

Binary Search Tree

Mai Dahshan

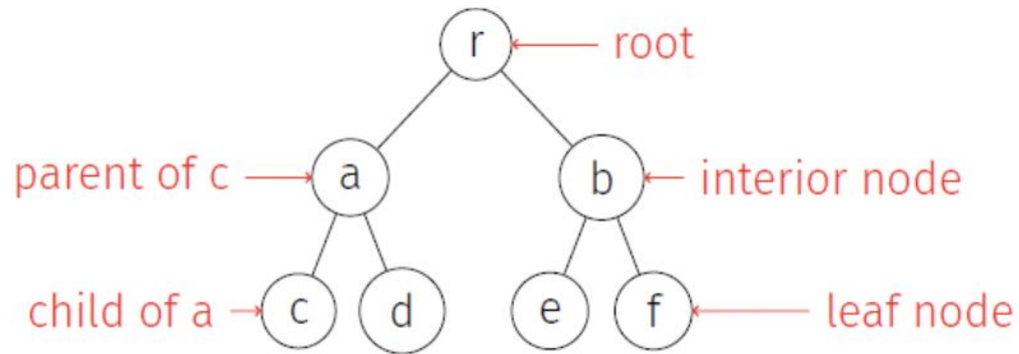
October 7, 2024



Learning Objectives

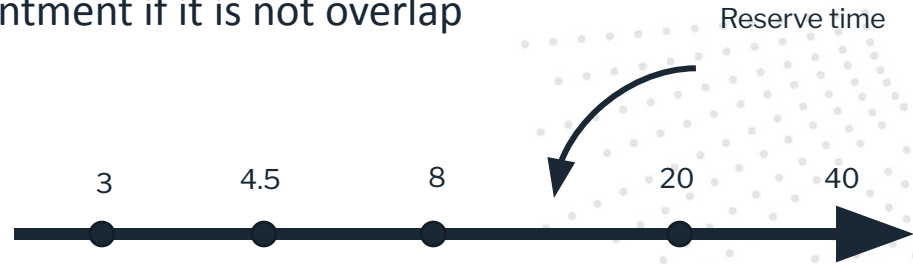
- Understand the concept of BSTs, their properties, and how they are organized
- Searching, inserting and deleting in BSTs
- Analyzing BST time complexity

Binary Tree



Binary Search Trees: Motivation

- Runway reservation system
 - You have to make reservation for plane landing
 - Each landing in 3 mins
 - Reserve request for future landing
 - Add time t to the set of appointment if it is not overlap



Binary Search Trees: Motivation

- What is our possible solution and what wrong with them?
 - Unsorted array
 - Sorted array
 - Sorted linked list

Binary Search Trees: Motivation

- What is our possible solution and what wrong with them?
 - Unsorted array
 - Sorted array
 - Sorted linked list

Data Structure	Operation	Average Case	Worst Case
Sorted Array	Search	$O(\log n)$	$O(\log n)$
	Insert	$O(n)$	$O(n)$
Unsorted Array	Search	$O(n)$	$O(n)$
	Insert	$O(1)$	$O(1)$
Sorted Linked List	Search	$O(n)$	$O(n)$
	Insert	$O(n)$	$O(n)$

choose this for frequent searches

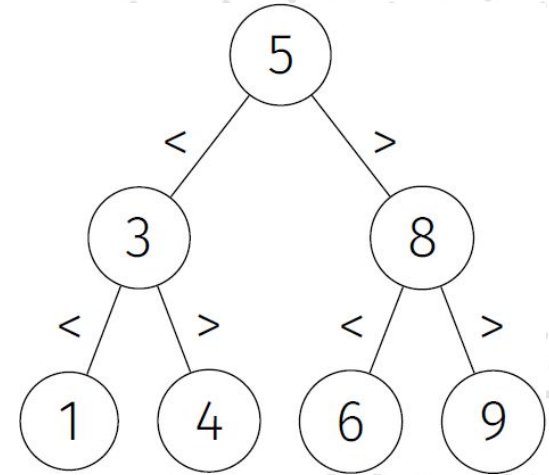
choose this for frequent insertions

Binary Search Trees: Motivation

- Binary search tree is the one would allow us to do the above things in $O(\log n)$

Binary Search Tree

- Binary tree with **comparable** key values
- **Binary search tree property:**
 - Every node in the **left** subtree has key whose value is **less** than the value of the root's key value, and
 - Every node in the **right** subtree has key whose value is **greater** than the value of the root's key value.



BST visualization : <https://visualgo.net/en/bst>

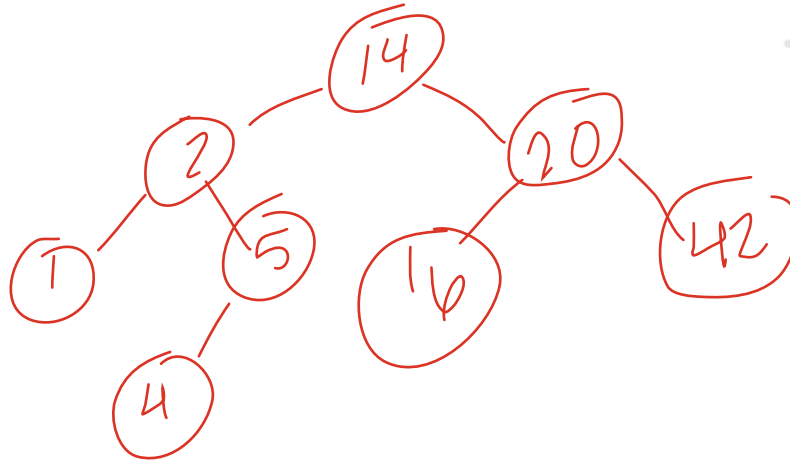
Binary Search Tree

- How could we traverse a BST so that the nodes are visited in **sorted** order?
 - *In-order* traversal: left tree, node, right tree

Binary Search Tree Practice

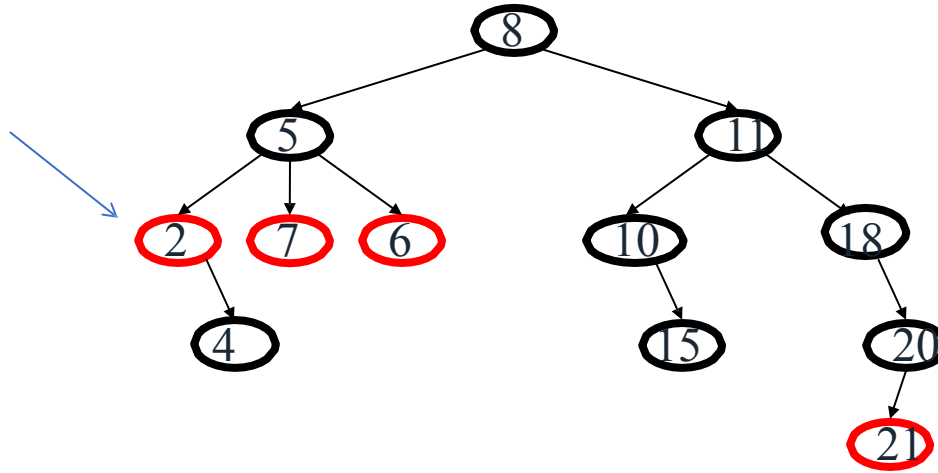
Start with an empty tree. Insert the following values, in the given order:

14, 2, 5, 20, 42, 1, 4, 16



Binary Search Tree

Three
child
nodes

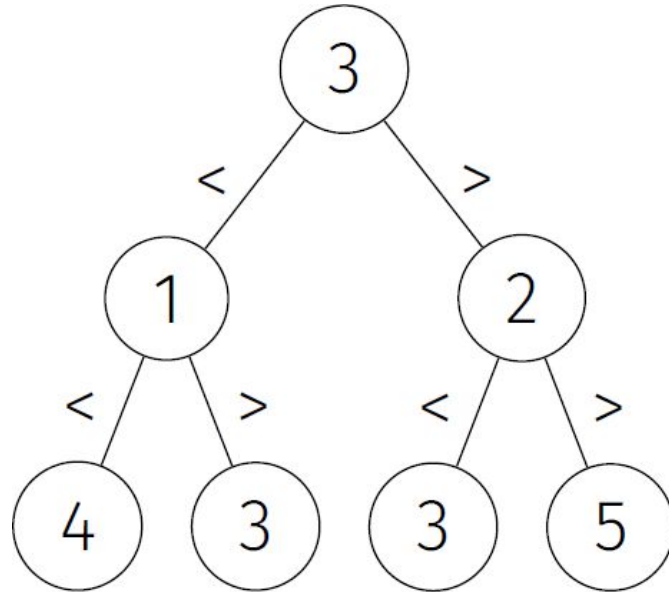


Left node not
less than root

NOT A BINARY SEARCH TREE

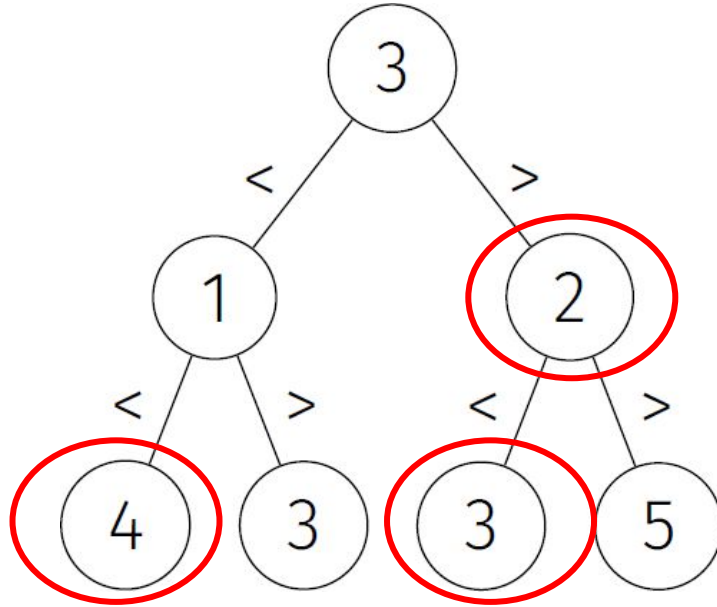
Quiz

- Is this a binary search tree?



Quiz

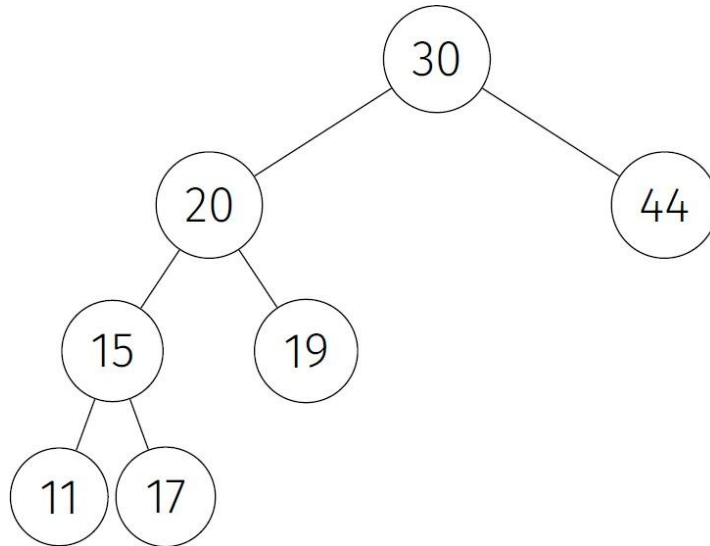
- Is this a binary search tree?



No! Binary search tree property not preserved

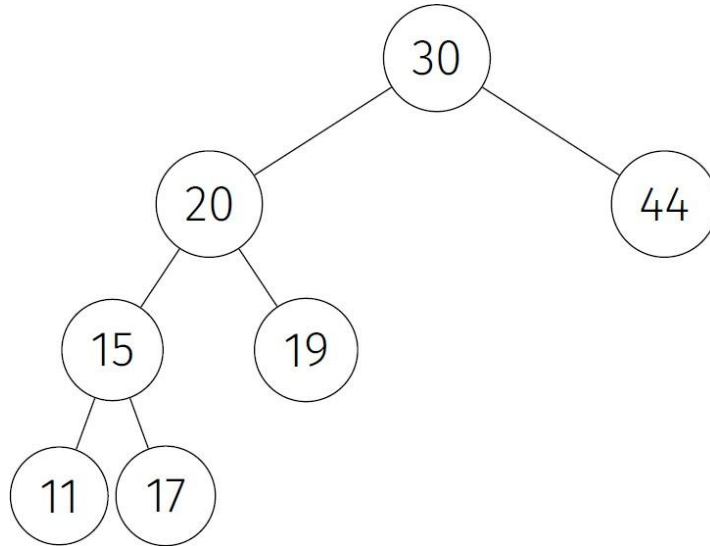
Quiz

- Is this a binary search tree?



Quiz

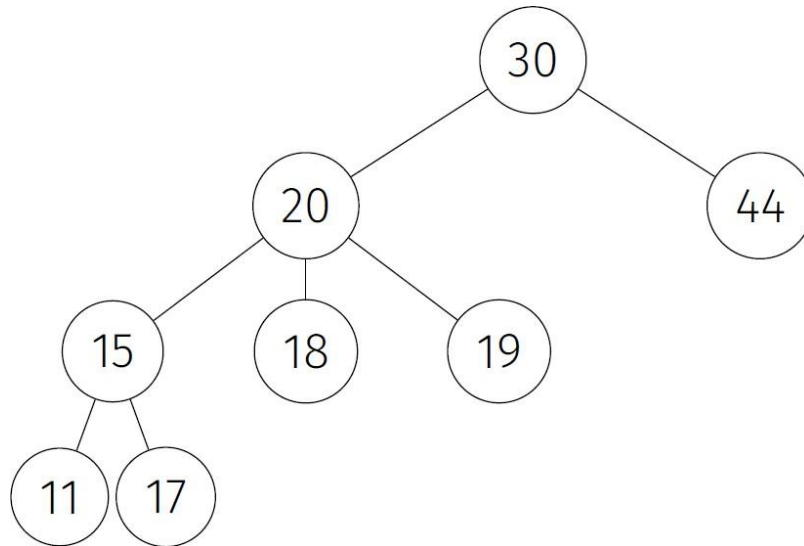
- Is this a binary search tree?



No! Binary search tree
property not preserved

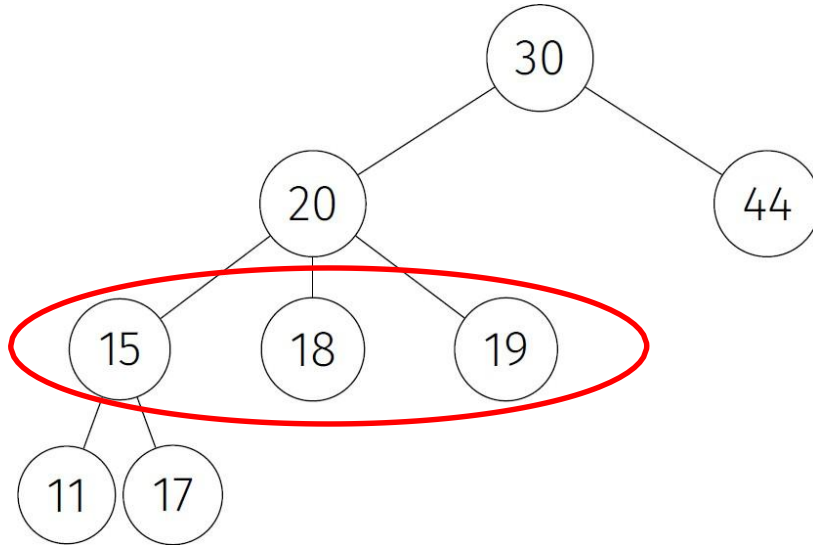
Quiz

- Is this a binary search tree?



Quiz

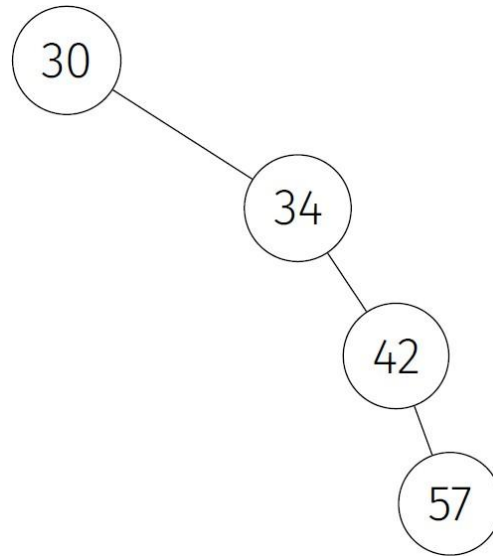
- Is this a binary search tree?



No! Binary search tree
property not preserved

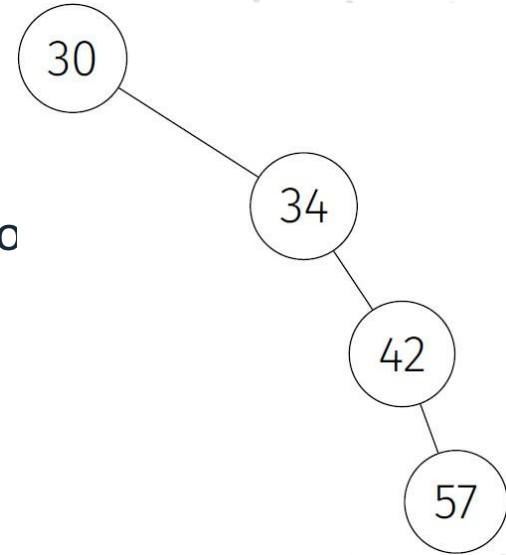
Quiz

- Is this a binary search tree?



Quiz

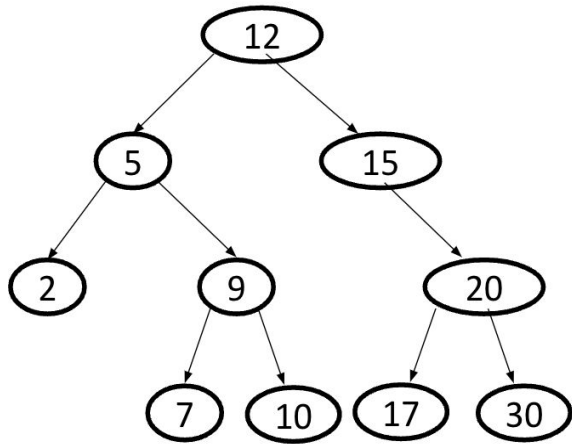
- Is this a binary search tree? **Yes!**
- However, this tree is unbalanced!
 - This is an ordered list
- A **balanced** binary tree
- Guarantees height of child subtrees differ by no



How do we search an element in BST?

- **Find** an element in the tree
 - Compare with root, if less traverse left, else traverse right; repeat
 - Stops when found or at a leaf

How do we search an element in BST?



Data **find**(Data value, Node root)

```
if(root == null)
  return null
```

```
if(key < root.value)
  return find(value, root.left)
```

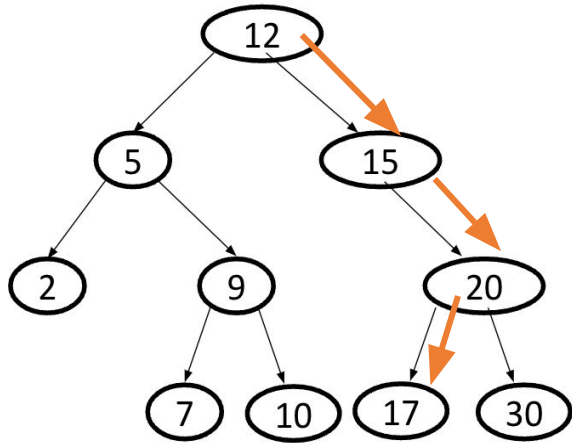
```
if(key > root.value)
  return find(value, root.right)
```

```
return root.value;
```

Sounds like **binary search**!

How do we search an element in BST?

Find (17)



```
Data find(Data value, Node root)
```

```
    if(root == null)  
        return null
```

```
    if(key < root.value)  
        return find(value, root.left)
```

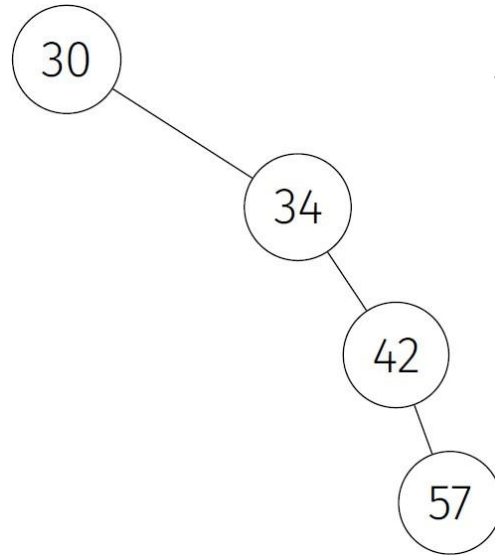
```
    if(key > root.value)  
        return find(value, root.right)
```

```
    return root.value;
```

What is the running time?

How do we search an element in BST?

Find (57)



What is the running time?

How do we search an element in BST?



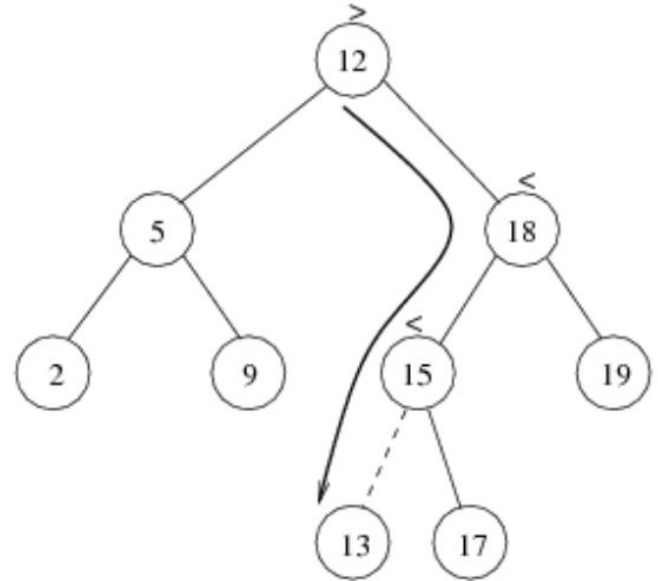
- what is *minimum* node in BST?
- what is the *maximum* node in BST?

Time Complexity of Searching BST

Operation	Best Case	Average Case	Worst Case
Searching	$O(1)$	$O(\log n)$	$O(n)$

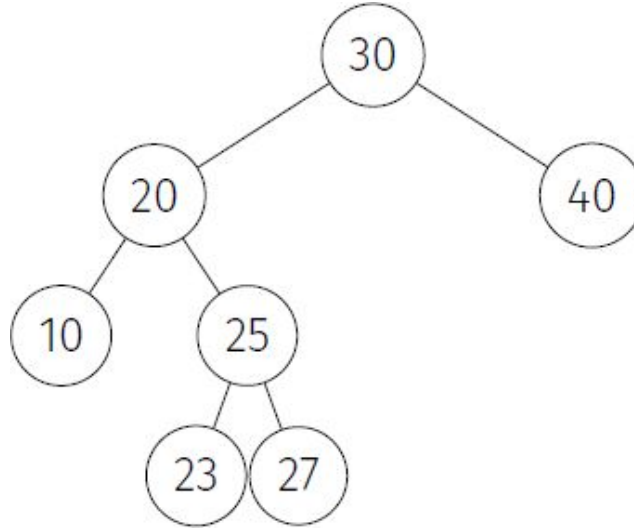
How do we insert an element in BST?

- Proceed down the tree as you would with a search
- Insert the new element at the last spot on the path traversed (add it at the leaf node)



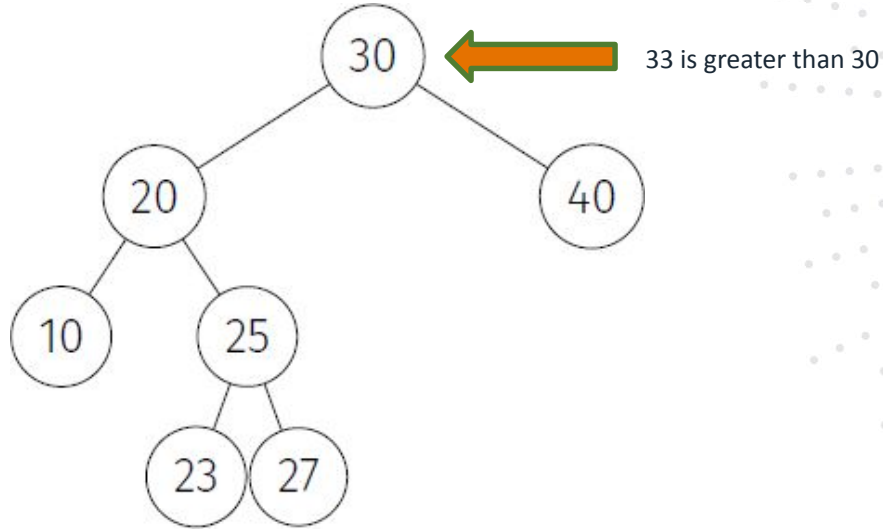
How do we insert an element in BST?

Insert 33 into the following binary search tree



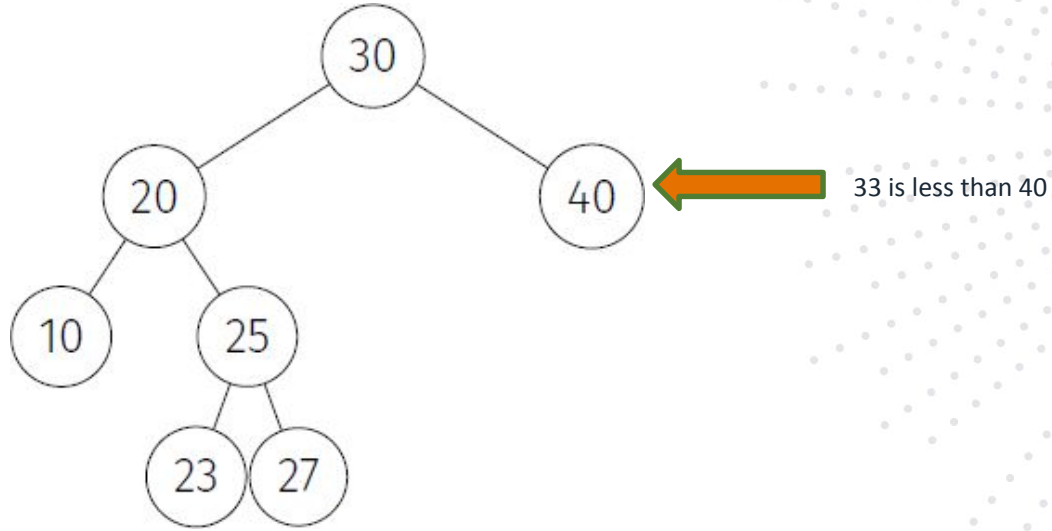
How do we insert an element in BST?

Insert 33 into the following binary search tree



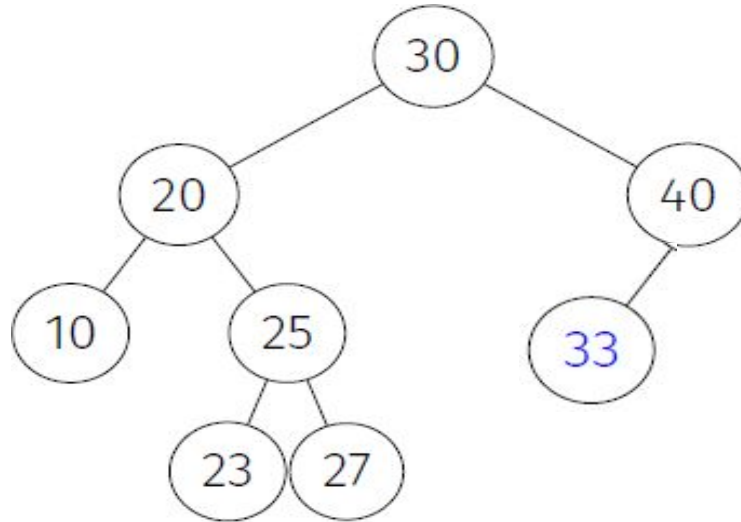
How do we insert an element in BST?

Insert 33 into the following binary search tree



How do we insert an element in BST?

Insert 33 into the following binary search tree



Time Complexity of Insertion BST

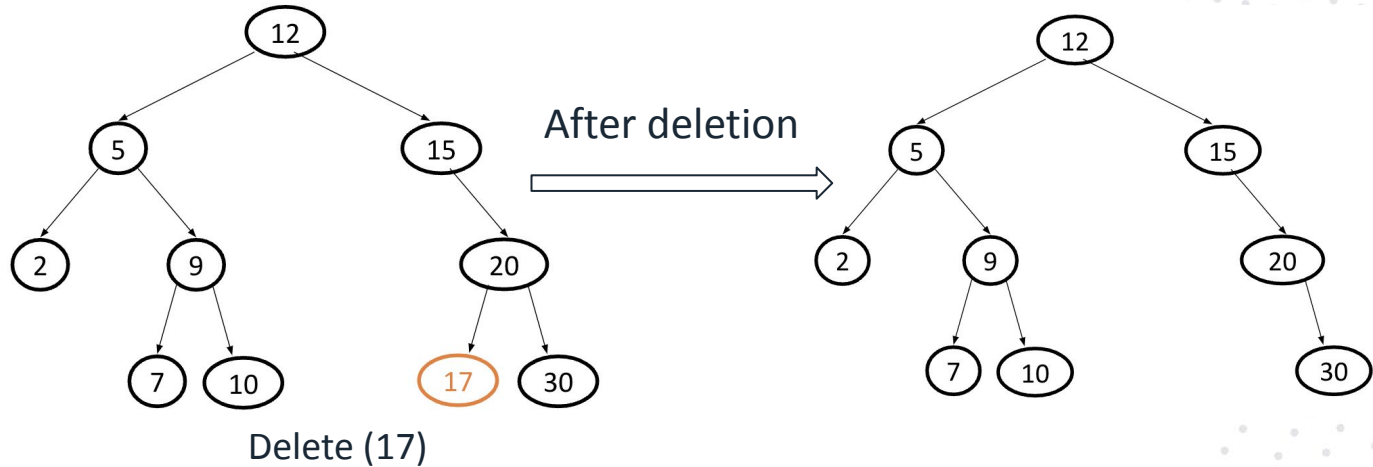
Operation	Best Case	Average Case	Worst Case
Searching	$O(1)$	$O(\log n)$	$O(n)$

How do we delete an element from BST?

- When we delete a node, we need to consider how we take care of the children of the deleted node
 - This has to be done such that the property of the search tree is maintained
 - More complicated – we need to select a node as replacement!
- Deletion under Different Cases
 - Case 1: the node is a leaf
 - Case 2: the node has one child
 - Case 3: the node has 2 children

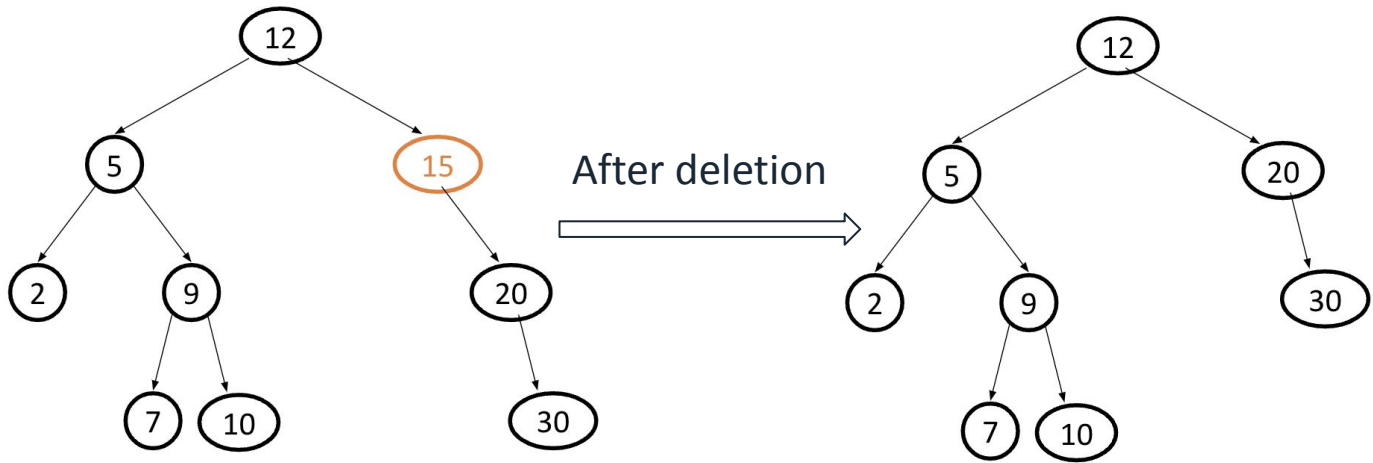
How do we delete an element from BST?

- Case 1: the node is a leaf
 - Delete it immediately



How do we delete an element from BST?

- Case 2: the node has one child
 - Adjust a pointer from the parent to bypass that node



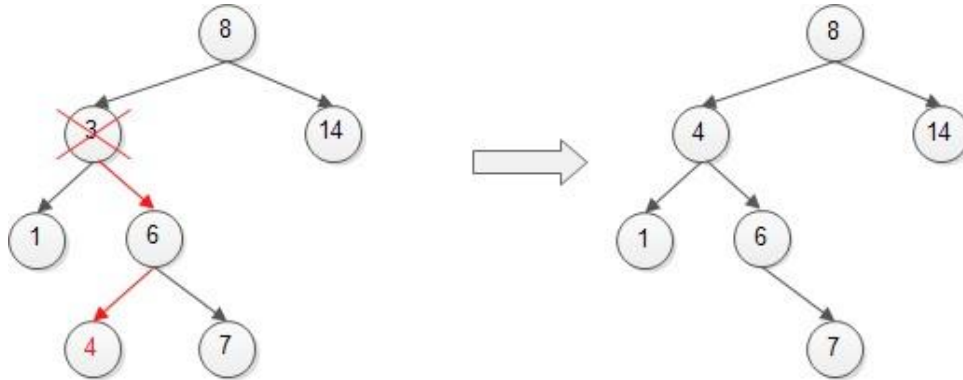
Delete (15)

How do we delete an element from BST?

- Case 3: the node has 2 children
 - Replace the deleted node with its successor
- Where to find the successor? there are 2 options:
 - The next “largest” element - in left subtree
 - The next “smallest” element - in right subtree

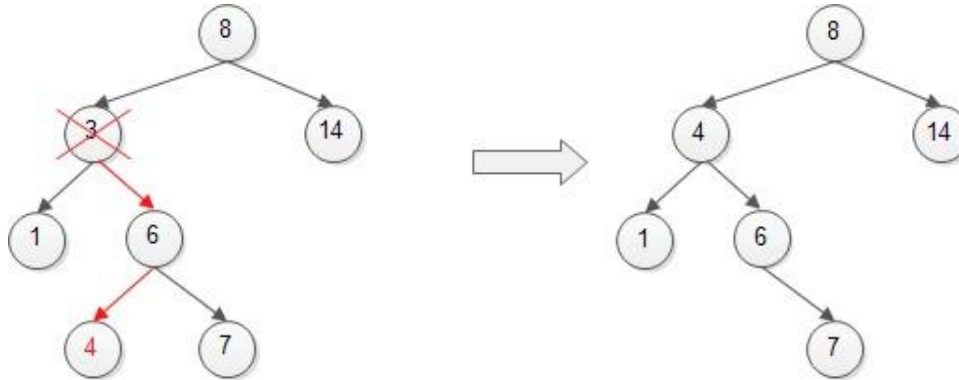
How do we delete an element from BST?

- Find successor (of 3) in its **right** subtree (i.e. node **4**) – finding the **minimum** (*leftmost node*) of right subtree



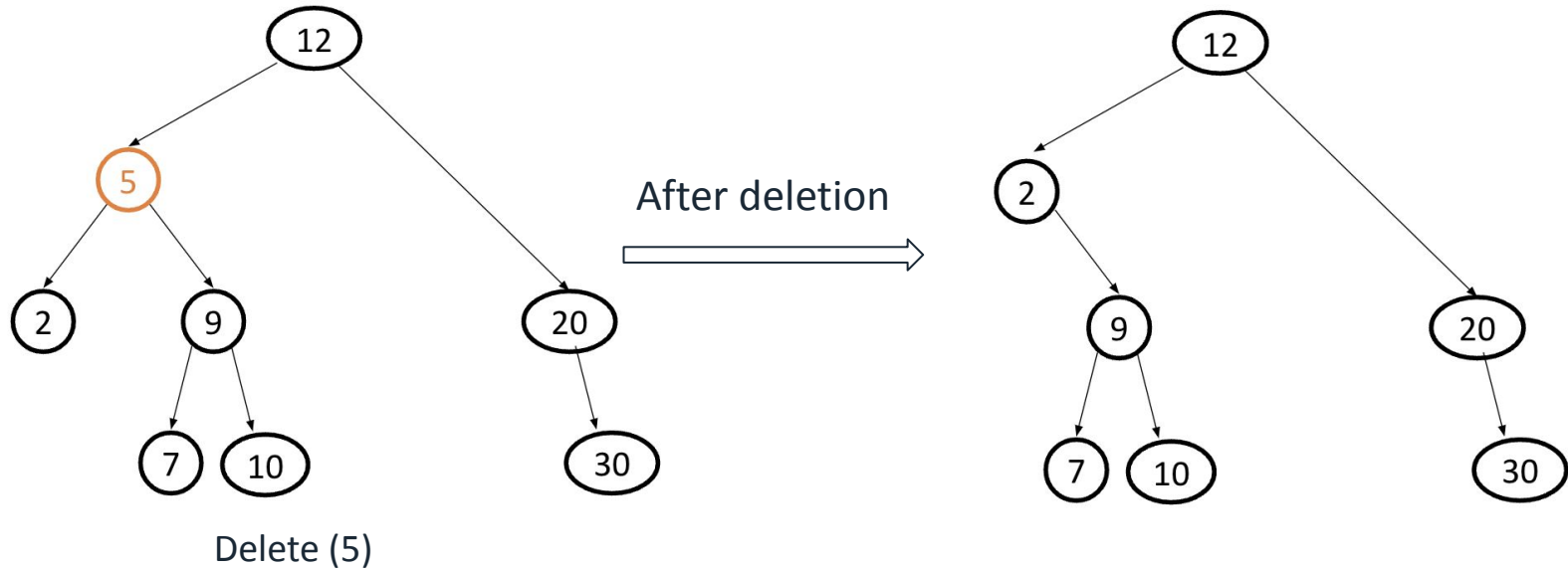
How do we delete an element from BST?

- Find successor (of 3) in its **right** subtree (i.e. node **4**) – finding the **minimum** (*leftmost node*) of right subtree



How do we delete an element from BST?

- Largest number of **left** subtree, which is the next smallest number



Time Complexity of Deletion BST

Operation	Best Case	Average Case	Worst Case
Searching	$O(1)$	$O(\log n)$	$O(n)$

Quiz

What is the time complexity of building BST in average and worst cases?