Day 24: Comprehensive Study Notes

1. Aptitude – Data Sufficiency

Data Sufficiency (DS) tests your ability to identify the *minimum* information required to solve a problem, rather than finding the numerical answer itself.

The 5-Option Framework

Most DS questions follow this standard evaluation logic:

1. **Statement (1) ALONE** is sufficient, but (2) alone is not.
2. **Statement (2) ALONE** is sufficient, but (1) alone is not.
3. **BOTH statements TOGETHER** are sufficient, but neither alone is.
4. **EACH statement ALONE** is sufficient.
5. **Statements (1) and (2) TOGETHER** are NOT sufficient.

Strategy Tip: "The Don't Solve" Rule

- Once you realize a unique value or a definite "Yes/No" can be determined, **stop calculating**.
- Beware of "C" traps: Sometimes Statement 1 looks insufficient, but once combined with Statement 2, they provide the missing variable needed for a linear equation.

2. Programming – Longest Common Prefix (LCP)

The goal is to find the longest string that is a prefix of all strings in an array.

Logic: Vertical Scanning

Compare characters at the same index across all strings.

- **Time Complexity:** $O(S)$, where S is the sum of all characters in all strings.
- **Space Complexity:** $O(1)$.

Python Implementation

```
def longestCommonPrefix(strs):
    if not strs:
        return ""

    # Take the first string as a reference
    prefix = strs[0]

    for i in range(len(prefix)):
        char = prefix[i]
        for j in range(1, len(strs)):
            # If index out of bounds or characters don't match
```

```
                if i == len(strs[j]) or strs[j][i] != char:
                    return prefix[:i]

        return prefix
```

3. Concept – Python Shallow Copy vs. Deep Copy

Understanding how Python handles objects in memory is crucial for avoiding bugs in mutable data structures.

Feature	Shallow Copy (`copy.copy()`)	Deep Copy (`copy.deepcopy()`)
Definition	Creates a new object but inserts *references* to the items found in the original.	Creates a new object and recursively adds *copies* of the items found in the original.
Nested Objects	Changes to nested objects in the copy **reflect** in the original.	Changes to nested objects in the copy **do not affect** the original.
Performance	Faster, consumes less memory.	Slower, consumes more memory.

Code Example

```
import copy

original = [[1, 2, 3], [4, 5, 6]]
shallow = copy.copy(original)
deep = copy.deepcopy(original)

original[0][0] = 'X'

print(shallow[0][0]) # Output: 'X' (Reference shared)
print(deep[0][0])    # Output: 1 (Independent copy)
```

4. C/C++ Concept – STL Basics

The Standard Template Library (STL) is a powerful set of C++ template classes.

Core Components

1. **Containers:** Objects that store data.
   - *Sequence:* `vector` , `list` , `deque` .
   - *Associative:* `set` , `map` (sorted by key).
   - *Unordered Associative:* `unordered_map` , `unordered_set` (hash tables).
2. **Algorithms:** Functions to process containers (e.g., `sort()` , `binary_search()` , `reverse()`).

3. **Iterators:** Objects that point to elements within containers, acting as a bridge between algorithms and containers.

5. SQL – CASE Statement

The `CASE` statement is SQL's way of handling `if-then-else` logic within a query.

Syntax

```
SELECT
    name,
    score,
    CASE
        WHEN score >= 90 THEN 'A'
        WHEN score >= 80 THEN 'B'
        WHEN score >= 70 THEN 'C'
        ELSE 'F'
    END AS grade
FROM students;
```

Key Use Cases

- **Categorization:** Grouping numerical data into buckets (e.g., Age 0-18 = 'Minor').
- **Pivoting:** Using `SUM(CASE WHEN ... THEN 1 ELSE 0 END)` to count specific occurrences in a single row.
- **Dynamic Updates:** Changing values in an `UPDATE` statement based on conditions.