FOSSEE AUTUMN INTERNSHIPS 2025

On

Development of an Animated Educational Series on Data Structures and Algorithms

Submitted by:

Aditi Biswas
B. Tech in Computer Science and Design
Dr. B.C. Roy Engineering College, Durgapur

Under the guidance of

Prof. Prabhu Ramchandran,

Principal Investigator,

Aerospace, CFD Lab, IIT Bombay, Mumbai - 400 076.

Acknowledgement

I would like to express my sincere gratitude to **Prof. Prabhu Ramchandran**, whose vision of the **FOSSEE initiative** has enabled students like me to work on such a noble and creative endeavour, helping in the dissemination of knowledge and making complex concepts easier to understand.

I would also like to thank my mentor **Mr. Khushalsingh K. Rajput (FOCAL Lead and Sr. Software Engineer, FOSSEE, IIT Bombay)**. His insightful guidance, quick-witted suggestions, and creative ideas played a crucial role in shaping this animation project. Without his valuable feedback and encouragement, the animations would not have developed into their current form. His ability to recognize the strengths of students and assign projects aligned with their talents greatly improves the quality of the work and helps students gain deeper knowledge of the subject.

I would also like to take this opportunity to express my gratitude to **Dr. B. C. Roy Engineering College, Durgapur**, for providing me with the academic environment and support that helped me undertake and successfully complete this project.

Finally, I would like to thank everyone who directly or indirectly supported me during the course of this internship. I consider this opportunity an important milestone in my career development. I will strive to utilize the skills and knowledge gained from this experience in the best possible way and continue to improve in order to achieve my future academic and professional goals. I also hope to continue cooperation with all of you in the future.

With Regards,
Aditi Biswas

Abstract

Data Structures and Algorithms are fundamental concepts in computer science that enable efficient data organization, processing, and retrieval. However, these topics are often perceived as complex and abstract by beginners, especially when explained solely through theoretical or textual approaches. Visual learning methods such as animation can significantly improve comprehension by transforming abstract ideas into clear and engaging visual representations.

This project focuses on the creation of a structured **educational animation series consisting of twelve chapters**, each dedicated to explaining a key concept in Data Structures and Algorithms. The animations were designed to simplify complex computational ideas through visual storytelling, step-by-step demonstrations, and minimal narration. Each chapter introduces a concept, demonstrates its functionality visually, and connects it to real-world applications.

The animations were developed using **Synfig Studio**, a vector-based 2D animation tool, allowing the use of clear diagrams, animated objects, and structured transitions. A guiding character named **Algo** accompanies viewers throughout the series, exploring different "data worlds" that represent various data structures and algorithms. This narrative approach makes the learning experience more engaging while maintaining a strong educational focus.

The primary objective of this project was to create **beginner-friendly instructional animations** that can assist students in understanding fundamental data structure concepts such as arrays, stacks, queues, recursion, linked lists, trees, graphs, sorting algorithms, and hashing techniques. By combining visual explanation with structured narration, the series aims to bridge the gap between theoretical computer science concepts and intuitive understanding.

Contents

1. Introduction

Data Structures and Algorithms form the backbone of modern computing systems. Every software application, database system, and digital platform relies on efficient data organization and processing methods to function effectively. Understanding these concepts is essential for students studying computer science, programming, or information technology.

Despite their importance, these topics are often difficult for beginners to grasp due to their abstract nature. Concepts such as trees, graphs, and hashing involve structural relationships and algorithmic processes that may not be immediately clear through static diagrams or textual descriptions.

Educational animations provide a powerful solution to this challenge. By visually representing data structures and demonstrating how algorithms operate step by step, animations help learners build intuitive mental models of complex processes.

This project involves the development of a **twelve-chapter animated learning series** focused on the fundamental concepts of Data Structures and Algorithms. Each animation is designed to present information clearly, gradually building knowledge as learners progress from simple structures to more advanced computational techniques.

A consistent visual style and narrative framework were used throughout the series to maintain learner engagement. The animations emphasize simplicity, clarity, and conceptual understanding rather than excessive technical detail.

2. Objectives of the Project

The primary objectives of this project were:

- To simplify complex Data Structures and Algorithms concepts using visual explanations.
- To design educational animations that are suitable for beginner-level learners.
- To create a structured learning sequence covering key topics in computer science.
- To demonstrate how algorithms operate step by step through animation.
- To develop a reusable animation framework that can be applied to multiple educational topics.

By achieving these objectives, the project aims to provide a valuable learning resource for students beginning their journey in computer science.

3. Tools and Development Methodology

3.1 Animation Software

The animations were created using **Synfig Studio**, an open-source 2D vector animation software. Synfig allows the creation of smooth and scalable animations using vector graphics, making it suitable for educational content that requires diagrams, shapes, and object transformations.

The software supports keyframe animation, layer-based composition, and transformation tools that allow elements such as nodes, arrays, and graphs to move smoothly during demonstrations.

3.2 Visual Design Approach

The visual design of the animations follows a **minimal and educational style**. Instead of complex character animation, the focus is placed on clear diagrams and structured motion.

Key visual elements include:

- Simple geometric shapes representing nodes and data elements
- Highlighting and glow effects to emphasize active components
- Sequential animations to demonstrate algorithm steps
- Consistent color coding to distinguish different elements

This design approach ensures that viewers focus primarily on the concepts being explained.

3.3 Narrative Framework

To make the learning experience more engaging, a narrative element was incorporated into the animation series.

A guiding character named **Algo** explores various “data worlds,” each representing a different concept in Data Structures and Algorithms. As Algo progresses through each chapter, new structures and algorithms are introduced.

At the end of each chapter, a symbolic **door unlocks**, representing progress toward the next concept. This storytelling device helps create continuity across the entire series.

4. Structure of the Animation Series

The animation series consists of **twelve chapters**, each approximately six minutes long. Each chapter follows a consistent structure:

1. Introduction of the concept
2. Visual explanation of the structure or algorithm
3. Demonstration through animated examples
4. Brief discussion of practical applications
5. Transition to the next chapter

This consistent structure helps learners follow the progression of ideas more easily.

5. Chapters Developed

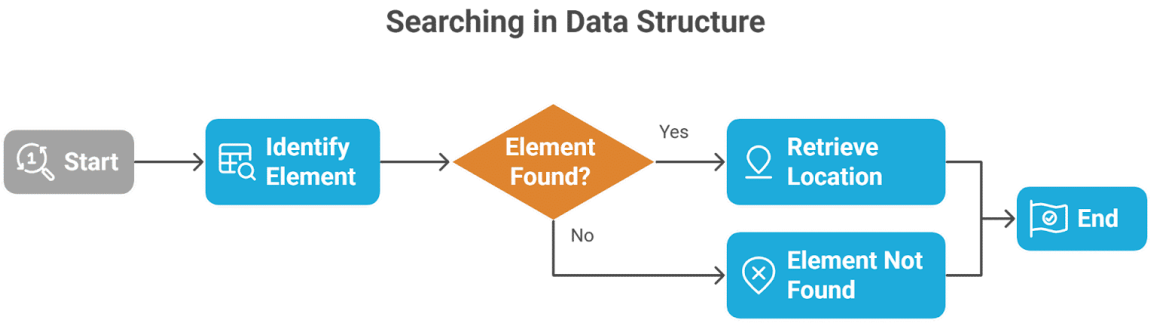
Chapter 1: Introduction to Data Structures and Searching

The first chapter introduces the concept of **data structures** and explains why organizing data is essential in computing systems. In the digital world, massive amounts of data are generated and processed every second. Without proper structures, retrieving specific information would become inefficient and time-consuming.

The chapter begins with the fundamental concept of **searching**, which is the process of locating a particular element within a collection of data. Two basic searching techniques are demonstrated:

- **Linear Search** is presented as the simplest approach, where each element is checked one by one until the target element is found. The animation visually shows the character Algo examining each box in a sequence until the correct value appears.
- **Binary Search**, on the other hand, is introduced as a faster method that works on sorted data. Instead of checking each element sequentially, the search space is repeatedly divided in half. The animation demonstrates this by highlighting the middle element and gradually narrowing down the search range.

This chapter establishes the foundation for understanding how data can be stored and retrieved efficiently.



Chapter 2: Arrays and Sparse Matrices

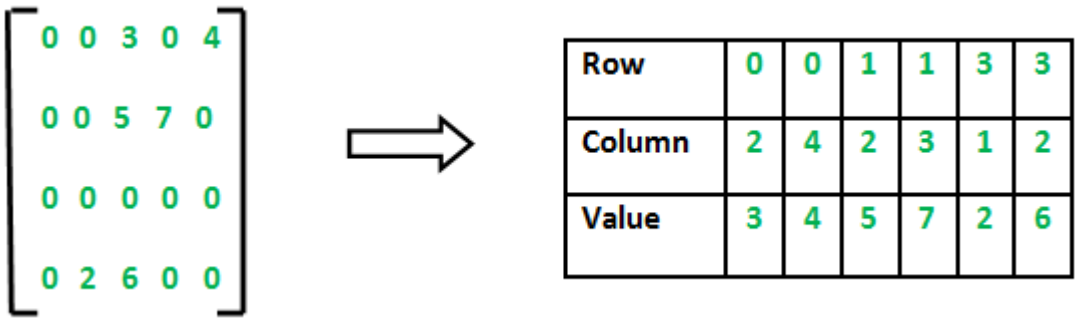
The second chapter focuses on **arrays**, one of the most fundamental and widely used data structures. Arrays store elements in **contiguous memory locations**, allowing fast access to elements using index positions.

The animation represents arrays as a series of boxes arranged in a straight line, similar to lockers in a storage system. Each box holds a value and can be accessed using its index number.

The chapter also introduces **sparse matrices**, which are used when a matrix contains mostly zero values. Storing all elements in such cases would waste memory. Instead, sparse matrices store only the non-zero values along with their row and column positions.

The animation visually shows a large matrix with mostly empty cells and then demonstrates how only the meaningful data is compressed into a smaller, efficient structure.

This chapter highlights the importance of **memory optimization and efficient storage**.



Chapter 3: Stack and Its Applications

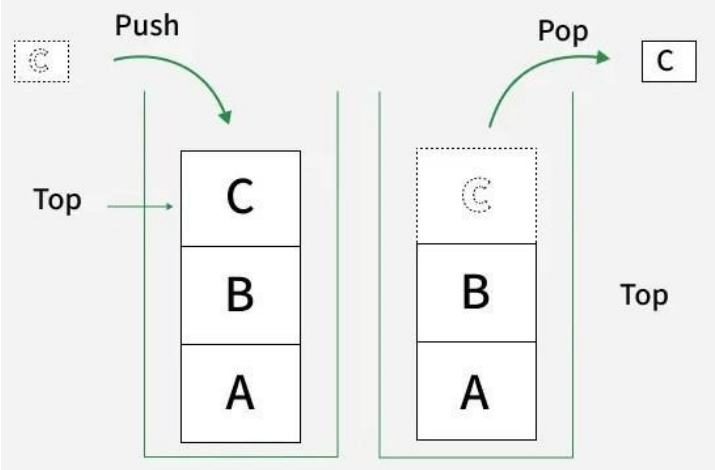
This chapter introduces the **stack**, a linear data structure that follows the **Last In First Out (LIFO)** principle.

The animation illustrates the stack as a tower of blocks where new elements are placed on top. Two main operations are demonstrated:

- **Push**, which adds a new element to the top of the stack
- **Pop**, which removes the most recently added element

Through simple animations, boxes slide upward to represent push operations and slide downward when popped.

The chapter also briefly introduces a practical application of stacks in **expression conversion**, where mathematical expressions can be converted from **infix to postfix notation**. This demonstrates how stacks are used internally in programming language compilers and calculators.



Chapter 4: Queue and Its Variants

The fourth chapter introduces the **queue**, another linear data structure that follows the **First In First Out (FIFO)** principle.

Queues behave similarly to real-life waiting lines where the first person to enter the line is the first one to be served. The animation shows elements entering the queue from the rear and leaving from the front.

The chapter also explores two important variations of queues:

Circular Queue
In a circular queue, the last position connects back to the first position, forming a circular structure. This allows efficient use of available memory space.

Deque (Double Ended Queue)
A deque allows insertion and deletion from both ends, making it more flexible than a standard queue.

Through animated arrows and rotating structures, the chapter demonstrates how these variations work.

Queue Data Structure

Chapter 5: Linked Lists and Memory Management

This chapter introduces **linked lists**, which are dynamic data structures that overcome the fixed-size limitation of arrays.

In a linked list, each element is stored in a **node** containing two parts:

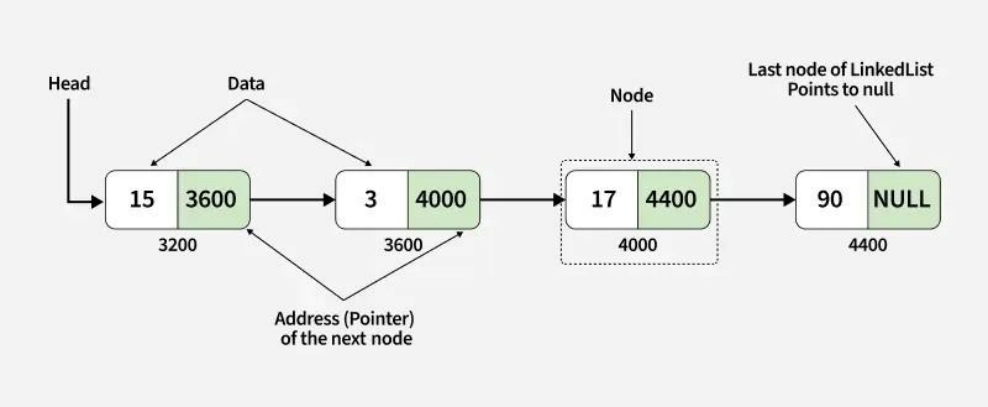
- Data
- A pointer to the next node

Unlike arrays, nodes in a linked list do not need to occupy consecutive memory locations. Instead, they are connected through pointers.

The animation visually represents nodes as boxes connected by arrows. The chapter demonstrates operations such as inserting a node between two existing nodes and removing nodes from the list.

Variations such as **doubly linked lists** and **circular linked lists** are also briefly illustrated.

Additionally, the concept of **dynamic memory allocation** is introduced through operations like malloc and free.



Chapter 6: Trees and Binary Trees

Trees are introduced as hierarchical data structures that resemble branching structures found in nature.

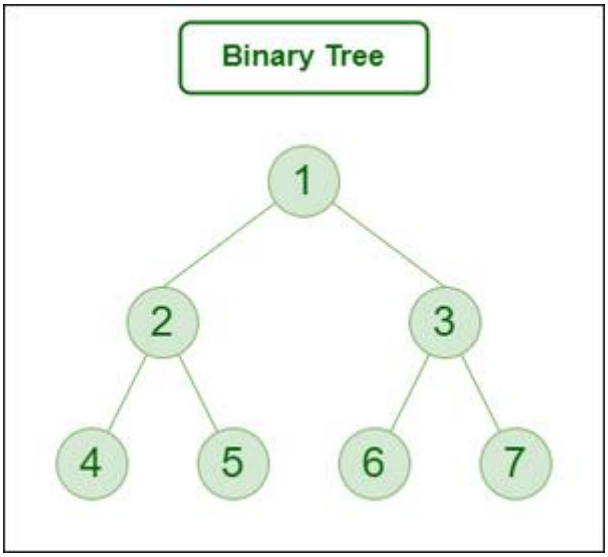
A tree begins with a **root node**, which branches into **child nodes**. These nodes can further branch out, forming a hierarchical structure.

The chapter focuses on **binary trees**, where each node can have at most two children: a **left child** and a **right child**.

The animation demonstrates different **tree traversal methods**, including:

- Preorder traversal
- Inorder traversal
- Postorder traversal

During these traversals, nodes are highlighted sequentially as Algo walks along different paths in the tree.



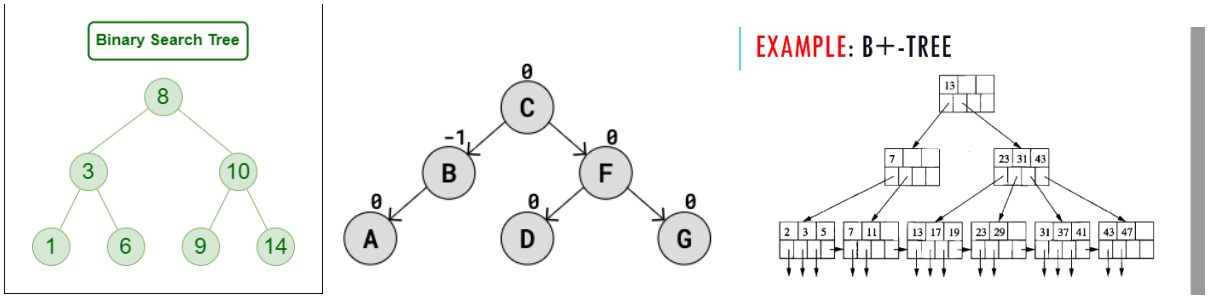
Chapter 7: Advanced Trees (BST, AVL Trees, and B+ Trees)

This chapter explores more advanced tree structures used for efficient data storage and retrieval.

Binary Search Trees (BST) maintain sorted data, allowing faster searching operations. The animation demonstrates how values smaller than a node go to the left while larger values go to the right.

However, BSTs can become unbalanced if elements are inserted in certain orders. This leads to the introduction of **AVL Trees**, which automatically rebalance themselves using rotation operations.

The chapter also introduces **B+ Trees**, which are widely used in database systems and file indexing. These trees can store many keys in a single node, making them efficient for handling large datasets.



Chapter 8: Graphs and Their Representations

Graphs are used to represent networks of interconnected elements. In this chapter, nodes represent entities while edges represent relationships between them.

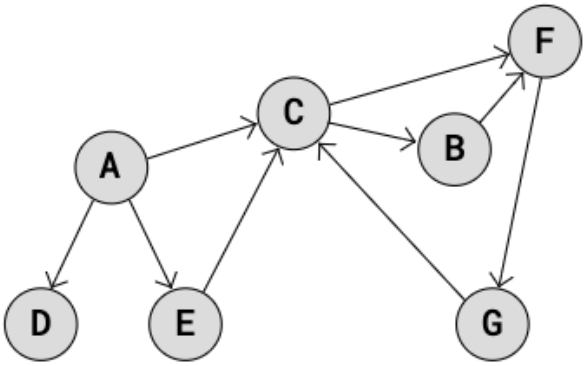
The animation shows a network structure where multiple nodes are connected with lines.

Two main methods of representing graphs are introduced:

Adjacency Matrix
A table where rows and columns represent nodes, and cells indicate connections.

Adjacency List
A list-based representation where each node stores a list of its neighboring nodes.

The animation highlights how both representations store connections differently.



Chapter 9: Graph Traversal (DFS and BFS)

Once graphs are created, algorithms are needed to explore them. This chapter explains two major traversal methods.

Depth First Search (DFS) explores a path deeply before backtracking to explore other paths. The animation shows Algo walking along a single branch until it reaches the end.

Breadth First Search (BFS) explores nodes level by level, moving outward from the starting node. The animation illustrates this as expanding layers, similar to ripples in water.

These traversal techniques are essential in applications such as network analysis, pathfinding, and artificial intelligence.

Chapter 10: Sorting Algorithms (Part 1)

Sorting algorithms organize data into a specific order, such as ascending or descending.

This chapter introduces three simple sorting algorithms:

Bubble Sort repeatedly compares adjacent elements and swaps them if they are in the wrong order.

Insertion Sort builds a sorted portion of the array by inserting elements into their correct positions.

Selection Sort repeatedly selects the smallest element and places it in the correct position.

The animations demonstrate these processes by moving numbers within an array until they become sorted.

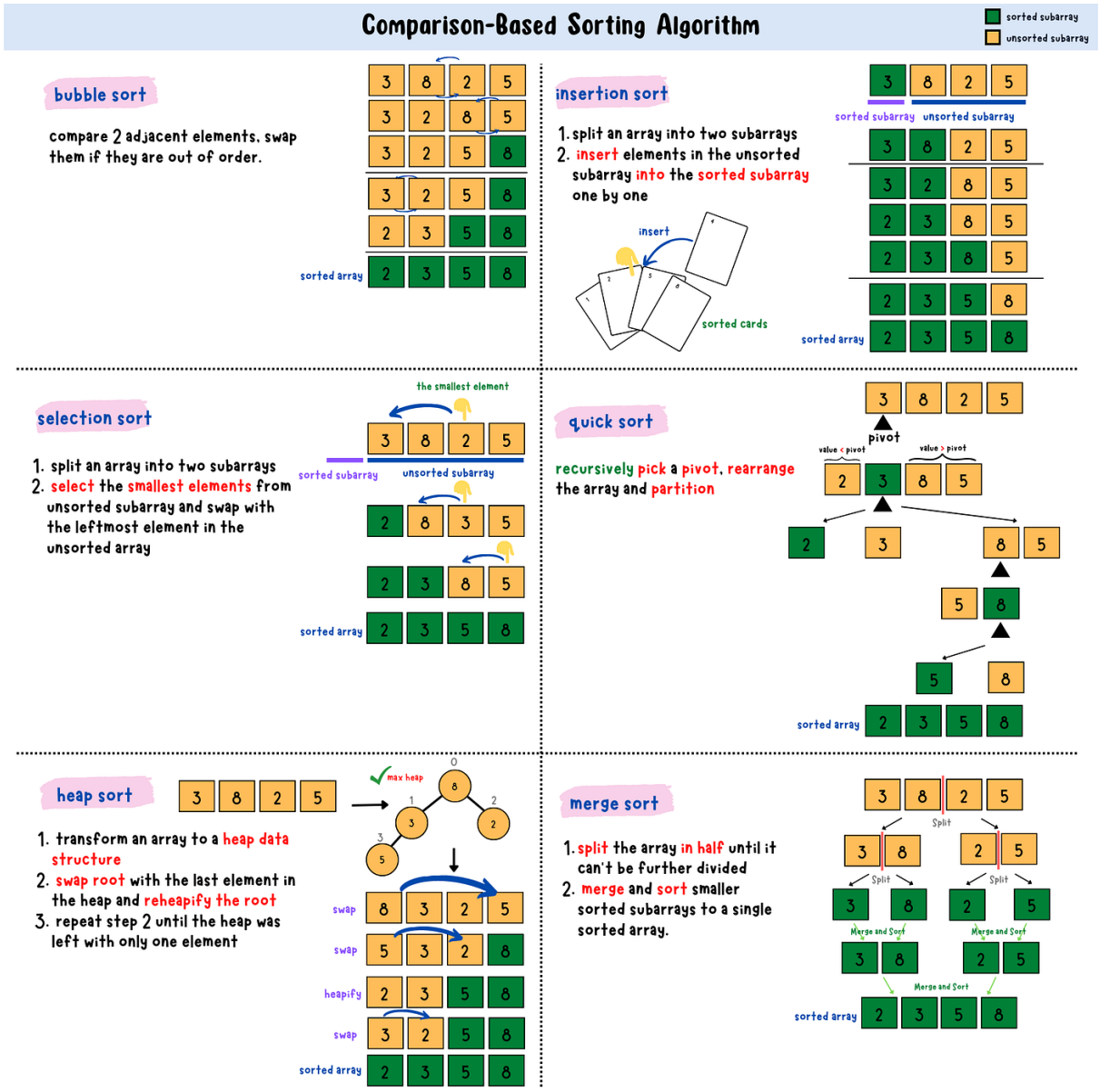
Chapter 11: Sorting Algorithms (Part 2)

The second sorting chapter introduces more advanced algorithms designed for larger datasets.

These include:

- **Merge Sort**, which divides the array into smaller parts and merges them back in sorted order
- **Quick Sort**, which uses a pivot element to partition the array
- **Heap Sort**, which organizes elements using a heap structure
- **Radix Sort**, which sorts numbers based on their digits

Animations demonstrate how these algorithms process large data sets efficiently through structured operations.



Chapter 12: Hashing and Collision Handling

The final chapter introduces **hashing**, a technique used for extremely fast data retrieval.

A **hash function** converts a key value into an index in a hash table. This allows data to be accessed almost instantly without searching through all elements.

However, multiple keys may sometimes map to the same index, creating a **collision**.

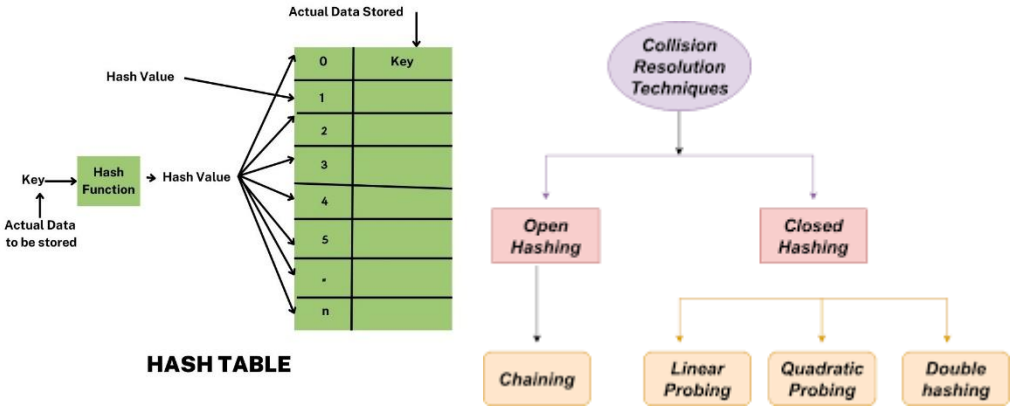
Two major collision resolution techniques are demonstrated:

Open Hashing (Chaining)
Multiple values are stored in a linked list at the same index.

Open Addressing (Closed Hashing)
The system searches for another available slot using methods such as:

- Linear Probing
- Quadratic Probing
- Double Hashing

The animation shows keys entering a “hash machine” and being placed directly into the correct table slot.



6. Closing Scene of the Animation Series

The final animation concludes the learning journey.

Algo reaches the final data world where all the structures combine into a unified **Data Landscape** representing the interconnected nature of computer science.

The final message emphasizes that every piece of data follows a structure and every algorithm tells a story.

Algo walks through a glowing door as the screen fades out, symbolizing the completion of the learning journey and the beginning of deeper exploration into computer science.

7. Conclusion

This project successfully transformed fundamental concepts of Data Structures and Algorithms into a structured animation-based learning series. Through visual storytelling, simplified explanations, and step-by-step demonstrations, the animations make complex topics accessible to beginners.

The series demonstrates the potential of animation as an effective educational tool for teaching technical subjects. By combining creativity with computational concepts, the project provides a unique approach to learning computer science fundamentals.

8. References

1. *Data Structures Using "C" – Lecture Notes* by
   Dr. Subasish Mohapatra, Department of Computer Science and Application,
   College of Engineering and Technology, Bhubaneswar,
   Biju Patnaik University of Technology, Odisha.

   Available at: https://www.cet.edu.in/noticefiles/280_DS%20Complete.pdf

2. **GeeksforGeeks.** *Data Structures and Algorithms Tutorials.*
   Available at: https://www.geeksforgeeks.org

3. **YouTube** - Concept Study

4. **Various Online Educational Resources** used for understanding algorithm visualizations and conceptual explanations.

5. **Project Files and Assets (Google Drive Folder).**
   Link:
   https://drive.google.com/drive/folders/1jnnVEPNmTcAvHMoKbMw23CpKz7l8DY4w?usp=sharing