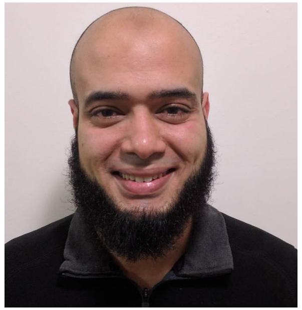
Data Structures
BST Insertion

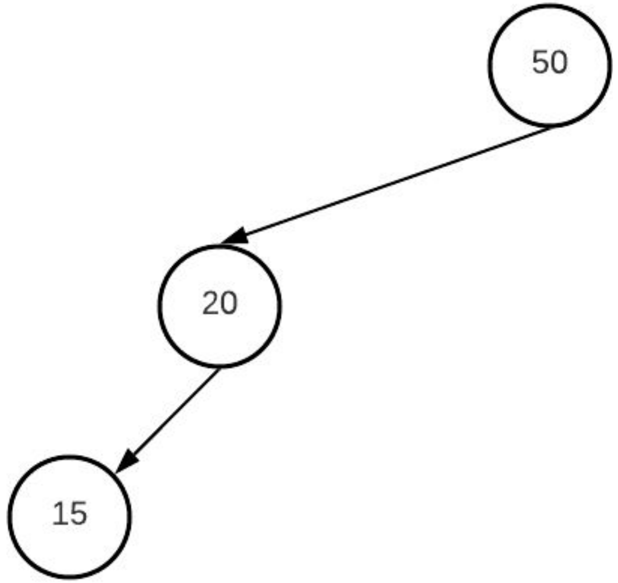
Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)



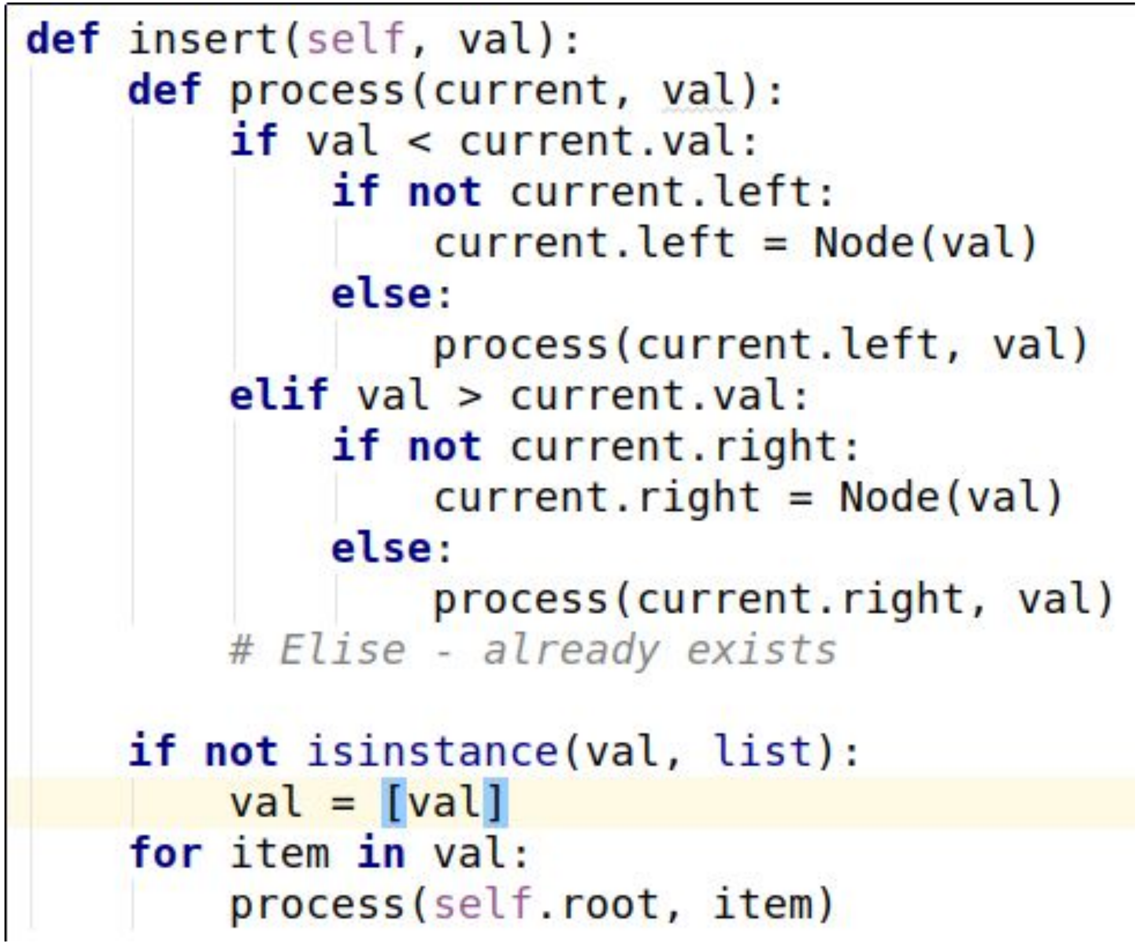
Insertion

- Assume we have the following BST
- How can we insert values: 45, 35?
- We need to find the correct parent and add the value to it
- Try to code: ded **insert**(self, target)



Insertion

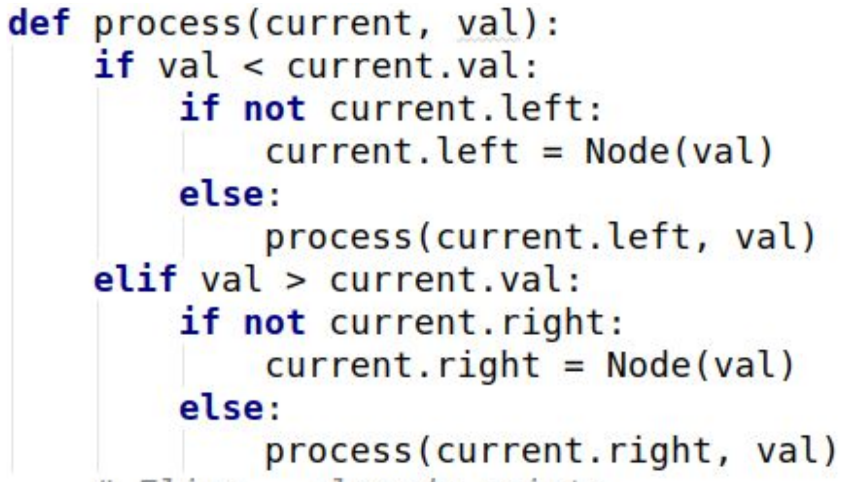
- Given a value, we can identify where to insert it
- The function receives a single value or list of values



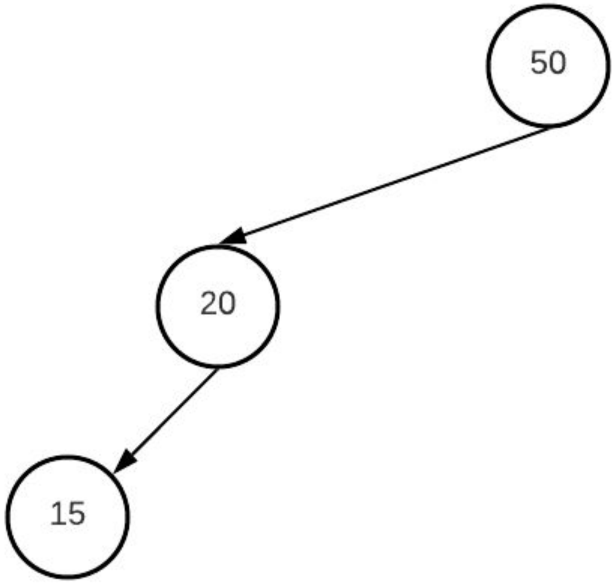
```python
def insert(self, val):
    def process(current, val):
        if val < current.val:
            if not current.left:
                current.left = Node(val)
            else:
                process(current.left, val)
        elif val > current.val:
            if not current.right:
                current.right = Node(val)
            else:
                process(current.right, val)
        # Elise - already exists

    if not isinstance(val, list):
        val = [val]
    for item in val:
        process(self.root, item)
```

Insert 45

- At 50: go left
- At 20: go right
    - No right
    - Create right(45)



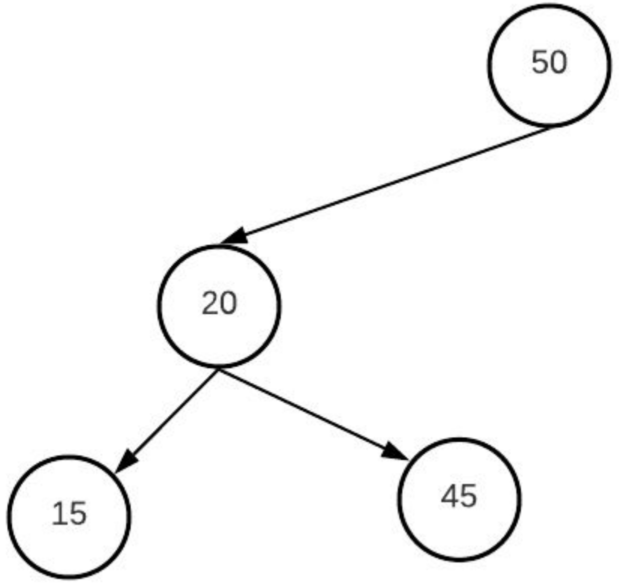
```
def process(current, val):
    if val < current.val:
        if not current.left:
            current.left = Node(val)
        else:
            process(current.left, val)
    elif val > current.val:
        if not current.right:
            current.right = Node(val)
        else:
            process(current.right, val)
```



Insert 35

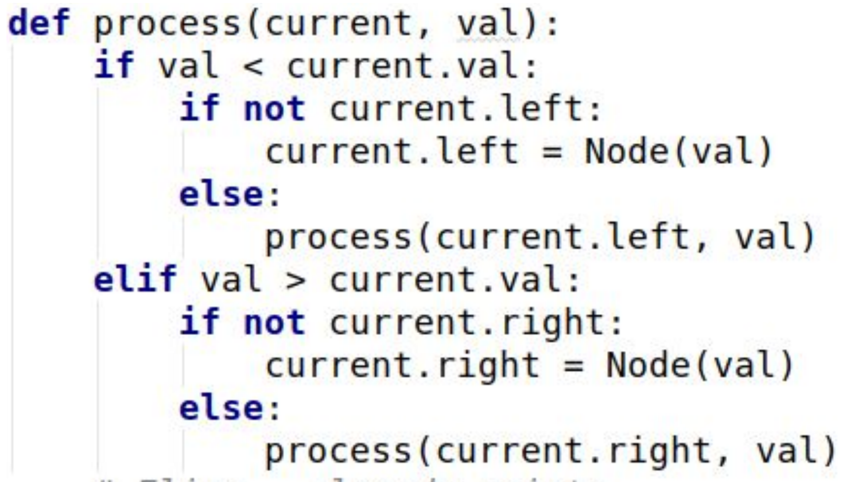
- At 50: go left
- At 20: go right
- At 45: go left
  - No left
  - Create left(35)

```python
def process(current, val):
    if val < current.val:
        if not current.left:
            current.left = Node(val)
        else:
            process(current.left, val)
    elif val > current.val:
        if not current.right:
            current.right = Node(val)
        else:
            process(current.right, val)
```

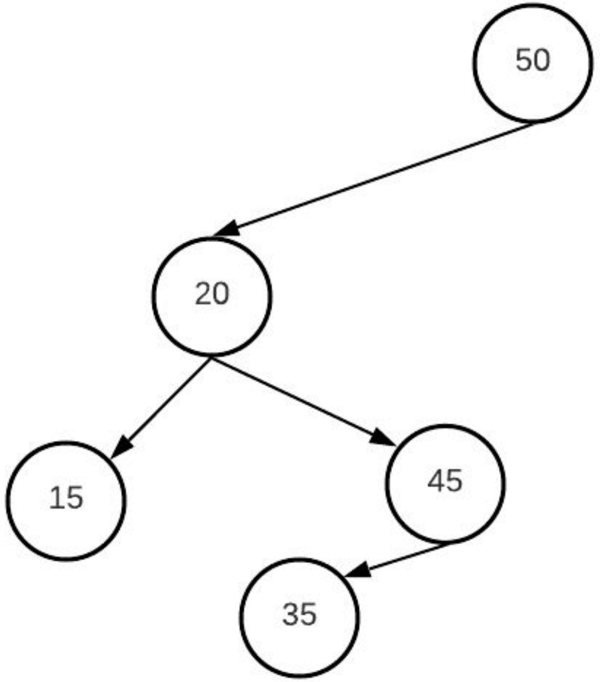


Insert 70

- At 50: go right
- No right
    - Create right(70)

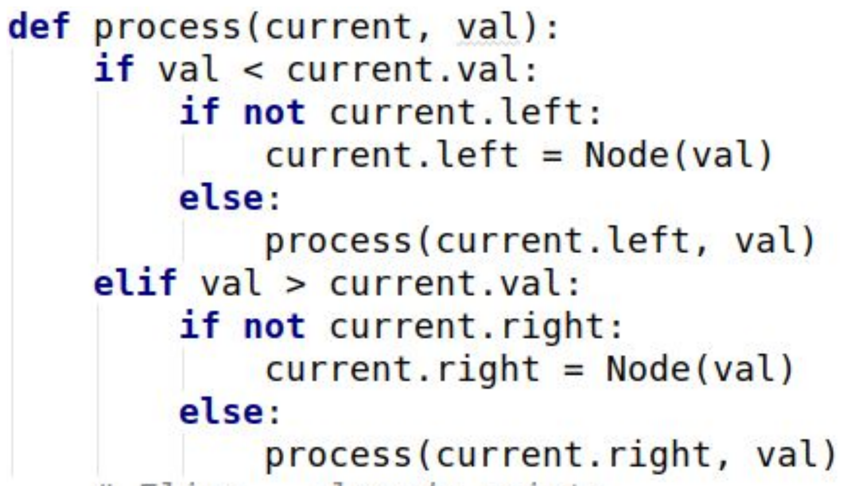


```python
def process(current, val):
    if val < current.val:
        if not current.left:
            current.left = Node(val)
        else:
            process(current.left, val)
    elif val > current.val:
        if not current.right:
            current.right = Node(val)
        else:
            process(current.right, val)
```

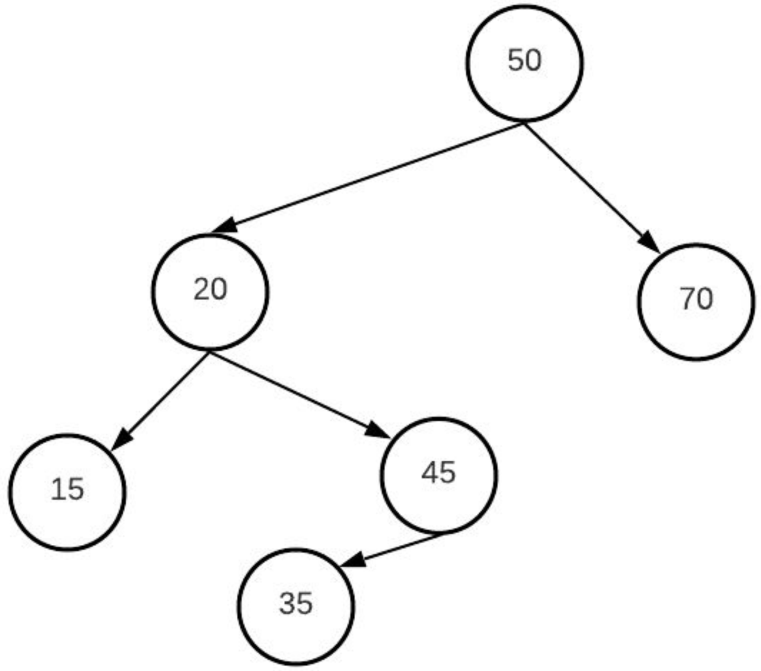


Insert 60

- At 50: go right
- At 70: go left
    - No left
    - Create left(60)

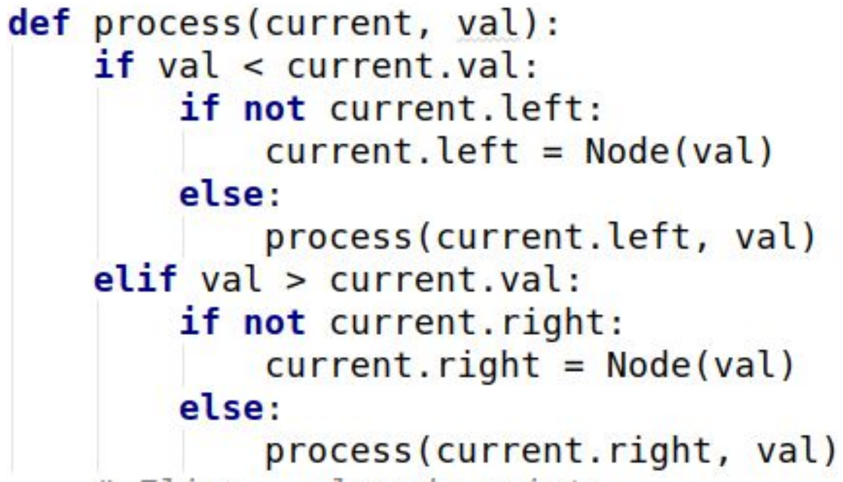


```python
def process(current, val):
    if val < current.val:
        if not current.left:
            current.left = Node(val)
        else:
            process(current.left, val)
    elif val > current.val:
        if not current.right:
            current.right = Node(val)
        else:
            process(current.right, val)
```

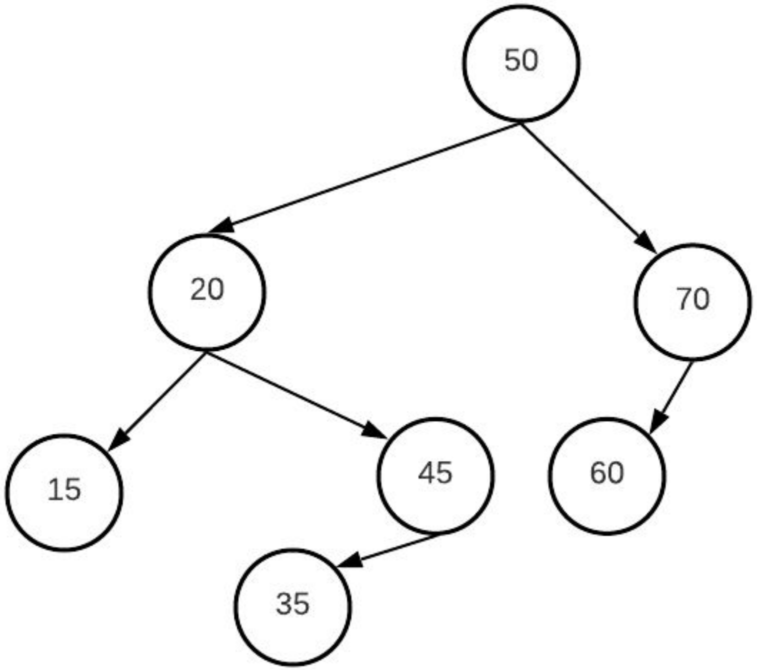


Insert 73

- At 50: go right
- At 70: go right
  - No right
  - Create right(73)

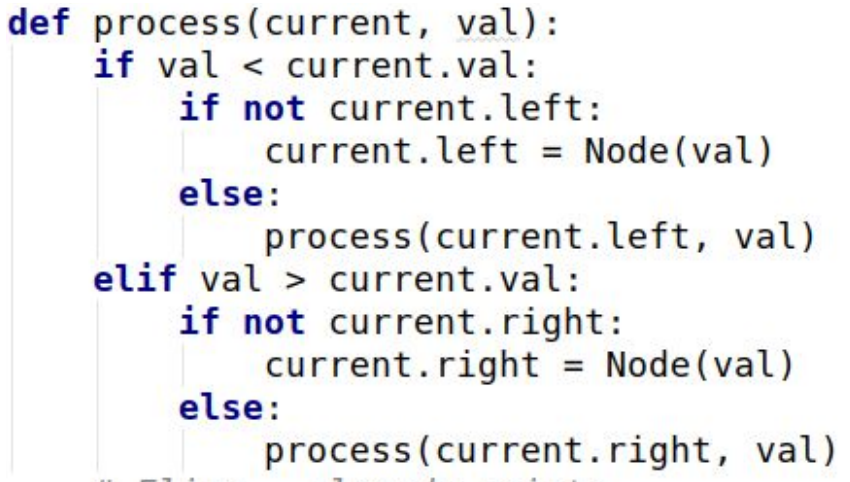


```
def process(current, val):
    if val < current.val:
        if not current.left:
            current.left = Node(val)
        else:
            process(current.left, val)
    elif val > current.val:
        if not current.right:
            current.right = Node(val)
        else:
            process(current.right, val)
```

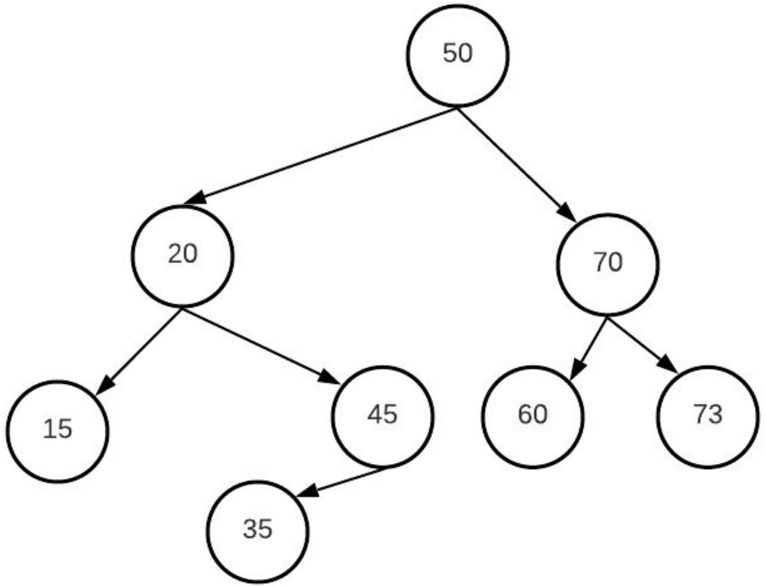


Insertion complexity

- O(h) time and memory
    - Time relates to the number of nodes/links in the 'chain' for an element
    - Memory: Auxiliary for stack

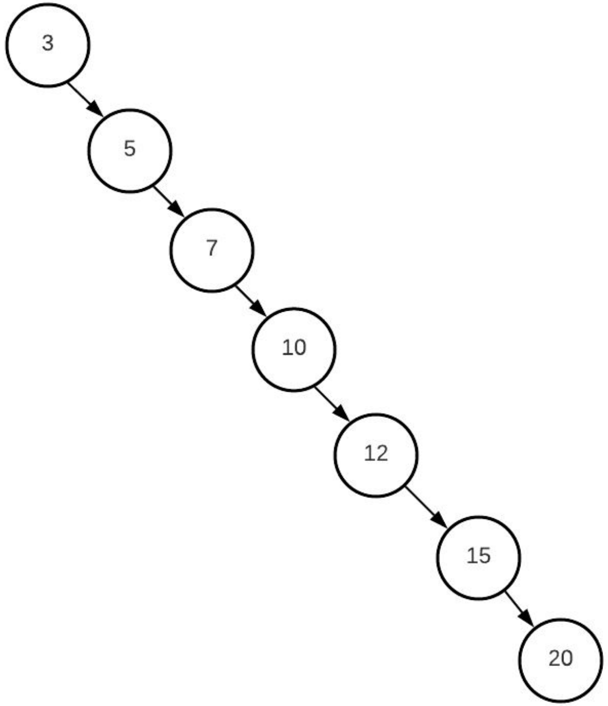


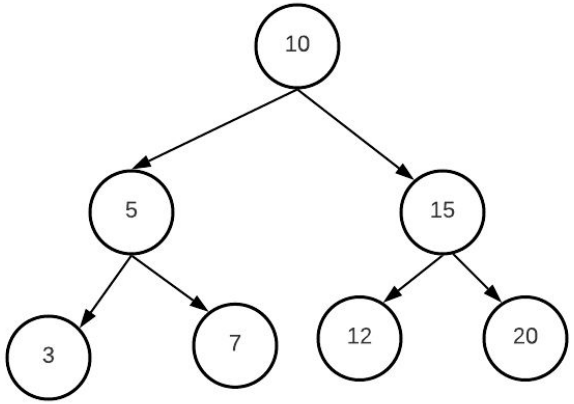
```
def process(current, val):
    if val < current.val:
        if not current.left:
            current.left = Node(val)
        else:
            process(current.left, val)
    elif val > current.val:
        if not current.right:
            current.right = Node(val)
        else:
            process(current.right, val)
```



Order of insertion

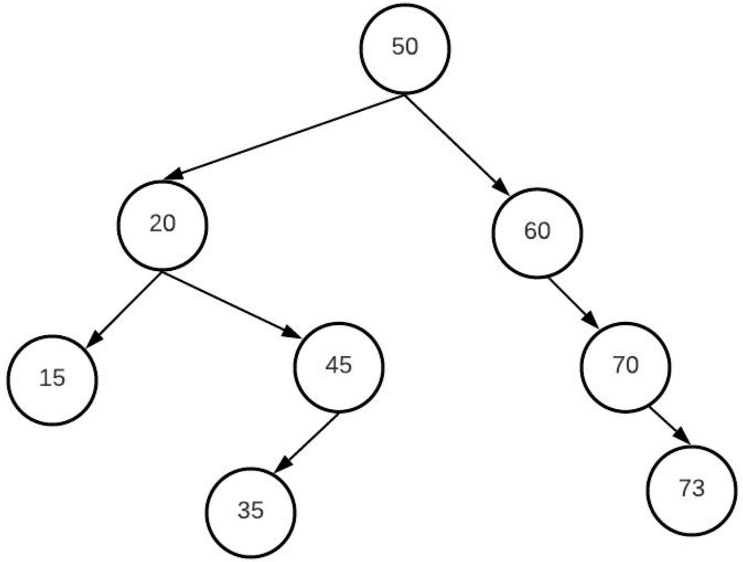
- Tree shape depends on **insertion order**
- In the **best case**, we get a balanced tree
- But in the **worst cast** it could be degenerate!
- Shape affects insertion/search time
  - From O(logn) to o(n)





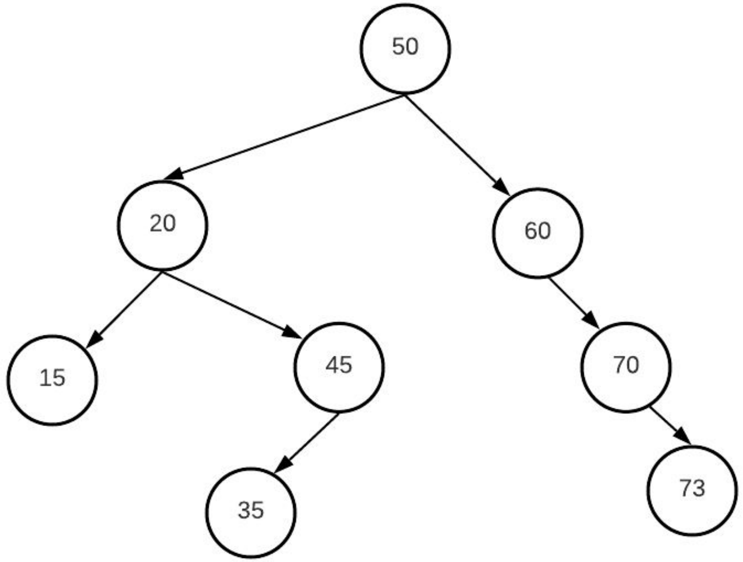
Minimum?

- Given a subtree root, what is the minimum value in it? max value?



Inorder **Successor** in Binary Search Tree?

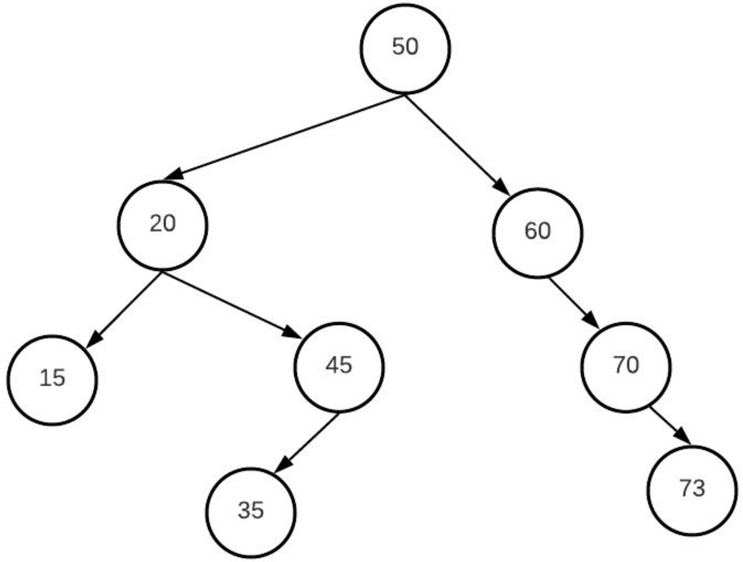
- Given node x, find node y that is the smallest y > x   [in O(h)]
- In other words: get inorder traversal
  - 15 20 35 45 50 60 70 73
  - It is the **next value** in the array
  - 70 ⇒ 73
  - 20 ⇒ 35
  - 45 ⇒ 50
  - 35 ⇒ 45
- Think for 15 min about 2 cases:
  - 1) x has right   2) x doesn't have right



Node deletion?

- Give yourself 20 minutes to think about how we can delete a node given a value
  - After deletion, the tree must remain a BST
  - Utilize the successor idea
- Consider 3 cases:

73	[no children]
60	[1 child]
20	[2 children]



“Acquire knowledge and impart it to the people.”

“Seek knowledge from the Cradle to the Grave.”