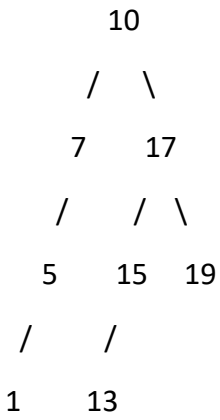
Find Minimum and Maximum Element in BST GFG

Given a Binary Tree, find maximum and minimum elements in it.

Example:



Output: The Minimum Element: 1(Leftmost) and Maximum Element: 19 (Rightmost)

Minimum Element:

Approach 1: Function to find the minimum value in the BST using a recursive approach

- A recursive function **minValueRecursive** is used to find the minimum value in the BST.
- It recursively explores the left subtree until it reaches the leftmost node, which contains the minimum value.
- **Time Complexity: O(H), where H is the height of the tree.**
- **Space Complexity: O(H) due to the function call stack.**

Approach 2: Function to find the minimum value in the BST using an iterative approach

- An iterative function **minValueIterative** is used to find the minimum value in the BST.
- It iteratively traverses the leftmost path of the BST to reach the node with the minimum value.
- **Time Complexity: O(H), where H is the height of the tree.**
- **Space Complexity: O(1) as it doesn't use additional space.**

Maximum Element:

Approach 1: Recursive function to find the maximum value in the BST

- A recursive function **maxValueRecursive** is used to find the maximum value in the BST.
- It recursively explores the right subtree until it reaches the rightmost node, which contains the maximum value.
- **Time Complexity: O(H), where H is the height of the tree.**
- **Space Complexity: O(H) due to the function call stack.**

Approach 2: Iterative function to find the maximum value in the BST

- An iterative function **maxValueIterative** is used to find the maximum value in the BST.
- It iteratively traverses the rightmost path of the BST to reach the node with the maximum value.
- **Time Complexity: O(H), where H is the height of the tree.**
- **Space Complexity: O(1) as it doesn't use additional space.**

Conclusion:

The iterative approaches for finding minimum and maximum values have a space complexity of O(1), making them more memory-efficient.