LCM of Two Numbers

The code aims to find the Least Common Multiple (LCM) of two given numbers, 'a' and 'b', using two different approaches. The first approach is the Brute Force Approach, which iteratively finds the LCM by checking potential multiples. The second approach utilizes the Euclidean Algorithm for finding the GCD to calculate the LCM.

Approach 1: Function to find LCM using the Brute Force Approach

- This function takes two integers, 'a' and 'b', as input.
- It initializes a variable 'lcm' to the maximum of 'a' and 'b'.
- The function then uses a while loop to iteratively increment 'lcm' until it finds the smallest common multiple of 'a' and 'b'.
- Inside the loop, it checks if 'lcm' is a common multiple for both 'a' and 'b' by using the modulo operator.
- When a common multiple is found, the loop breaks, and the LCM is returned.
- **Time Complexity: The time complexity of this approach is O(a * b) in the worst case. It can take up to 'a * b' iterations to find the LCM.**
- **Space Complexity: The space complexity is O(1) since the function uses only a constant amount of additional memory.**

Approach 2: Function to find LCM using GCD with the help of Euclid Algorithm

- This function takes two integers, 'a' and 'b', as input.
- It finds the GCD of 'a' and 'b' using the **findGCDUsingEuclidAlgorithm** function (Euclidean Algorithm).
- It then calculates the LCM using the formula: LCM(a, b) = (a * b) / GCD(a, b).
- The LCM is returned as the result.
- **Time Complexity: The time complexity of the Euclidean Algorithm is O(log(min(a, b))). It performs efficiently even for large numbers.**
- **Space Complexity: The space complexity is O(log(min(a, b))) due to the recursion, which can go up to the depth of 'log(min(a, b))'.**