Navigating the World of Binary Trees

Binary Tree Node 🌳 vs. Linked List Node 🔗

Think of data structures as building blocks:
🔗 Linked List Node:

- It's like a train compartment. Each compartment knows only about the one before it and the one after it.

🌳 Binary Tree Node:

- Picture a fork in the road. This node knows about two paths: left and right, but it doesn't look back to where it came from.

That's the essence! One is linear like a train, and the other branches out like roads.

Exploring the Binary Tree 🌳

🔍 Using Recursion:

- 🚶 Pre_Order: Visit root, then left, then right.
- 🚶 In_Order: Visit left, then root, then right.
- 🚶 Post_Order: Visit left, then right, then root.
- 🕐 Time & Space: o(n) (Imagine visiting every room in a house)

🔍 Using a Queue:

- 🚌 Level Order Traversal: Like checking each floor of a building, one by one.
- 🕐 Time & Space: o(n) (Same as above, every room in the house!)

Journey through the tree, whether by foot or by bus, and you'll spend about the same amount of time!

Key Techniques for Binary Tree Problems 🌳

1️⃣ Backtrack the Values 🔙:

- Use recursion to navigate the tree. This approach helps you bring values up to the root, much like retracing your steps to find something you've missed.

2️⃣ Transform as Needed 🛠️:

- View the tree as a flexible puzzle. You can rearrange nodes or adjust their connections, making problem-solving more manageable.

Keep these tools in your toolkit, and tree challenges will feel a lot smoother! 🌟

Hop in to the questions

Q1. Given a non-empty binary tree, find the maximum path sum. For this problem, a path is defined as any sequence of nodes from some starting node to any node in the tree along the parent-child connections. The path must contain at least one node and does not need to go through the root.(leet code: https://leetcode.com/problems/binary-tree-maximum-path-sum/)

Input1

```
    -10
    / \
   9   20
      / \
     15   7
```

Output: 42
Explanation: The optimal path is 15 -> 20 -> 7 with a sum of 42.

Input2

```
    -10
    /  \
  -20   -5
     /  \
    -2    6
```

Output: 6
Explanation: Sum of any other parts would reduce the overall total

Q2. Given the root of a binary tree, flatten the tree into a "linked list":

- The "linked list" should use the same TreeNode class, but all the right child references should be set, and all left child references should be null.
- The "linked list" should be in the same order as a pre-order traversal of the binary tree.
- Leet code:https://leetcode.com/problems/flatten-binary-tree-to-linked-list/

Input1

```
   1
  / \
 2   5
/ \   \
3   4   6
```

Output:

```
 1
 \
  2
   \
    3
     \
      4
       \
        5
         \
          6
```

Note:

- After flattening the tree, the pre-order traversal of the tree should return nodes in the sequence: 1,2,3,4,5,6.

Constraints:

- The number of nodes in the tree is in the range [0, 2000].
- -100 <= Node.val <= 100

Q3. Given the root of a binary tree, the value of a target node target, and an integer k, return *an array of the values of all nodes that have a distance* k *from the target node.*

You can return the answer in **any order**.
Leet code: https://leetcode.com/problems/all-nodes-distance-k-in-binary-tree/

Input1

```
    3
   / \
  5   1
 / \ / \
6  2 0  8
   / \
  7  4
```

root = [3,5,1,6,2,0,8,null,null,7,4], target = 5, k = 2

Output: [7, 1]

Input2

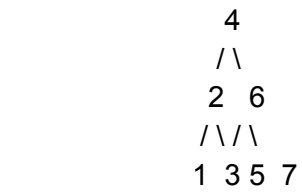
root = [1], target = 1, k = 3
Output: []

Constraints:

- The number of nodes in the tree is in the range [1, 500].
- 0 <= Node.val <= 500
- All the values of Node.val are **unique**.
- target is the value of one of the nodes in the tree.
- 0 <= k <= 1000

Q4. Given the root of a binary tree, write a function to perform an in-order traversal without using recursion. Return the values of the nodes in the order they were traversed.

Input



Output:
[1, 2, 3, 4, 5, 6, 7]

Note:
An in-order traversal visits the left subtree, then the current node, and finally the right subtree.
Constraints:

- The number of nodes in the tree is in the range [0, 5000].
- -10^4 <= Node.val <= 10^4