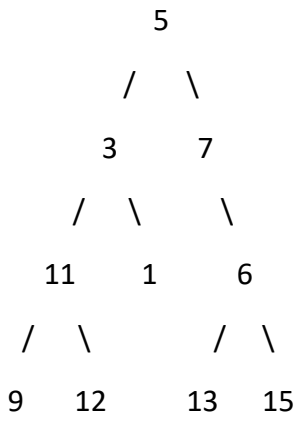
Right View of Binary Tree LeetCode

Given the root of a binary tree, imagine yourself standing on the **right side** of it, return *the values of the nodes you can see ordered from top to bottom*.

Example:



Output: The Right View of Binary Tree: 5, 7, 6, 15

Approach 1: Function to print Right View of a binary tree using a recursive approach.

- The recursive approach involves starting from the root and traversing the tree while maintaining the current level.
- It keeps track of the rightmost node encountered at each level, adding it to the result vector.
- The traversal begins from the root and proceeds first to the right child and then to the left child.
- The result vector contains the right view nodes.

Time Complexity:

- **The time complexity is O(N), where N is the number of nodes in the tree, as each node is visited once.**

Space Complexity:

- **The space complexity is O(H), where H is the height of the binary tree, due to the function call stack.**

Approach 2: Function to print the Right View of a binary tree using an iterative approach

- The iterative approach uses level-order traversal (BFS) to traverse the tree.
- It maintains a queue with a pair of nodes and their levels.

- While traversing, it checks if the size of the result vector is equal to the current level. If so, it adds the rightmost node at that level to the result vector.
- This ensures that the right view nodes are added in left-to-right order.
- Finally, it returns the result vector.

Time Complexity:

- **The time complexity is O(N), where N is the number of nodes in the tree, as each node is visited once during level-order traversal.**

Space Complexity:

- **The space complexity is O(N), where N is the number of nodes in the tree, due to the queue used for level-order traversal.**

Conclusion:

Both the recursive and iterative approaches effectively print the right view of a binary tree. The recursive approach uses a depth-first traversal, while the iterative approach employs a breadth-first traversal.

The time complexity for both approaches is O(N), where N is the number of nodes in the tree. The recursive approach has a space complexity of O(H), while the iterative approach has a space complexity of O(N).

The choice between the two approaches depends on the specific requirements and constraints of the problem.