BST Insert Remove

Thursday, July 25, 2024 3:05 PM

Solution

```python
def minValueNode(root):
    curr = root
    while curr and curr.left:
        curr = curr.left
    return curr

def remove(root, val):
    if not root:
        return None

    if val > root.val:
        root.right = remove(root.right, val)
    elif val < root.val:
        root.left = remove(root.left, val)
    else:
        if not root.left:
            return root.right
        elif not root.right:
            return root.left
        else:
            minNode = minValueNode(root.right)
            root.val = minNode.val
            root.right = remove(root.right, minNode.val)
    return root
```

find min value

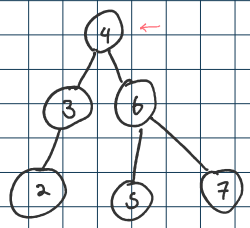
search for node

for single child, return the child that exists

two child algorithm

Algorithm

Ex. Remove 4

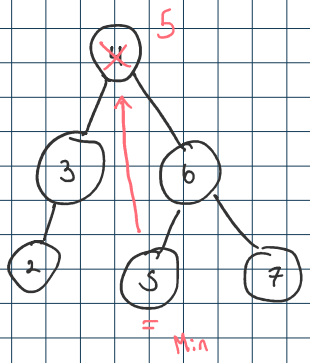


1. Search through BST until on node
   - → check if 0 or 1 child

   false ↓

   - → Execute case for 2 children

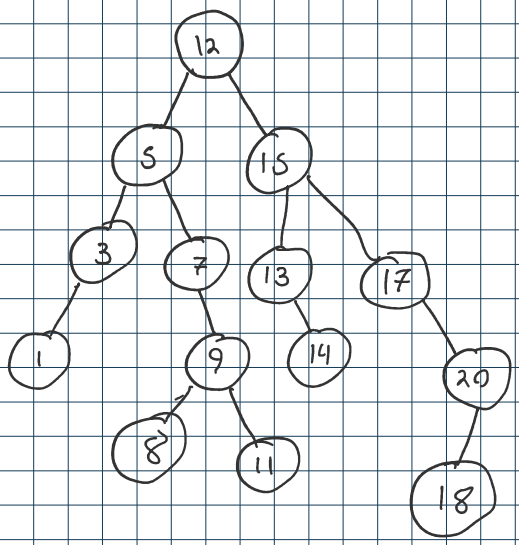
2. Find min node of right sub tree, replace the node you want to delete with that node



3. Recursively call the remove() function on the right subtree to re adjust the nodes to balance the BST

Example

Delete node with 2 children



Let's try deleting the node (15)

Solution

1. find the minimum node in the right subtree of (15), this can be identified as (17)

2. Replace (15) with 17, then recursively call the algorithm again on the right subtree to remove (17)

Active Recall + Corrections

Delete Node Cases:

Case 1: 0 children

1. check if left child exist, if not, point to righ child

Case 2: 1 child

1. same logic as case 1

case 3 : 2 children

1. find min on right subtree of root.
2. Replace root with this value
3. recursively call function on right subtree

```python
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
#         self.left = left
#         self.right = right
class Solution:
    def minValue(self, root):
        curr = root
        while curr and curr.left:
            curr = curr.left
        return curr
    def deleteNode(self, root: Optional[TreeNode], key: int) -> Optional[TreeNode]:
        if not root:
            return None

        if key > root.val:
            self.deleteNode(root.right, key)
        elif key < root.val:
            self.deleteNode(root.left, key)
        else:
            if not root.right:
                return root.left
            elif not root.left:
                return root.right
            else:
                min = self.minValue(root.right)
                root = min
                self.deleteNode(root.right, min)
        return root
```

Missing this before

root.right = this line

Same logic with this line

this should be root.val = min.val

also on this line set root.right = this line