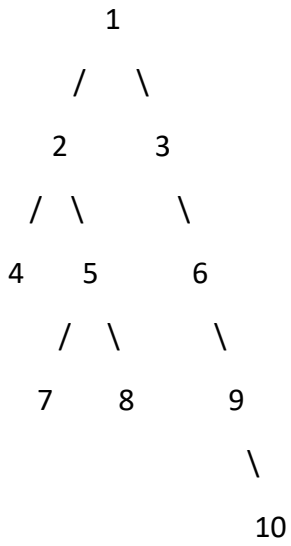
Minimum Time Required to Burn Binary Tree GFG

Given a binary tree and a **node data** called **target**. Find the minimum time required to burn the complete binary tree if the target is set on fire. It is known that in 1 second all nodes connected to a given node get burned. That is its left child, right child, and parent.
Note: The tree contains unique values.

Example:



TargetNode: 8

Output: The Minimum Time to Burn the Tree from TargetNode: 8: 7

Approach 1: function to find the minimum time for the tree to burn from the target node

1. Implement a helper function **parentMapping** to create a parent mapping of the binary tree using Breadth-First Search (BFS).
    - Create a queue for BFS and initialize the parent of the root as nullptr.
    - Traverse the tree while mapping each node to its parent node.
    - If the target node is found, return it.
2. Implement a second helper function **burnTreeHelper** to calculate the time required for the tree to burn using BFS.
    - Maintain a map to track visited nodes and a queue for BFS.
    - Initialize a variable **timeRequired** to store the time.
    - Mark the target node (root) as visited and add it to the queue.
    - While nodes are still burning at the current level:

        - Process nodes at the current level, checking left child, right child, and parent node.
        - If any nodes are still burning at this level, increment the time.
    - Return the total time required for the tree to burn.
3. In the main function **minBurnTime**:
    - Create a parent mapping of the tree using the **parentMapping** function.
    - Calculate the time required to burn the tree using the **burnTreeHelper** function from the target node.
4. Handle the case when the tree is empty and return the calculated time.

Time Complexity:

1. Creating Parent Mapping (**parentMapping** function):
    - The BFS traversal of the tree takes O(N) time, where N is the number of nodes in the tree.
    - During BFS, we map each node to its parent node.
    - Overall, creating the parent mapping takes O(N) time.
2. Calculating Burn Time (**burnTreeHelper** function):
    - We use BFS to traverse the tree from the target node and calculate the burn time.
    - In the worst case, we may visit all nodes of the tree.
    - Thus, calculating the burn time also takes O(N) time.
3. Overall Time Complexity (Main Function):
    - The **minBurnTime** function first creates the parent mapping (O(N)) and then calculates the burn time (O(N)).
    - **The overall time complexity is O(N).**

Space Complexity:

1. Creating Parent Mapping (**parentMapping** function):
    - We use additional data structures to store the parent mapping.
    - The space required for the queue, map for parent mapping, and variables is O(N).
2. Calculating Burn Time (**burnTreeHelper** function):
    - We use additional data structures to store visited nodes and the queue for BFS.

- The space required for the queue, map for visited nodes, and variables is O(N).

3. Overall Space Complexity (Main Function):
    - The **minBurnTime** function combines the space used for parent mapping and burn time calculation.
    - **The overall space complexity is O(N).**