W8 - Data Structure & Algorithms

Problem 1: Calculate (X^N) % 1,000,000,007

Overview: Efficiently compute the power of a number X raised to an exponent N, modulo 1,000,000,007. This is a common problem in computational mathematics, especially useful in fields like cryptography and number theory.

Algorithm Steps:

1. **Modular Exponentiation**:
    - Use a fast exponentiation algorithm while keeping the results within the modulo 1,000,000,007.
    - This process reduces the complexity from O(N) to O(log N).
2. **Algorithm Explanation**:
    - If N is even, recursively calculate (X^(N/2))^2.
    - If N is odd, return X * (X^(N-1)) % 1,000,000,007.
    - Utilize the property (a*b) % c = ((a % c) * (b % c)) % c.
3. **Handling Large N**:
    - Since N can be as large as 10^9, the algorithm avoids direct multiplication of large numbers which might cause overflow.
4. **Code**:

```
public class PowerModulo {

   private static final long MOD = 1000000007;

   // Function to calculate (x^n) % MOD
   public static long power(long x, long n) {
       // Base case
       if (n == 0) return 1;

       // If n is even
       long halfPower = power(x, n / 2);
       if (n % 2 == 0) {
           return (halfPower * halfPower) % MOD;
       } else {
           // If n is odd
           return (x * (halfPower * halfPower % MOD)) % MOD;
       }
   }

   // Main method to test the function
   public static void main(String[] args) {
       long x = 2; // Base
       long n = 10; // Exponent
       System.out.println("Power of " + x + " raised to " + n + " modulo " + MOD + " is: " + power(x, n));
   }
}
```

Problem 2, 3, 4

↳ 3 cells hidden