Modular Exponential CodeStudio

Given three integers 'x', 'n', and 'm', where 'x' is the base, 'n' is the exponent, and 'm' is the modulo, find (x^n) % m.

Approach 1: Brute Force (Naive) Approach

1. The **powerModBruteForce** function calculates (x^n) % m using a brute force approach. It uses a loop to iteratively multiply 'x' with the result and takes the modulo 'm' at each step. The loop runs 'n' times, effectively computing (x^n) % m.

Time Complexity: O(n)

- **The loop runs 'n' times, where 'n' is the exponent, performing constant time operations in each iteration.**

Space Complexity: O(1)

- **The function uses a constant amount of space, regardless of the input values.**

Approach 2: Exponentiation by Squaring Approach

2. The **powerModExponentiationBySquaring** function calculates (x^n) % m using an optimized approach called "Exponentiation by Squaring." It uses bitwise operations to efficiently calculate the power.

Time Complexity: O(log n)

- **The number of iterations in the while loop is approximately log2(n), where 'n' is the exponent. This makes the approach significantly faster for large exponents compared to the brute force approach.**

Space Complexity: O(1)

- **The function uses a constant amount of space, regardless of the input values.**