Lab Assignment 8

Binary Search Trees and Heap

1. Write program using functions for binary tree traversals: Pre-order, In-order and Post order using recursive approach.

2. Implement following functions for Binary Search Trees

(a) Search a given item (Recursive & Non-Recursive)

(b) Maximum element of the BST

(c) Minimum element of the BST

(d) In-order successor of a given node the BST

(e) In-order predecessor of a given node the BST

3. Write a program for binary search tree (BST) having functions for the following operations:

(a) Insert an element (no duplicates are allowed),

(b) Delete an existing element,

(c) Maximum depth of BST

(d) Minimum depth of

4. Write a program to determine whether a given binary tree is a BST or not.

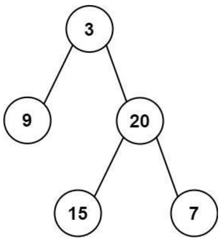
5. Implement Heapsort (Increasing/Decreasing order).

6. Implement priority queues using heaps.

Additional Questions

Q1. Given the root of a binary tree, return the sum of all left leaves. A leaf is a node with no children. A left leaf is a leaf that is the left child of another node.

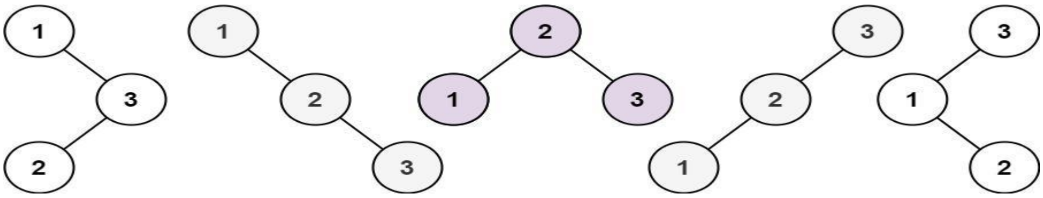
Example:



Input: root = [3,9,20,null,null,15,7]
Output: 24
Explanation: There are two left leaves in the binary tree, with values 9 and 15 respectively.

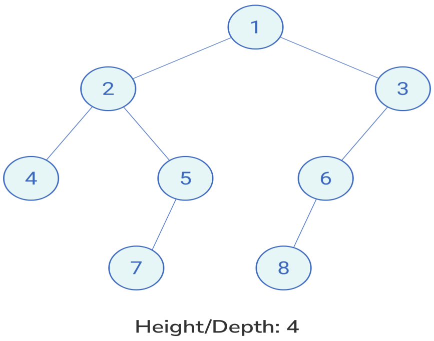
Link: Sum of Left Leaves - LeetCode

Q2. Given an integer n, return all the structurally unique BST's (binary search trees), which has exactly n nodes of unique values from 1 to n. Return the answer in any order.
Example 1:



Input: n = 3
Output: [[1,null,2,null,3],[1,null,3,2],[2,1,3],[3,1,null,null,2],[3,2,null,1]]
Example 2:
Input: n = 1
Output: [[1]]
Constraints:
1 <= n <= 8
Link: 95. Unique Binary Search Trees II - Medium | DSA Corner | Talentd
Companies: Microsoft, Adobe, Bloomberg, Uber

Q3. Given a binary tree, return its maximum depth. The depth of a binary tree is the number of nodes from the root node to any of the leaf nodes. The maximum depth is the maximum of the depths across all the paths.
Example:

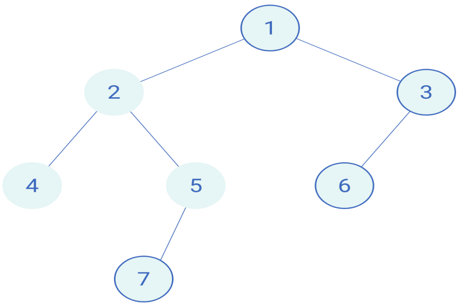


Testing
Input Format
The first line contains an integer T denoting the number of test cases.
For each test case, the input has 2 lines:
The first line contains an integer n denoting the number of nodes in the tree (including the NULL nodes).
The second line contains n space-separated integers that will form the binary tree. The integers follow level order traversal of the tree where -1 indicates a NULL node.
Output Format
For each test case, the output has an integer denoting the maximum depth of the tree.

Sample Input
5
12

1 2 3 4 5 6 -1 -1 -1 7 -1 8
7
1 2 -1 4 -1 5 6
7
8 -1 9 -1 10 11 12
5
28 14 11 -1 48
1
6
Expected Output
4
4
4
3
1
Link: Maximum Depth of Binary Tree | Practice Problem
Companies: Amazon, Facebook, Microsoft

Q4. There are different ways to look at a binary tree. The right view of a binary tree contains the set of nodes that will be visible if you look at the binary tree from the right side. Given the root node of a binary tree, return an array containing the node elements in the right view, from top to bottom.



Right View: [1, 3, 6, 7]

Testing
Input Format
The first line contains an integer T denoting the number of test cases.
For each test case, the input has 2 lines:
The first line contains an integer n denoting the number of nodes in the tree (including the NULL nodes).
The second line contains n space-separated integers that will form the binary tree. The integers follow level order traversal of the tree where -1 indicates a NULL node.
Output Format
For each test case, the output contains a line with space-separated integers representing the right view of the binary tree.

Sample Input
6
7
1 2 -1 4 -1 5 6

```
3
6 -1 4
7
8 -1 9 -1 10 11 12
5
28 14 11 -1 48
1
6
7
3 -1 2 1 5 -1 6
```

Expected Output

```
1 2 4 6
6 4
8 9 10 12
28 11 48
6
3 2 5 6
```

Link: [Right View of Binary Tree | Practice Problem]()

Companies: Adobe, Amazon, Apple, Uber

Q5. Given preorder and in-order traversals, write a program to construct the Binary Tree.

Q6. Given in-order and post-order traversals, write a program to construct the Binary Tree.

Q7. Write a program to merge two BSTs into a doubly-linked list in sorted order.

Example Input:

```
  20
T1   / \
 10   30
   / \
 25  100
  50
T2   / \
  5    70
```

Output:

5 <—> 10 <—> 20 <—> 25 <—> 30 <—> 50 <—> 70 <—> 100 <—> null