Data Structures and Algorithms –

BOARD INFINITY

A Training Report

Bachelor of Technology

(Computer Science and Engineering)

Submitted to

LOVELY PROFESSIONAL UNIVERSITY

PHAGWARA, PUNJAB

From 28th May 2024 to 25th July 2024

SUBMITTED BY

Name : Deepshikha Pal

Registration Number: 12217173

To whom so ever it may concern

I, **Deepshikha Pal**, **12217173**, hereby declare that the work done by me on **"Data Structures and Algorithms"** from **28h May 2024 to 25th July 2024**, is a record of original work for the partial fulfilment of the requirements for the award of the degree, **Bachelor of Technology**.

Name of the Student: Deepshikha Pal

Registration Number: 12217173

Dated: 24th August 2024

SUMMER TRAINING CERTIFICATE

ACKNOWLEDGEMENT

First and foremost, I am deeply grateful for the opportunity to expand my knowledge and skills in a new technology. I would like to extend my sincere thanks to the instructor of the DSA Self-Paced course from Board Infinity, whose guidance has been invaluable in facilitating my learning journey from home.

I am also thankful to my college, Lovely Professional University, for providing access to such an enriching course that not only enhanced my programming abilities but also introduced me to new technologies.

I would like to express my heartfelt appreciation to my parents and friends for their unwavering support, valuable advice, and encouragement in choosing this course. Lastly, my gratitude extends to my classmates, whose collaboration and assistance have been greatly appreciated.

LIST OF CONTENTS

LIST OF ABBREVIATION

DSA – Data Structures and Algorithms

INTRODUCTION

The DSA course, short for Data Structures and Algorithms, is designed to be self-paced, allowing participants to enroll at any time. All course materials are accessible immediately upon enrollment, and students can complete the course at their own pace.

➢ **What is a Data Structure?** A data structure is a method of organizing and managing data so that operations can be performed efficiently. It involves arranging data elements based on specific relationships to improve organization and storage. For instance, if we have data with a player's name "Virat" and age 26, "Virat" is a String type, while 26 is an Integer type.

➢ **What is an Algorithm?** An algorithm is a set of step-by-step instructions or logic, arranged in a specific order, to achieve a particular task. It is the fundamental solution to a problem, presented either as a high-level description in pseudocode or through a flowchart, but it is not the full code or program.

➢ This course provided a comprehensive learning experience, guiding me through Data Structures and Algorithms from the basics to advanced concepts. The curriculum is structured into 8 weeks, allowing participants to work through practice problems and assessments at their own pace. The course offers a variety of programming challenges that are invaluable in preparing for interviews with leading companies like Microsoft, Amazon, Adobe, and others.

TECHNOLOGY LEARNT

Analysis of Algorithms

- **Analysis of Algorithm:** This section focused on understanding background analysis through programs and their functions.
- **Order of Growth:** I learned about the mathematical representation of growth analysis using limits and functions, including a straightforward approach to determining the order of growth.
- **Asymptotic Notations:** This covered the best, average, and worst-case scenarios, explained through example programs.
- **Big O Notation:** The concept was explained both graphically and mathematically, with calculations and applications demonstrated using Linear Search.
- **Omega Notation:** I explored this notation through graphical and mathematical explanations, along with calculations.
- **Theta Notation:** The Theta notation was explained with graphical and mathematical insights, including detailed calculations.
- **Loop Analysis:** I studied the analysis of single, multiple, and nested loops.
- **Recursion Analysis:** This included various calculations using the Recursion Tree method.

- **Space Complexity:** I explored basic programs, auxiliary space, and analyzed the space complexity of recursive functions and the Fibonacci sequence.

Mathematics

- **Digit Counting:** Techniques for finding the number of digits in a number.
- **Progressions:** Arithmetic and geometric progressions were covered.
- **Quadratic Equations:** Solving quadratic equations.
- **Statistical Measures:** Concepts of mean and median.
- **Prime Numbers:** Understanding prime numbers.
- **LCM and HCF:** Learning about Least Common Multiple (LCM) and Highest Common Factor (HCF).
- **Factorials:** Calculating factorials.
- **Permutations and Combinations:** Exploring permutations and combinations.
- **Modular Arithmetic:** Learning the basics of modular arithmetic.

Bitwise Operations

- **Bitwise Operators in C++:** I explored the operations of AND, OR, XOR, Left Shift, Right Shift, and Bitwise NOT in C++.

- **Bitwise Operators in Java:** The operations of AND, OR, Bitwise NOT, Left Shift, Right Shift, and unsigned Right Shift in Java were studied.
- **Problem Solving:** I tackled problems like checking if the Kth bit is set, using methods such as Left Shift and Right Shift.

Recursion

- **Introduction:** An introduction to recursion, including its applications.
- **Writing Base Cases:** Learning to write base cases in recursion through examples like factorial and N-th Fibonacci number.

Arrays

- **Introduction and Benefits:** An overview of arrays and their advantages.
- **Types of Arrays:** Covered fixed-sized and dynamic-sized arrays.
- **Array Operations:** Studied operations like searching, insertion, deletion, comparison with other data structures, and reversing arrays with complexity analysis.

Searching

- **Binary Search:** I learned both iterative and recursive approaches to binary search, along with associated problems.
- **Two Pointer Approach:** Explored problems that utilize the two-pointer approach.

Sorting

- **STL Sort in C++:** Implementing the **sort()** function in arrays and vectors in C++ with a focus on time complexities.
- **Sorting in Java:** Understanding **Arrays.sort()** and **Collection.sort()** in Java.
- **Stability in Sorting Algorithms:** Analyzed stable and unstable sorting algorithms with examples.
- **Sorting Algorithms:** I implemented and analyzed sorting algorithms like Insertion Sort, Merge Sort, and Quick Sort (including Lomuto and Hoare partitioning methods), along with their time and space complexities, pivot selection, and worst-case scenarios.
- **Sorting Algorithms Overview:** A broad overview of sorting algorithms.

Matrices

- **Introduction to Matrices:** Basics of working with matrices in C++ and Java.

- **Multidimensional Matrices:** Handling multidimensional matrices.
- **Matrix Operations:** Operations like passing a matrix as an argument, printing in a snake pattern, transposing, rotating, boundary traversal, spiral traversal, matrix multiplication, and searching in row-wise and column-wise sorted matrices were covered.

Hashing

- **Introduction to Hashing:** Time complexity analysis and applications of hashing were introduced.
- **Hash Functions:** I learned about Direct Address Tables and various hashing functions.
- **Collision Handling:** Discussed different collision handling techniques, including chaining and open addressing, with implementations.
- **Hashing in C++ and Java:** Explored **unordered_set** and **unordered_map** in C++, and **HashSet** and **HashMap** in Java.

Strings

- **String Data Structure:** Studied string data structures in C++ and Java.
- **String Algorithms:** Implemented string algorithms like Rabin Karp and KMP.

Linked Lists

- **Introduction:** An introduction to linked lists with implementations in C++ and Java.
- **Types of Linked Lists:** Explored doubly linked lists and circular linked lists.
- **Loop Problems:** Covered problems like detecting loops using Floyd's cycle detection algorithm, and detecting and removing loops in linked lists.

Stacks

- **Understanding Stacks:** An overview of the stack data structure and its applications.
- **Stack Implementation:** Implementation of stacks using arrays and linked lists in C++ and Java.

Queues

- **Introduction to Queues:** Basics and applications of queues.
- **Queue Implementation:** Implemented queues using arrays and linked lists in C++ STL and Java, including stack-based queue implementation.

Deques

- **Introduction to Deques:** Studied deque data structure and its applications.

- **Deque Implementation:** Implemented deques in C++ STL and Java.
- **Problem Solving:** Tackled problems like finding the maximum of all subarrays of size k, using **ArrayDeque** in Java, and designing a data structure with min-max operations.

Trees

- **Introduction to Trees:** Covered the basics of trees, including binary trees and their applications.
- **Tree Traversals:** Implemented various tree traversal techniques like Inorder, Preorder, Postorder, Level Order (line by line), and Spiral Form traversal.

Binary Search Trees (BST)

- **Introduction to BST:** An overview of Binary Search Trees, including background, introduction, and applications.
- **BST Operations:** Implemented search, insertion, deletion, and floor operations in BST, along with exploring self-balancing BSTs like AVL trees.

Heaps

- **Introduction to Heaps:** Basics and implementation of heaps.

- **Binary Heap Operations:** Implemented binary heap operations like insertion, heapify, extract, decrease key, delete, and building a heap.
- **Heap Sort:** Studied heap sort, and priority queues in C++ and Java.

Graphs

- **Introduction to Graphs:** Basics of graph data structures and their representation using adjacency matrix and adjacency list in C++ and Java.
- **Graph Traversal:** Implemented Breadth-First Search (BFS) and Depth-First Search (DFS) along with their applications.
- **Shortest Path Algorithms:** Studied algorithms for finding the shortest path in Directed Acyclic Graphs, Prim's Algorithm for Minimum Spanning Tree, Dijkstra's Shortest Path Algorithm, Bellman-Ford Algorithm, Kosaraju's Algorithm, articulation points, bridges in graphs, and Tarjan’s Algorithm.

Greedy Algorithms

- **Introduction to Greedy Algorithms:** Learned the basic concepts and applications.
- **Greedy Problems:** Implemented and analyzed problems like Activity Selection, Fractional Knapsack, and Job Sequencing.

Backtracking

- **Backtracking Concepts:** Introduction to backtracking with examples like the Rat in a Maze and N-Queen problem.

Dynamic Programming

- **Introduction to Dynamic Programming:** Studied the fundamentals of dynamic programming.
- **Dynamic Programming Techniques:** Learned and implemented memoization and tabulation methods.

Segment Trees

- **Segment Tree Introduction:** Covered the basics of segment trees, including construction, range queries, and update queries.

Disjoint Sets

- **Introduction to Disjoint Sets:** Learned about disjoint set data structures, including find and union operations, union by rank, and path compression, along with implementing Kruskal's Algorithm.

➢ Enhanced my problem-solving abilities by tackling a variety of challenges to become a more proficient developer.

- **Practice Problems:** This track offers numerous essential practice problems that are crucial for mastering Data Structures and Algorithms.

➢ Sharpened my analytical skills in Data Structures, enabling me to utilize them more effectively.

➢ Successfully tackled problems commonly encountered in interviews with product-based companies.

➢ Competed in contests that mirror the coding rounds for Software Development Engineer (SDE) roles.

PROJECT OF DSA

A Project Report –

Simple Library Management System using Data Structures

Session : 2024-2025

Lovely Professional University Jalandhar-Delhi

G.T. Road, Phagwara, Punjab(India)-144411

Submitted by: Deepshikha Pal

Reg No: 12217173

Introduction: In this project, I have developed a basic library management system using C++. The system is designed to cater to two types of users: librarians and students. It provides a command-line interface for easy interaction and management of library resources.

System Overview: The system is built around two main structures: Book and User. These form the core of our data management. I implemented the following key components:

1. Main Menu: I created a central hub for user interaction, allowing users to log in or exit the program.
2. Authentication System: I developed a simple yet effective login mechanism to differentiate between librarian and student access.
3. Librarian Functionalities: For librarians, I implemented a comprehensive set of features including:
    - Displaying all books
    - Adding new books
    - Removing books
    - Searching for books
    - Filtering books by genre
    - Registering new students
4. Student Functionalities: For students, I created a more limited set of features:
    - Viewing available books
    - Searching for books
    - Viewing books by genre
5. Data Management: I utilized C++ vectors and unordered maps for efficient storage and retrieval of book and user data.
6. User Interface: I designed intuitive menu systems for both librarian and student interactions, ensuring clear prompts and feedback.

7. Error Handling: Basic input validation and error messages were implemented to enhance system robustness.
8. Modular Design: I structured the code into separate functions for each major operation, improving readability and maintainability.

Implementation Details: The system is implemented entirely in C++, leveraging standard libraries for data structures and input/output operations. Key programming concepts utilized include:

- Structs for data organization
- Functions for modular code design
- Control structures (loops, switch statements) for program flow
- STL containers for data management

Challenges and Solutions: One challenge was balancing functionality with simplicity. I addressed this by creating distinct menus for librarians and students, ensuring each user type has access to appropriate features.

Future Enhancements: While the current system provides core functionalities, there's room for expansion. Future iterations could include:

- Book borrowing and return system for students
- Persistent data storage using file I/O
- More advanced search and filter options

Conclusion: This library management system demonstrates a practical application of C++ programming concepts. It provides a solid foundation for managing library resources and can be easily extended for more complex requirements.

By implementing this system, I've gained valuable experience in software design, user interface creation, and data management in C++.

- Here is the input and output of the code . The code implements a simple library management system using C++. The system allows librarians to manage books and users, and students to issue and return books. Input and Output

```
-----------------------------------------
| WELCOME TO THE LIBRARY |
-----------------------------------------
Please Choose an appropriate option:
=====================================
0. Exit
1. LogIn as Librarian
2. LogIn as Student
Please Pick any one option: 1
```

2. Librarian Login:

```
-------------------------------------------------
| LogIn as Librarian |
-------------------------------------------------
Username: admin
Password: admin123
Access granted.
```

3. Librarian Menu:

```
++++++++++ M E N U ++++++++++
Welcome Librarian
0. Exit
1. Display All Books
2. Insert a Book
3. Delete a Book
4. Search Book
5. Show Books by Genre
6. Register a Student
7. LogOut
Select an option: 2
```

4. Insert a Book:

```
Book Name: The Hobbit
Author Name: J.R.R. Tolkien
Book Genre: Fantasy
```

```
Book Quantity: 10
Book added successfully.
```

5. Display All Books:

```
Select an option: 1
ID: 1
Book Name: The Hobbit
Author: J.R.R. Tolkien
Genre: Fantasy
Quantity Available: 10
```

6. Search Book:

```
Select an option: 4
Book Name: The Hobbit
[BOOK FOUND]
Book Title: The Hobbit
Book Author: J.R.R. Tolkien
Book Genre: Fantasy
Book Quantity: 10
```

7. Register a Student:

```
Select an option: 6
Student Name: johndoe
Password: pass123
[USER ADDED]
```

8. Librarian Logout:

```
Select an option: 7
Logging out...
```

9. Student Login:

```
-----------------------------------------
| WELCOME TO THE LIBRARY |
-----------------------------------------
Please Choose an appropriate option:
=====================================
0. Exit
1. LogIn as Librarian
2. LogIn as Student
Please Pick any one option: 2
```

```
-------------------------------------------------------
| LogIn as Student |
-------------------------------------------------------
Username: johndoe
Password: pass123
Access granted.
```

10. Student Menu:

Copy

```
++++++++++ M E N U ++++++++++
Welcome Student
0. Exit
1. Display All Books
2. Issue a Book
3. Return a Book
4. Show all Issued Books
5. Search Book
6. Show Books by Genre
7. LogOut
Select an option: 1
```

11. Display All Books (Student view):

Copy

```
ID: 1
Book Name: The Hobbit
Author: J.R.R. Tolkien
Genre: Fantasy
Quantity Available: 10
```

12. Search Book (Student view):

Copy

```
Select an option: 5
Book Name: The Hobbit
[BOOK FOUND]
Book Title: The Hobbit
Book Author: J.R.R. Tolkien
Book Genre: Fantasy
Book Quantity: 10
```

13. Show Books by Genre:

Copy

```
Select an option: 6
Enter Genre: Fantasy
The Hobbit by J.R.R. Tolkien
```

14. Student Logout:

```
Select an option: 7
Logging out...
```

15. Exit Program.

```
-----------------------------------------
| WELCOME TO THE LIBRARY |
-----------------------------------------
Please Choose an appropriate option:
=====================================
0. Exit
1. LogIn as Librarian
2. LogIn as Student
Please Pick any one option: 0
Exiting...
```

REASON FOR CHOOSING DSA

- With rapid advancements in technology, programming has become an essential and highly sought-after skill for Software Developers. Everything from smart devices like TVs and ACs to traffic lights relies on programming to execute user commands.
- **Efficiency is Key:** To remain irreplaceable, one must be efficient. Mastery of Data Structures and Algorithms is a hallmark of a skilled Software Developer. Technical interviews at leading tech companies such as Google, Facebook, Amazon, and Flipkart often focus heavily on a candidate's proficiency in these areas. This focus is due to the significant impact that Data Structures and Algorithms have on enhancing a developer's problem-solving capabilities.
- **Learning Method:** This course offered comprehensive video lectures on all topics, which suited my preferred learning style. While books and notes have their importance, video lectures facilitate faster understanding through practical involvement.
- **Extensive Practice:** The course included over 200 algorithmic coding problems with video explanations, providing a rich resource for hands-on learning.
- **Structured Learning:** The course was organized with track-based learning and weekly assessments to continually test and improve my skills.

- **Productive Use of Time:** During the Covid-19 pandemic, this course was an excellent way for me to invest my time in learning, rather than wasting it on unproductive activities.
- **Lifetime Access:** The course offers lifetime access, allowing me to revisit the material whenever I need to refresh my knowledge, even after completing the initial training.

LEARNING OUTCOMES

Programming fundamentally revolves around the use of data structures and algorithms. Data structures manage and store data, while algorithms solve problems by processing that data.

Data Structures and Algorithms (DSA) explore solutions to common problems in depth, providing insights into the efficiency of different approaches. Understanding DSA empowers you to select the most effective methods for any given task. For example, if you need to search for a specific record in a database of 30,000 entries, you could choose between methods like Linear Search and Binary Search. In this scenario, Binary Search would be the more efficient choice due to its ability to quickly narrow down the possibilities.

Knowing DSA allows you to tackle problems with greater efficiency, making your code more scalable, which is crucial because:

- Time is valuable.
- Memory is costly.

DSA is also essential for excelling in technical interviews at product-based companies. Just as we prefer someone who can complete a task quickly and efficiently in our daily lives, companies look for developers who can solve complex problems effectively and with minimal resources.

For instance, if you're asked to calculate the sum of the first N natural numbers during an interview:

- One candidate might use a loop:

```
int sum = 0;
for (int n = 1; n <= N; n++) {
    sum += n;
}
```

- Another candidate might use the formula: $Sum=N\times(N+1)/2$

The latter approach is more efficient and would likely be favored by the interviewer.

Familiarity with data structures like Hash Maps, Trees, Graphs, and algorithms equips you to solve problems effectively, much like a mechanic uses the right tools to fix a car. Interviewers seek candidates who can select and apply the best tools to address the challenges they are presented with. Understanding the characteristics of different data structures helps you make informed decisions in problem-solving.

Practical Use of DSA: If you enjoy solving real-world problems, DSA is invaluable.

Consider these examples:

- **Organizing Books in a Library:** Suppose you need to find a book on Data Science. You'd first go to the technology section, then look for the Data Science subsection. If the books were not organized in this way, it would be much more challenging to find the specific book. Data structures help organize information in a computer system, enabling efficient processing based on the provided input.

- **Managing a Playlist:** If you have a playlist of your favorite songs, you might arrange them in a particular order, such as by genre or mood. A **Queue** could be used to manage the order in which the songs play, ensuring that they play in the sequence you prefer.
- **Navigating a City:** When finding the shortest route between two locations in a city, you could use a **Graph** data structure to represent the roads and intersections, with algorithms like Dijkstra's or A* to find the optimal path.

These examples illustrate the importance of choosing the right data structure and algorithm for real-world problems, helping to solve them more efficiently.

Data Structures and Algorithms deepen your understanding of problems, allowing you to approach them more effectively and gain a better grasp of the world around you.

BIBLIOGRAPHY

- Board Infinity
- Google