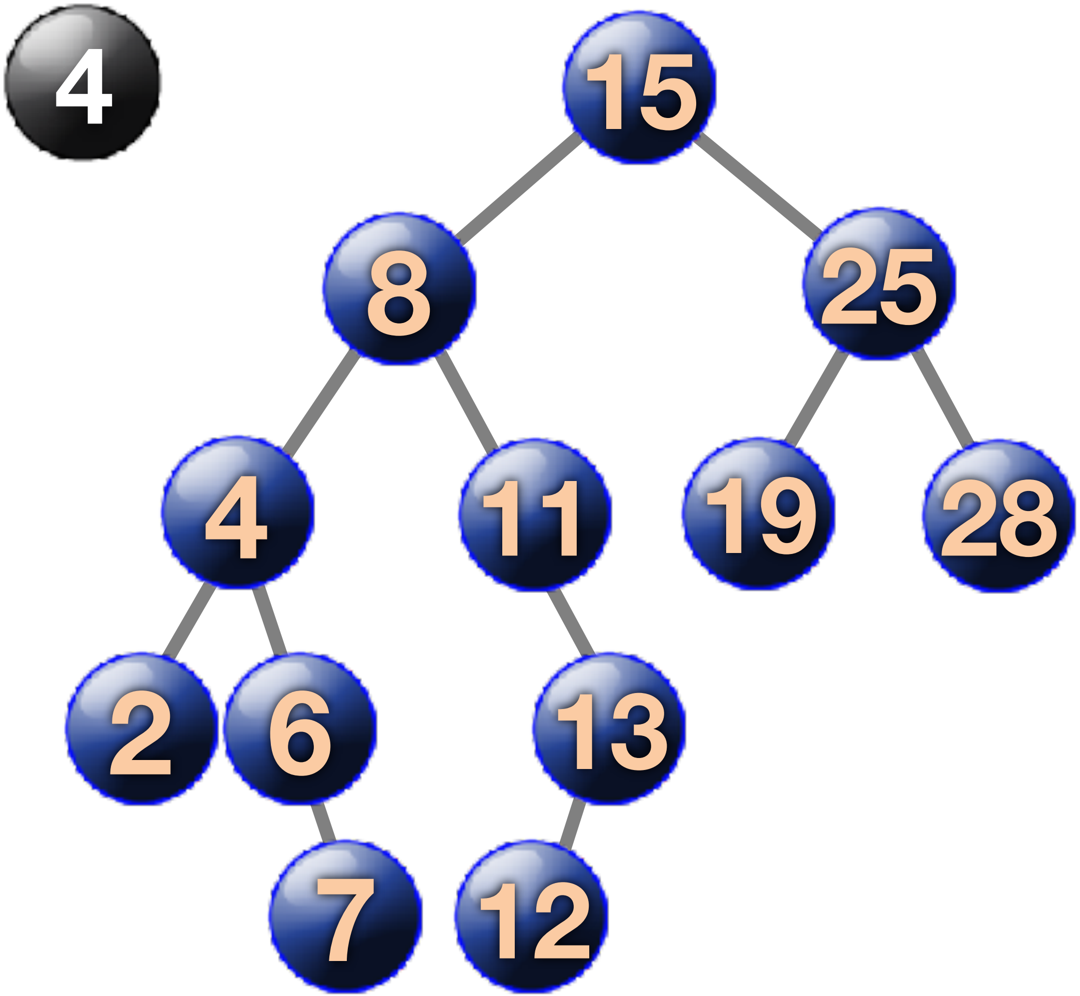
Find the next higher node,



BST Delete

- 2 children

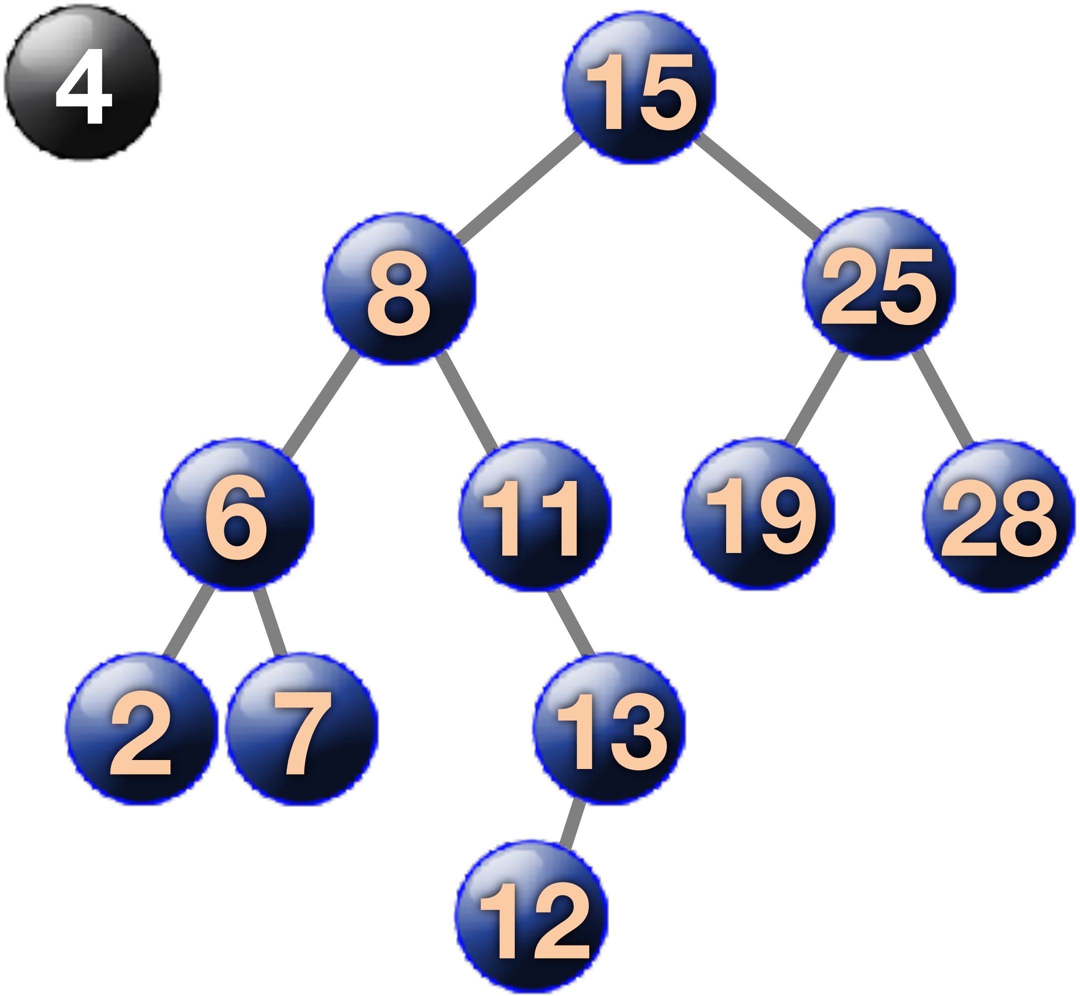
Find the next higher node, change 4 to 6, then delete node 6



BST Delete

- 2 children

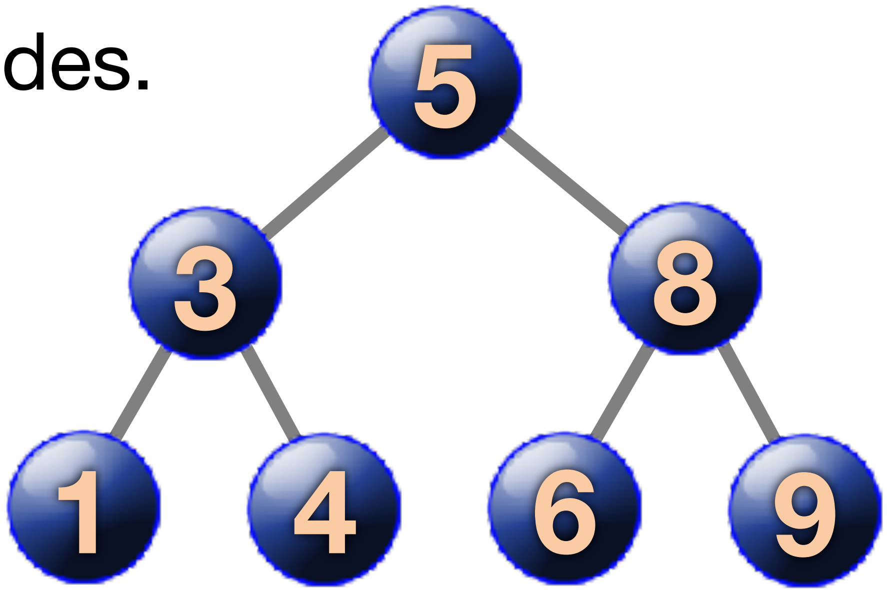
Find the next higher node, change 4 to 6, then delete node 6



Get_size

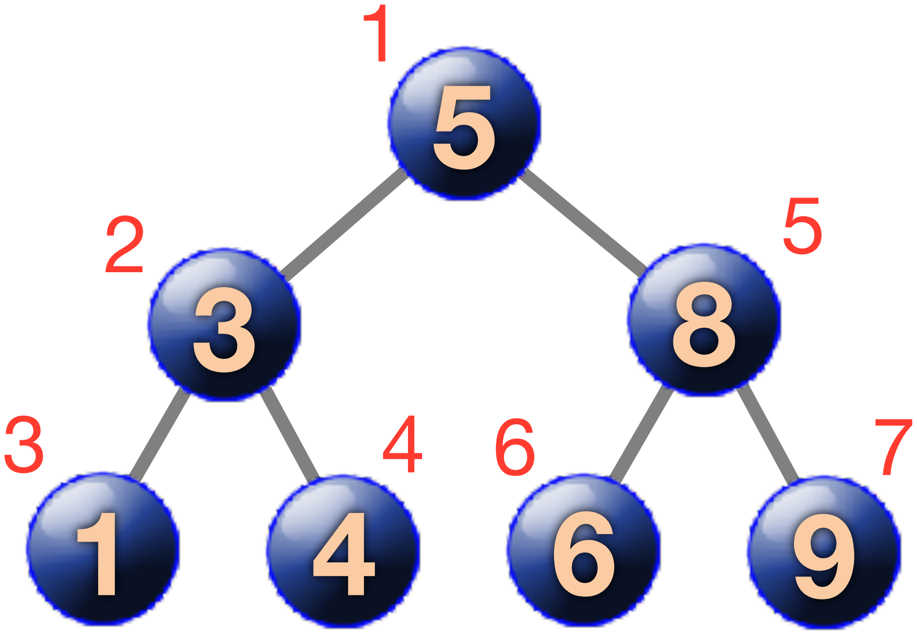
Returns number of nodes.
Works recursively

size = 1
    + size(left subtree)
    + size(right subtree)



Preorder Traversal

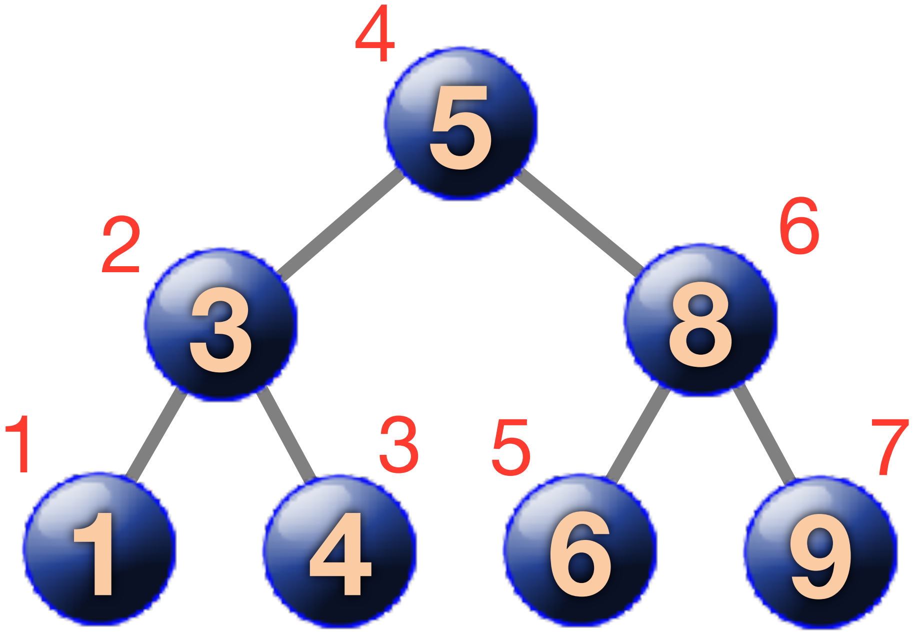
Visit root before visiting the root's subtrees.



Inorder Traversal

Visit root between visiting the root's subtrees.

Gives values in sorted order.



Advantages of Binary Search Trees?

Advantages of Binary Search Trees?

Because trees use recursion for most operations, they are fairly easy to implement.

Advantages of Binary Search Trees?

SPEED

Advantages of Binary Search Trees?

SPEED

Insert, Delete, Find in

O(h) = O(log n)

Advantages of Binary Search Trees?

SPEED

In a balanced BST
with 10,000,000 nodes
Find takes 30 comparisons!

Advantages of Binary Search Trees?

SPEED

Why are trees so fast?

Because each comparison *cuts in half* the number of nodes to search.