HARDIK KHARE | 70765344

HW 5.1 (MANDATORY 50 points)

Implement an AVL tree whose node values are integers. It should be able to support three operations: ***insert***, ***find***, and ***delete***. Implement any replacement operations to use the **predecessor**.

Input:

- a sequence of commands to stdin that ***insert***, ***find***, or ***delete*** an integers.

Output:

- for each command executed, a list of the node-values within the tree that are traversed to find the initial position in the tree to begin to process the command.
  - for ***insert*** it will be the list of values traversed before initially inserting the new value, followed by the value inserted
  - for ***find***, it will be the list of nodes traversed before finding the value we are looking for or before being able to assert the value is not present in the tree. Asserts whether value was found or not.
  - for ***delete***, it will be the list of nodes traversed before finding the node to be deleted or before being able to assert the value is not present in the tree. Asserts whether the node to be deleted was found or not.

Section 1: Successful compilation of program

```
hkhare@circinus-5 02:26:19 ~/hw5
$ javac -Xlint AVLTree.java
hkhare@circinus-5 02:26:43 ~/hw5
$
```

Section 2: program running on the provided example from the assignment

hardikkhare — hkhare@circinus-21:~/hw5 — ssh hkhare@openlab.ics.uci.edu

```
Enter choice (Insert/Delete/Find):
insert 50
50 (inserted)
Enter choice (Insert/Delete/Find):
insert 25
50 25 (inserted)
Enter choice (Insert/Delete/Find):
insert 10
50 25 10 (inserted)
Enter choice (Insert/Delete/Find):
insert 5
25 10 5 (inserted)
Enter choice (Insert/Delete/Find):
insert 7
25 10 5 7 (inserted)
Enter choice (Insert/Delete/Find):
insert 3
25 7 5 3 (inserted)
Enter choice (Insert/Delete/Find):
insert 30
7 25 50 30 (inserted)
Enter choice (Insert/Delete/Find):
insert 20
7 25 10 20 (inserted)
Enter choice (Insert/Delete/Find):
insert 8
7 25 10 8 (inserted)
Enter choice (Insert/Delete/Find):
insert 15
7 25 10 20 15 (inserted)
Enter choice (Insert/Delete/Find):
find 10
10 (found)
Enter choice (Insert/Delete/Find):
find 12
10 25 20 15  (not found!)
Enter choice (Insert/Delete/Find):
delete 4
10 7 5 3  (not found!)
Enter choice (Insert/Delete/Find):
delete 20
10 25 20  (deleted)
Enter choice (Insert/Delete/Find):
find 22
10 25 15  (not found!)
Enter choice (Insert/Delete/Find):
delete 50
10 25 50  (deleted)
Enter choice (Insert/Delete/Find):
find 30
10 25 30 (found)
Enter choice (Insert/Delete/Find):
delete 10
10  (deleted)
Enter choice (Insert/Delete/Find):
find 7
8 5 7 (found)
```

Section 3: Provided test input

~ No Test Input provided on Piazza ~

Section 4: Edge Case #1
Description: Delete value from empty tree

Input: delete 10

Expected Output: Nothing will be deleted and program should not throw any error

```
hkhare@circinus-5 02:26:43 ~/hw5
$ java AVLTree

Enter choice (Insert/Delete/Find):
delete 10
 (not found!)
```

Section 5: Edge Case #2
Description: Insert duplicate node

Input:
Insert 5
Insert 5

Expected Output: Duplicate value should not be inserted

Output:

```
Enter choice (Insert/Delete/Find):
insert 5
5 (inserted)
Enter choice (Insert/Delete/Find):
insert 5
5
```

Section 6: Edge Case #3
Description: Find a missing node

Input:
insert 5
insert 5
find 4

Expected Output: Not Found!

Output

```
Enter choice (Insert/Delete/Find):
insert 5
5 (inserted)
Enter choice (Insert/Delete/Find):
insert 5
5
Enter choice (Insert/Delete/Find):
find 4
5  (not found!)
```

Section 7: Edge Case #4
Description: Finding a deleted node

Input:
Insert 5
Insert 6
Insert 4
delete 5
find 5

Expected Output: Not found!

Output:

```
Enter choice (Insert/Delete/Find):
insert 5
5 (inserted)
Enter choice (Insert/Delete/Find):
insert 5
5
Enter choice (Insert/Delete/Find):
find 4
5  (not found!)
Enter choice (Insert/Delete/Find):
insert 6
5 6 (inserted)
Enter choice (Insert/Delete/Find):
insert 4
5 4 (inserted)
Enter choice (Insert/Delete/Find):
delete 5
5  (deleted)
Enter choice (Insert/Delete/Find):
find 5
4 6  (not found!)
```

https://leetcode.com/problems/binary-tree-pruning/

LeetCode Explore Problems Interview Contest Discuss Store

LeetCoding Challenge + GIVEAWAY!

harry3997
My List
My Playground
Notebook
Submissions
Sessions
Progress
Points
Subscription
Orders
Sign out

Binary Tree Pruning

Submission Detail

30 / 30 test cases passed.

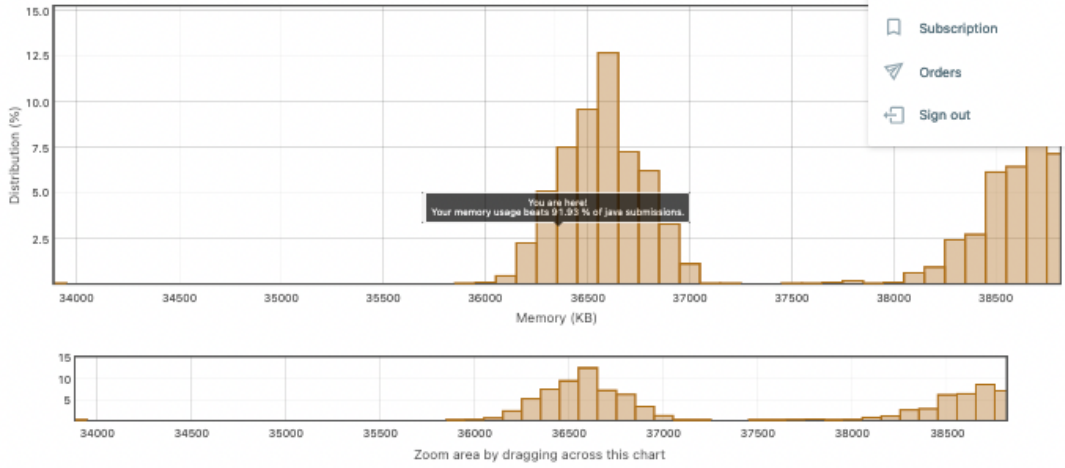
Runtime: **0 ms**

Memory Usage: **36.3 MB**

Accepted Solutions Runtime Distribution

Sorry. We do not have enough accepted submissions to show distribution chart.

Accepted Solutions Memory Distribution



Invite friends to challenge **Binary Tree Pruning**

Submitted Code: 1 day, 4 hours ago

Language: java

Edit Code

```java
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode() {}
 *     TreeNode(int val) { this.val = val; }
 *     TreeNode(int val, TreeNode left, TreeNode right) {
 *         this.val = val;
 *         this.left = left;
 *         this.right = right;
 *     }
 * }
 */
class Solution {
    public int postOrder(TreeNode root) {
        if(root==null) return 0;
        int a = postOrder(root.left);
        int b = postOrder(root.right);

        if(a==0) root.left=null;
        if(b==0) root.right=null;

        if(a==1||b==1||root.val==1)  return 1;
        return 0;
    }

    public TreeNode pruneTree(TreeNode root) {
        int v = postOrder(root);
        if(v==0) return null;
        return root;
    }
}
```

https://leetcode.com/problems/maximum-product-of-splitted-binary-tree/

LeetCode Explore Problems Interview Contest Discuss Store

LeetCoding Challenge + GIVEAWAY!

harry3997
My List
My Playground
Notebook
Submissions
Sessions
Progress
Points
Subscription
Orders
Sign out

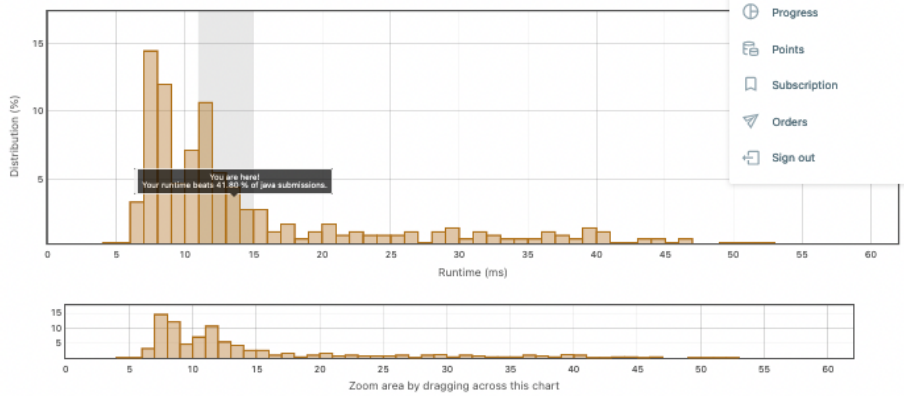
Maximum Product of Splitted Binary Tree

Submission Detail

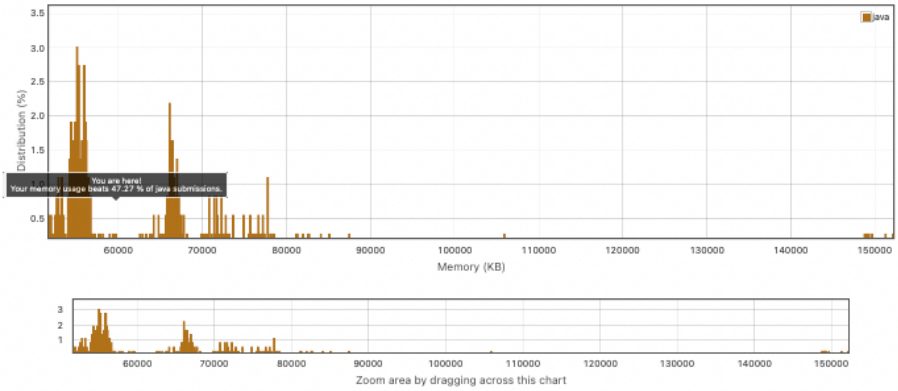
54 / 54 test cases passed.
Runtime: 13 ms
Memory Usage: 59.6 MB

Submitted

Accepted Solutions Runtime Distribution



Accepted Solutions Memory Distribution



Invite friends to challenge **Maximum Product of Splitted Binary Tree**

Submitted Code: 22 hours, 54 minutes ago

Language: java

Edit Code

```java
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode() {}
 *     TreeNode(int val) { this.val = val; }
 *     TreeNode(int val, TreeNode left, TreeNode right) {
 *         this.val = val;
 *         this.left = left;
 *         this.right = right;
 *     }
 * }
 */
class Solution {
    public long aux(TreeNode root, ArrayList<Long> pq) {
        if(root==null) return 0;
        long v=root.val+aux(root.left,pq)+aux(root.right,pq);
        pq.add(v);
        return v;
    }

    public int maxProduct(TreeNode root) {
        long sum=0,mod=1000000007;
        ArrayList<Long> pq = new ArrayList<>();
        sum=aux(root,pq);
        long rest=0,v=0;int i=0;
        while(i<pq.size()){
            long x=pq.get(i++);
            v=(x*(sum-x));
            rest=Math.max(rest,v);
        }
        return (int)(rest%mod);
    }
```