Maximum Sum of Non-Adjacent Nodes GFG

Given a binary tree with a value associated with each node, we need to choose a subset of these nodes such that sum of chosen nodes is maximum under a constraint that no two chosen node in subset should be directly connected that is, if we have taken a node in our sum then we can’t take its any children or parents in consideration and vice versa.

Example:

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

Output: The Maximum Sum of Non-Adjacent Nodes: 16 (1 + 4 + 5 + 6)

Approach 1: Function to return the maximum sum of non-adjacent nodes

- Define a helper function **solve** that returns a pair of integers: the first element represents the maximum sum including the current node, and the second element represents the maximum sum excluding the current node.
- In the helper function:
    - Base case: If the node is null, return a pair (0, 0).
    - Recursively calculate the maximum sums for the left and right subtrees using **solve**.
    - Calculate the maximum sum including the current node as the sum of the current node's value, the second element of the left subtree's result, and the second element of the right subtree's result.
    - Calculate the maximum sum excluding the current node as the maximum of the first and second elements of the left subtree's result plus the maximum of the first and second elements of the right subtree's result.
- In the **getMaxSum** function:
    - If the root is null, return 0.
    - Call the **solve** function to obtain the maximum sums.
    - Return the maximum of the first and second elements of the result pair as the maximum sum of non-adjacent nodes.

- Time Complexity: O(N) as it visits each node exactly once.
- Space Complexity: O(H) where H is the height of the tree (stack space for recursion).