Data Structures

BST Deletion 2

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)

Deletion Implementation

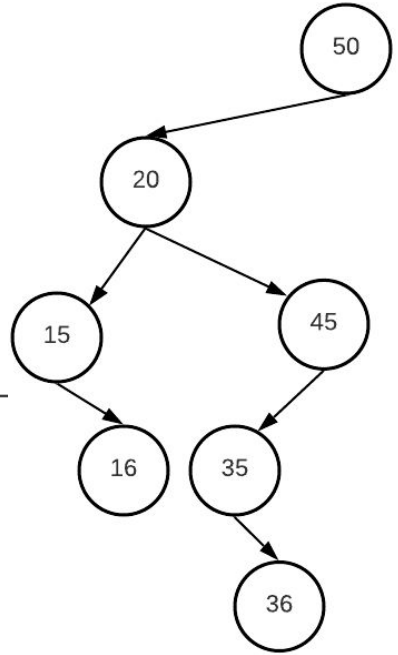
- Given that we need to find the node, then we do normal search procedure
- Then we need to identify the 3 possible cases and handle each one
- For 2 children case, we need to find the minimum in the right side
  - Copy data from it
  - Then remove this node itself
- Relinking
  - A critical coding step is to relink the current node to the **changed** subtree
  - E.g. node.left = change(node.left)
  - Keep think

```
def delete(self, val):
    def process(current, val):
        if not current:
            return

        if val < current.val:  # Value on the left side
            # the left subtree will be changed. This can be left itself
            current.left = process(current.left, val)   # must link
            return current

        if val > current.val:  # Value on the right side
            current.right = process(current.right, val)
            return current

        # we found the node: we have 3 cases
        if current.is_leaf():   # case 1: leaf
            return None         # Just remove
```

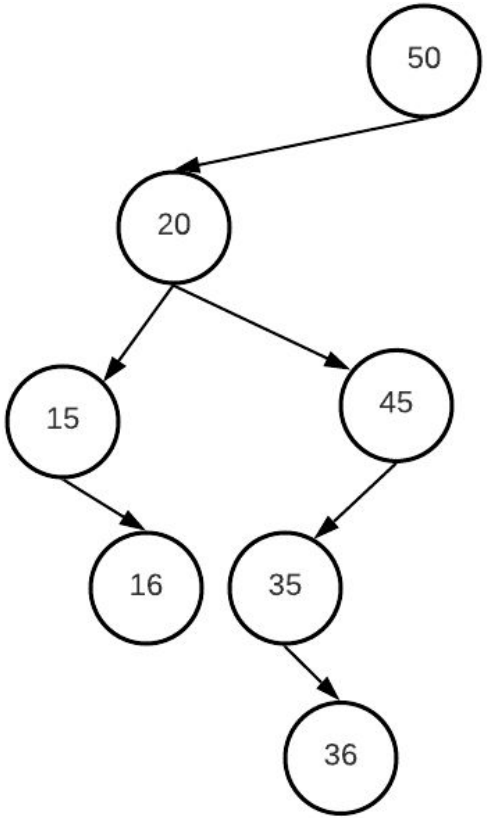


```
    if not current.right:       # case 2: has left only
        current = current.left
        return current

    if not current.left:        # case 2: has right only
        current = current.right
        return current

    # 2 children: Use successor
    mn = self.min_node(current.right)
    current.val = mn.val    # copy data
    current.right = process(current.right, mn.val)
    return current

process(self.root, val)
```



"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."