CS204(2025)

Tree

Instructions for Implementation

- Write the program in C or C++ using standard input/output.
- Follow the input/output format strictly.
- Ensure your code:
- Handles edge cases correctly.
- Meets the time and space complexity constraints.
- Do not use STL containers like vector, set, or map in C++. Use raw arrays unless explicitly allowed.
- You are encouraged to write clean, modular, and well-documented code.

Tree Questions:

- **Instructions :** All input is given as their level order traversal from left to right. You have to construct the tree from the level order traversal and do the program.

1. For the purposes of this challenge, we define a binary tree to be a binary search tree with the following ordering requirements:
   - The value of every node in a node's left subtree is *less than* the data value of that node.
   - The value of every node in a node's right subtree is *greater than* the data value of that node.

   Given the root node of a binary tree, can you determine if it's also a binary search tree?

   ## Input Format:

   - First line is the number of nodes (n) (non-null)
   - -1 indicate null node

   ## Output:

- T → if the given tree satisfies **all BST properties**.
- F→ if it violates BST rules anywhere.

Constraints:

Time Complexity = O(n) : Every node is visited only once

Test Cases:

1. Input :
   1
   10
   Output: T
2. Input :
   6
   10 5 15 2 7 -1 20
   output: T

2. You are given n value representing the nodes of a Binary Search Tree and the two values v1 and v2. Construct BST fiven the node value and print the Lowest Common Ancestor (LCA) of v1 and v2 in the Binary Search Tree.

   Note: If both the values are same then node itself is the LCA.

Input format: The first line contains an integer 'n', the number of nodes in the tree.
The second line contains n space-separated integers representing "node.data" values.
The third line contains two space-separated integers, v1 and v2.

Output Format : Return the a pointer to the node that is the lowest common ancestor of v1 and v2.

Test Cases:

1. Input :
   6
   4 2 3 1 7 6
   1 7

   Output : 4

2. Input:
   7
   5 3 8 2 4 7 9
   2 4

   Output : 3

3. You are given n integers corresponding to the node values of a binary search tree and values to be inserted into the tree. Insert the values into their appropriate position in the binary search tree and print the in-order traversal of the BST.

Input format:

a. The first line contains an integer **n** → the number of values to be inserted into the BST.

b. The next **n** integers represent the values to be inserted in the order they should appear.

Output Format:

Print the **in-order traversal** of the final BST after all insertions.
(In-order is standard for BST problems because it gives sorted order, which makes checking correctness easier.)

Test Cases:

1. Input:
   6
   4 2 3 1 7 6
   Output: 1 2 3 4 6 7

2. Input:
   5
   10 5 15 3 7
   Output:
   3 5 7 10 15

4) Postorder Traversal

Problem:
Given a binary tree, print its postorder traversal (Left → Right → Root).

Input Format:

- First line: Integer n (number of non-null nodes)
- Second line: n space-separated integers representing the node values in level order (using -1 for NULL nodes).

Output Format:

- Print the postorder traversal in a single line, space-separated.

Constraints:

- $1 \leq n \leq 10^4$
- Node values: integers in range of 32-bit signed int
- Time Complexity: O(n)
- Space Complexity: O(h), where h = tree height

Input:

6
1 2 3 4 5 -1 6
Output:
4 5 2 6 3 1

Input:

3
10 20 30

Output:

20 30 10

5) Path sum

Problem:
Given a binary tree and an integer target, return YES if the tree has a root-to-leaf path such that adding up all the values along the path equals target, otherwise return NO.

Input Format:

- First line: Integer n (number of nodes(non-null))
- Second line: n space-separated integers representing the node values in level order (using -1 for NULL nodes).
- Third line: Integer target

Output Format:

- Print YES if such a path exists, otherwise NO.

Constraints:

- $1 \leq n \leq 10^4$
- Node values: integers in range of 32-bit signed int
- Time Complexity: O(n)
- Space Complexity: O(h), where h = tree height

Input 1:

```
9
5 4 8 11 -1 13 4 7 2 -1 -1 -1 -1 -1 1
22
```

Output 1:

```
YES
```

Input 2:

```
3
1 2 3
5
```

Output 2:

NO

6) Height Balanced or Not

Problem:
Determine if a binary tree is height-balanced. A height-balanced tree is defined as a binary tree where the depth of the two subtrees of every node never differs by more than 1.

Input Format:

- First line: Integer n (number of nodes(non-null))
- Second line: n space-separated integers representing the node values in level order (using -1 for NULL nodes).

Output Format:

- Print YES if the tree is balanced, else NO.

Constraints:

- $1 \leq n \leq 10^4$
- Node values: integers in range of 32-bit signed int
- Time Complexity: O(n)
- Space Complexity: O(h), where h = tree height

Input :

5
3 9 20 -1 -1 15 7

Output:

YES

Input:

3
1 2 -1 3

Output:

NO

7 Problem:

Find the diameter of a binary tree. The diameter is the length of the longest path between any two nodes in a tree. This path may or may not pass through the root.

Input Format:

- First line: Integer n (number of nodes(non-null))
- Second line: n space-separated integers representing the node values in level order (using -1 for NULL nodes).

Output Format:

- Print a single integer, the diameter of the tree.

Constraints:

- $1 \leq n \leq 10^4$
- Node values: integers in range of 32-bit signed int
- Time Complexity: O(n)
- Space Complexity: O(h), where h = tree height

Input:

5
1 2 3 4 5

Output:

3

Input:

3
1 2 3

Output:

2

8.

Problem:
Given a binary tree, create its mirror (left and right children of all nodes are swapped). Print the level order traversal(left to right) of the mirrored tree.

Input Format:

- First line: Integer n (number of nodes)
- Second line: n space-separated integers representing the node values in level order (using -1 for NULL nodes).

Output Format:

- Print the level order traversal of the mirrored tree as space-separated integers.
- Make sure to use -1if a node is absent in the mirrored tree.

Constraints:

- $1 \leq n \leq 10^4$
- Node values: integers in range of 32-bit signed int
- Time Complexity: O(n)
- Space Complexity: O(h), where h = tree height

Input:

7
1 2 3 4 5 6 7

Output:

1 3 2 7 6 5 4

Input:

3
10 20 30

Output:

10 30 20