Data Structures

BST Homework 2

Mostafa S. Ibrahim
Teaching, Training and Coaching for 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)

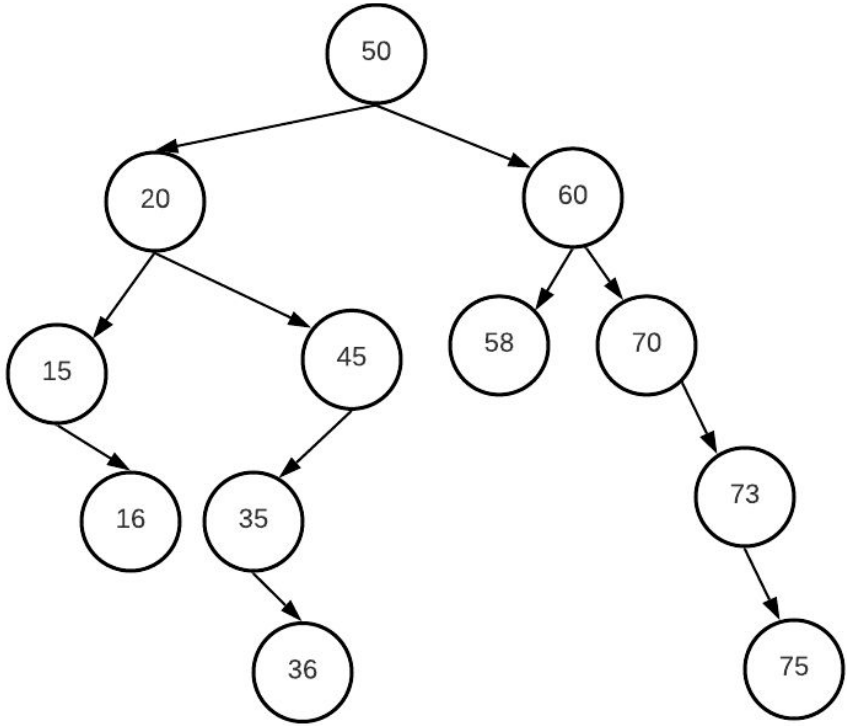
Problem #1: Parent Link

- In our implementation, we used an approach without parent links
- Rewrite the BST code for both the insert and successor functions, where your Node structure now includes the **parent link**

```
class Node:
    def __init__(self, val=None, parent=None, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right
        self.parent = parent
```

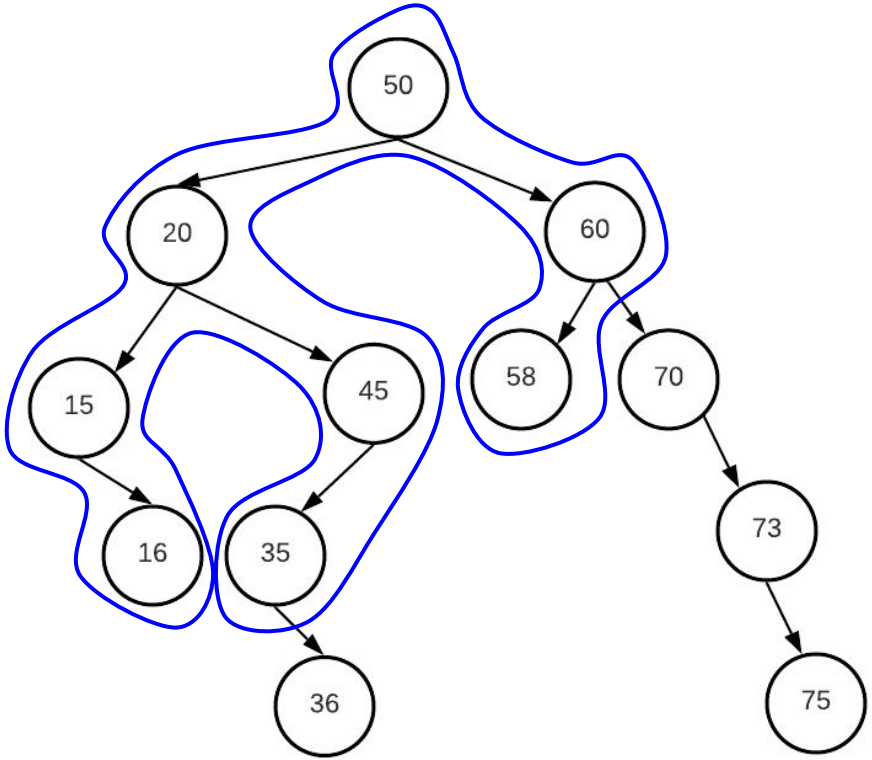
Problem #2: Queries of successors

- Assume we have a deque that contains **sorted** integers, and which we will use to find each item's successor
  - Input ⇒ {15, 20, 58}
  - Successors ⇒ {16, 35, 60}
  - *All the values in the deque are already in the tree and unique*
- Return a list of tuples, each contain (value, successor) for values that have successor

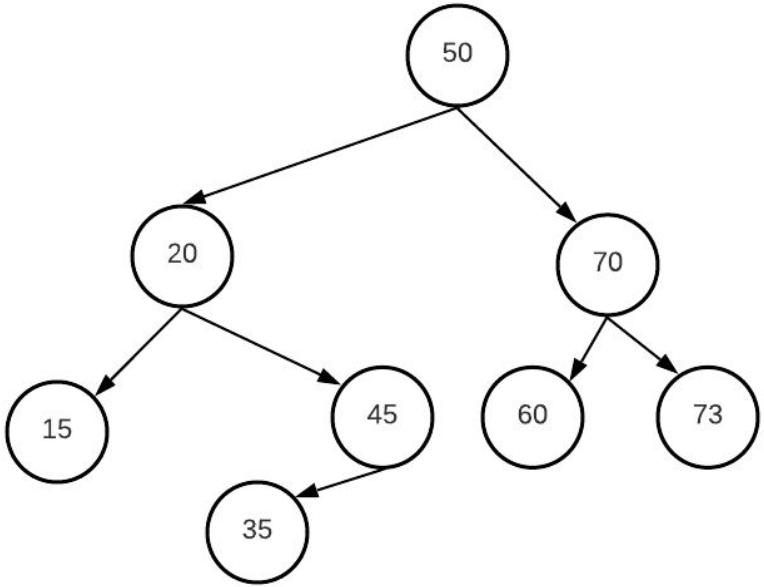


Problem #2: Queries of successors

- Develop a function that finds all of them, such that:
  - You don’t do complete traversal. **As in the lecture code, stop as early as possible!**
    - {15, 20, 58}
  - Don’t use the parent node
  - Don't retrieve a chain of ancestors, as in the lecture
- Tip:
  - We know that inorder traversal already moves towards our successor. Why don't we just 'catch' it once we find it?
  - The code is an adaptation or modification of the search function



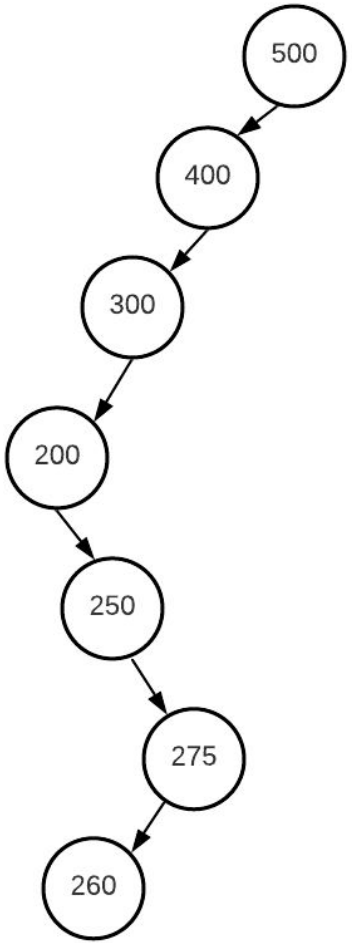
Another example




```
tree = BinaryTree(50)
lst = [20, 70, 15, 45, 60, 73, 35]
tree.insert(lst)

lst.append(50)
lst = sorted(lst)

import collections
nodes_queue = collections.deque()

for val in lst:
    nodes_queue.append(val)

print(tree.successor_queries(nodes_queue))
# [(15, 20), (20, 35), (35, 45), (45, 50),
# (50, 60), (60, 70), (70, 73)]
```

Problem #3: Is degenerate tree

- def is_degenerate(preorder)
- Given a **list** for preorder BST of N nodes, return true if it is degenerate tree of height N-1
- Do it in O(n).
  - 25, 8, 11, 13, 12 ⇒ True
  - 100, 70, 101 ⇒ False
  - 100, 70, 60, 75 ⇒ False
  - 100, 70, 60, 65 ⇒ True
  - 9, 8, 7, 6, 5, 4, 3 ⇒ True
  - 500, 400, 300, 200 , 250 , 275, 260 ⇒ True
  - 500, 400, 300, 200 , 250 , 275, 260, 280 ⇒ False



"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."