**Experiment No. 1**

**Title:** Python Program to Manage Borrowing Records in a Library

**Objective:**

* To develop a program that handles borrowing data for a library.
* To practice use of Python lists/dictionaries and functions.
* To analyze and compute average, maximum, minimum, count, and mode from data.
* To understand time and space complexity of implemented operations.

**Problem Statement:**

Write a Python program to manage the borrowing records of books in a library. Implement the following functionalities:

• Compute the average number of books borrowed by all library members.

• Find the book with the highest and lowest number of borrowings in the library.

• Count the number of members who have not borrowed any books (denoted by a borrow count of 0).

• Display the most frequently borrowed book (i.e., the mode of borrow counts).

After performing, determine the time and Space complexity of each operation

**Outcomes:**

* Students will be able to handle real-world data problems using Python.
* Students will learn how to analyze frequency and statistical operations on data.
* Students will be able to apply complexity analysis.

**Software and Hardware Requirements:**

* **Software:** Python 3.10+, Jupyter Notebook or any IDE (VS Code / PyCharm)
* **Hardware:** Minimum 2 GB RAM, 1 GHz processor, Any OS (Linux)

**Theory:**

The problem involves data representation and statistical analysis using programming. Lists or dictionaries can store borrow data. Various aggregate functions such as average, min, max, mode, and count are used to evaluate the dataset. Time and space complexity are evaluated based on the nature of operations like iteration, lookup, and insertion.

***Mathematical Model:***

***Let***

***M = {set of all members}***

***B = {set of books borrowed}***

***f(m) = number of books borrowed by member m***

***Objective:***

***Find avg(f(m))***

***max(f(m)), min(f(m))***

***count (m ∈ M | f(m) = 0)***

***mode(f(m))***

**Algorithm:-**

**Input:**

* book\_borrow\_count: Dictionary mapping book titles to total borrow counts.
* member\_borrow\_count: Dictionary mapping member names to total borrow counts.

**Step 1: Compute Average Number of Books Borrowed by Members**

1. Initialize total\_borrowed = 0

2. For each member in member\_borrow\_count:

total\_borrowed += borrow count

3. total\_members = number of members

4. If total\_members > 0:

average = total\_borrowed / total\_members

Else:

average = 0

5. Output average

⏱ **Time Complexity:** O(n), where n = number of members  
💾 **Space Complexity:** O(1)

**Step 2: Find Book with Highest and Lowest Borrowings**

1. If book\_borrow\_count is empty:

Return None for both

2. Initialize max\_borrow = -∞, min\_borrow = ∞

3. For each book in book\_borrow\_count:

If borrow count > max\_borrow:

max\_borrow = borrow count, max\_book = book

If borrow count < min\_borrow:

min\_borrow = borrow count, min\_book = book

4. Output max\_book and min\_book

⏱ **Time Complexity:** O(m), where m = number of books  
💾 **Space Complexity:** O(1)

**Step 3: Count Members with Zero Borrowings**

1. Initialize count = 0

2. For each member in member\_borrow\_count:

If borrow count == 0:

count += 1

3. Output count

⏱ **Time Complexity:** O(n), where n = number of members  
💾 **Space Complexity:** O(1)

**Step 4: Find the Most Frequently Borrowed Book (Mode)**

1. If book\_borrow\_count is empty:

Return None

2. Initialize frequency\_map = {}

3. For each book in book\_borrow\_count:

borrow\_count = book\_borrow\_count[book]

frequency\_map[borrow\_count] += 1

4. Find the borrow\_count with max frequency in frequency\_map → mode\_count

5. For each book in book\_borrow\_count:

If book\_borrow\_count[book] == mode\_count:

Add to mode\_books list

6. Output mode\_books

⏱ **Time Complexity:** O(m)  
💾 **Space Complexity:** O(k), where k = number of unique borrow counts (typically k << m)

**✅ Summary Table**

| **Operation** | **Time Complexity** | **Space Complexity** |
| --- | --- | --- |
| Average books borrowed by members | O(n) | O(1) |
| Book with highest/lowest borrowings | O(m) | O(1) |
| Count members with zero borrowings | O(n) | O(1) |
| Most frequently borrowed book (mode) | O(m) | O(k) |

**Test Cases:**

| **TestCase** | **Borrow Data** | **Expected Output** |
| --- | --- | --- |
| 1 | {"Alice": 2, "Bob": 0, "Charlie": 5} | Avg: 2.33, Max: 5, Min: 0, Zero Count: 1, Mode: 0/2/5 (any with freq=1) |
| 2 | {"A": 1, "B": 1, "C": 0, "D": 1} | Avg: 0.75, Max: 1, Min: 0, Zero Count: 1, Mode: 1 |
| 3 | {} | Handle empty input gracefully (no division by zero or errors) |

**Conclusion:**

In this assignment, we created a Python program to manage library borrowing records efficiently. The program calculated the average books borrowed by members, identified the most and least borrowed books, counted members with zero borrowings, and determined the most frequently borrowed book. Using Python dictionaries and built-in functions, we achieved clear, concise, and optimized solutions. We also analyzed the time and space complexities of each operation to ensure scalability.

**Questions:**

1. What data structures are used in this program and why?

2. What is the difference between max() and manually finding the maximum value?

3. Why do we use Counter from the collections module for finding the mode?