week 7 summary:

**Lecture Summary: Recursion**

**Main Topic:**

This lecture focuses on **recursion**, a fundamental concept in computer science used to solve problems by breaking them down into smaller, more manageable sub-problems.

**Key Concepts:**

1. **Recursive Function:**

A function that calls itself with a smaller subset of the problem until a base case is reached.

Example: Computing factorials:

2. **Recursive Procedure:**

Solves problems like listing files in a directory by applying the same procedure to smaller subdirectories.

3. **Recursive Data Structure:**

Structures like strings or trees, where each part resembles the whole.

Example: A string "abcd" can be seen as 'a' + "bcd".

**Examples Covered:**

1. **Factorial Calculation:**

Illustrated through both recursive and iterative approaches. Recursive solutions are elegant, while iterative ones are often more efficient.

2. **String Reversal:**

Recursively reverse a string by combining the reversal of the substring with the first character.

3. **Fibonacci Sequence:**

The Fibonacci series was introduced using recursion and iterative approaches, highlighting performance issues (e.g., exponential growth of recursive calls) and solutions like **memoization**.

4. **Power Calculation:**

Demonstrated a more efficient logarithmic recursive solution for calculating powers by leveraging properties of exponents.

5. **Printing Numbers in Reverse:**

Used recursion to first read inputs and then print them in reverse.

6. **Fractal Drawing:**

Discussed the recursive generation of fractal “H” figures, emphasizing the importance of a base case to stop infinite recursion.

7. **Permutations:**

Illustrated generating all permutations of a string using recursion, showing how each step reduces the problem size.

**Performance Optimization:**

• Recursive solutions, though elegant, can be inefficient due to repeated calculations.

• Techniques like **memoization** and converting recursion to iteration (via compilers) improve performance.

**Applications Highlighted:**

• Mathematical computations

• Data structure traversal

• Drawing fractals

• Generating permutations

This lecture establishes recursion as a versatile and powerful tool for problem-solving in computer science.