

Day 11 Study Notes

1. Aptitude: HCF & LCM

Goal: Understand the relationship between Highest Common Factor (HCF) and Least Common Multiple (LCM) to simplify calculations.

Definitions

- **HCF (GCD):** The largest number that divides two or more numbers without a remainder.
- **LCM:** The smallest non-zero number that is a multiple of two or more numbers.

Key Property (The "Product Formula")

For any two numbers a and b :

$$HCF(a, b) \times LCM(a, b) = a \times b$$

- **Application:** If you know two numbers and their HCF, you can instantly find the LCM without recalculating prime factors.
- **Example:**
 - Numbers: 12 and 15
 - $HCF(12, 15) = 3$
 - $LCM(12, 15) = \frac{12 \times 15}{3} = \frac{180}{3} = 60$

2. Programming: Factorial Using Recursion

Goal: Implement $n!$ using a function that calls itself.

Logic

The factorial of a number n ($n!$) can be defined in terms of $(n - 1)!$:

$$n! = n \times (n - 1)!$$

- **Base Case:** $0! = 1$ and $1! = 1$. This stops the recursion.
- **Recursive Step:** Multiply n by the result of the function called with $n - 1$.

Code Snippet (Python)

```
def factorial(n):
    # Base Case: Stop condition
    if n == 0 or n == 1:
        return 1
```

```
# Recursive Step: Function calls itself
return n * factorial(n - 1)

print(factorial(5)) # Output: 120
```

3. Concept: Python Recursion

Goal: Understand how to structure recursive functions safely.

Key Components

1. **Base Condition:** Crucial to prevent infinite loops. In Python, if a function calls itself indefinitely, it raises a `RecursionError: maximum recursion depth exceeded`.
2. **Recursive Call:** The logic where the problem is broken down into a smaller instance of the same problem.

Internal Handling

Python maintains a limit on recursion depth (usually 1000) to protect the system memory. You can check or change this using `sys.getrecursionlimit()` and `sys.setrecursionlimit()`.

4. C/C++ Concept: Recursion (Stack Behavior)

Goal: Understand the low-level memory implications of recursion.

The Call Stack

Unlike Python's high-level abstraction, understanding recursion in C/C++ requires visualizing the **Stack Memory**:

1. **Stack Frame:** Every time a function is called, a new "frame" is pushed onto the stack. This frame contains:
 - Local variables.
 - Parameters.
 - Return address (where to go back after the function finishes).
2. **LIFO:** The stack operates Last-In, First-Out. The last function called is the first one to resolve.
3. **Stack Overflow:** If the recursion is too deep (e.g., missing base case), the stack memory fills up, causing a program crash (Segmentation Fault or Stack Overflow).

5. SQL: BETWEEN Operator

Goal: Select values within a given range efficiently.

Characteristics

- **Inclusive:** The `BETWEEN` operator includes **both** the start and end values.

- **Data Types:** Works with numbers, text (alphabetical order), and dates.

Syntax

```
SELECT column_names  
FROM table_name  
WHERE column_name BETWEEN value1 AND value2;
```

Examples

1. **Numeric Range:** Find products with a price between 100 and 500 (inclusive).

```
SELECT * FROM Products  
WHERE Price BETWEEN 100 AND 500;
```

2. **Date Range:** Find orders placed in January 2023.

```
SELECT * FROM Orders  
WHERE OrderDate BETWEEN '2023-01-01' AND '2023-01-31';
```

3. **NOT BETWEEN:** Find items outside a specific range.

```
SELECT * FROM Employees  
WHERE Salary NOT BETWEEN 30000 AND 50000;
```