📚 Day 12 Study Notes

1. Aptitude: Logarithms

Goal: Simplify complex calculations involving exponents using logarithmic rules.

Definition

A logarithm is the inverse operation to exponentiation. It answers the question: "To what power must a base b be raised, to produce a given number x?"

$$\log_b(x) = y \iff b^y = x$$

Key Laws of Logarithms

1. **Product Rule:** $\log_b(mn) = \log_b(m) + \log_b(n)$
2. **Quotient Rule:** $\log_b(\frac{m}{n}) = \log_b(m) - \log_b(n)$
3. **Power Rule:** $\log_b(m^n) = n \cdot \log_b(m)$
4. **Base Change Formula:** $\log_a(b) = \frac{\log_c(b)}{\log_c(a)}$
5. **Identity:** $\log_b(b) = 1$ and $\log_b(1) = 0$

Example Problem

Simplify: $\log(50) + \log(20)$ **Solution:** Using the Product Rule:

$$
\log(50 \times 20) = \log(1000)
$$

$$\log_{10}(10^3) = 3 \cdot \log_{10}(10) = 3 \times 1 = 3$$

2. Programming: Anagram Check

Goal: Determine if two strings contain the exact same characters with the same frequencies.
Example: "listen" and "silent" are anagrams.

Logic Approaches

1. **Sorting Method ($O(N \log N)$):** Sort both strings alphabetically. If `sorted(str1) == sorted(str2)`, they are anagrams.
2. **Frequency Counter Method ($O(N)$):** Count the occurrences of each character in both strings. If the counts match for every character, they are anagrams.

3. Concept: Python File Handling

Goal: Manage persistent data storage using `open()`, `read()`, and `write()`.

Key Operations

- `open(filename, mode)`: Returns a file object.
  - `'r'`: Read (default). Error if file doesn't exist.
  - `'w'`: Write. Creates new file or overwrites existing one.
  - `'a'`: Append. Adds data to the end of the file.
- `close()`: Essential to free up system resources.

The `with` Statement (Context Manager)

Best practice is to use `with`. It automatically closes the file even if an error occurs.

```
# Writing to a file
with open("data.txt", "w") as file:
    file.write("Hello, World!")

# Reading from a file
with open("data.txt", "r") as file:
    content = file.read()
    print(content)
```

4. C/C++ Concept: File I/O

Goal: Understand how low-level languages interact with the file system.

C Style (`stdio.h`)

Uses a `FILE` pointer.

- `fopen()`: Opens file.
- `fprintf()`: Writes formatted output to file.
- `fscanf()`: Reads formatted input.
- `fclose()`: Closes the stream.

C++ Style (`fstream`)

Uses stream objects, similar to `cin` and `cout`.

- `ofstream`: Stream class to write on files.
- `ifstream`: Stream class to read from files.

```
#include <fstream>
#include <iostream>
using namespace std;

int main() {
    // Writing
```

```
    ofstream MyFile("filename.txt");
    MyFile << "Files can be tricky, but useful!";
    MyFile.close();
    return 0;
}
```

5. SQL: IN Operator

Goal: specific multiple values in a `WHERE` clause cleanly.

Usage

The `IN` operator is a shorthand for multiple `OR` conditions. $$ \text{column} = \text{val1} \lor \text{column} = \text{val2} \lor \dots $$

Syntax

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (value1, value2, ...);
```

Examples

1. Static List: Find employees located in 'NY', 'CA', or 'TX'.

```
SELECT * FROM Employees
WHERE State IN ('NY', 'CA', 'TX');
```

2. Subquery (Dynamic List): Find customers who have placed an order.

```
SELECT * FROM Customers
WHERE CustomerID IN (SELECT CustomerID FROM Orders);
```