International Islamic University Islamabad



Submitted By:-

0mer jabbar( 4768-FOC-BSSE-F23A)

mesum Abbas( 4736-FOC-BSSE-F23A)

syed bilal ali( 4748-FOC-BSSE-F23A)

Ahmed bilal( 4598-FOC-BSSE-F23A)

Submitted T0:-

Dr.prof Shakeel Ahmed

Data structure&algorithm

🔍 Sort and Search Algorithm Analysis

This system uses several searching and sorting techniques to manage gym members and locker assignments efficiently. Below is a structured explanation of how these algorithms are implemented in the provided code, including their usage, complexity, and justification.

---

🔸 1. Sorting of Members (Using std::sort() with Lambda)

In the program, member details are stored in a vector named members. Before displaying the list of all members, the program uses the standard C++ library function std::sort() along with a lambda expression to sort the vector based on the registration number of each member. The syntax used is:

sort(members.begin(), members.end(), [](const Member& a, const Member& b) {

return a.regNo < b.regNo;

});

This sorting ensures that the members are displayed in an ordered manner. The time complexity of std::sort() is O(n log n), which is optimal for sorting in most practical scenarios. The lambda function makes the sorting flexible by allowing custom comparison logic.

🔸 2. Searching Members (Using unordered\_map Lookup)

To allow fast lookup of members by their registration number, the program uses an unordered\_map<int, Member> named memberMap. When a user wants to search for a specific member, the code checks if the registration number exists in the map using:

auto it = memberMap.find(regNo);

This search operation has an average-case time complexity of O(1), which means it can retrieve a member's information almost instantly. Using a hash map (unordered\_map) is a very efficient choice here, especially when dealing with frequent searches.

---

🔸 3. Locker Assignment (Using Linear Search)

Lockers are managed using a boolean array isLockerTaken[101], where each index represents a locker number. When assigning a locker to a new member, the system performs a linear search from locker 1 to 100 to find the first free locker:

for (int i = 1; i <= 100; i++) {

if (!isLockerTaken[i]) {

locker = i;

isLockerTaken[i] = true;

break;

}

}

This algorithm has a worst-case time complexity of O(n), where n is the total number of lockers (100 in this case). Although linear search is not the most efficient method, it is acceptable here due to the small fixed size of the locker range.

🔸4. Locker Number Tracking (Using Counter)

A simple counter variable nextLockerNumber is used to keep track of the next available locker. This method works in constant time (O(1)) and provides quick assignment, though it still relies on checking availability through the array. This is a minimal, fast approach that suits the simplicity of the application.

🔸 5. Recommendations and Suitability

Overall, the combination of unordered\_map for search and std::sort() with lambda for sorting makes the system efficient for real-time access and organized display. The linear search for locker assignment could be optimized further by using a set<int> to store free lockers, allowing faster retrieval of the smallest available locker in O(log n) time. However, for a fixed locker range of 100, the current solution remains effective.