MIVA OPEN UNIVERSITY

Course:

DATA STRUCTURES

Title:

Description of Various Data Structures

School of Computing

LEARNING OBJECTIVES

At the end of this lesson, you should be able to:

- Define and differentiate between data structures and data types.
- Identify and classify linear and non-linear data structures.
- Describe the characteristics and real-life applications of arrays, linked lists, stacks, queues, trees, and graphs.
- State the significance of choosing the right data structure for solving computational problems.

PERSONAL GOALS FOR THIS TOPIC

_____

_____

_____

_____

_____

_____

FLEX YOUR BRAIN

😊

Before we start, think about the different ways you organise items in your daily life, such as stacking books, arranging files, or managing a to-do list. Based on this, how do you think different data structures might influence how efficiently data can be accessed or manipulated in a computer system?

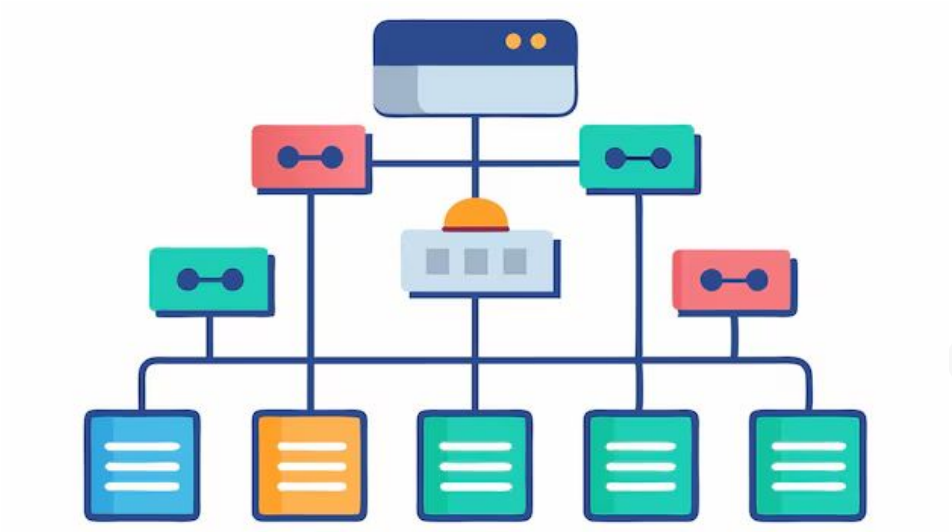
INTRODUCTION

Data structures come in various forms, such as arrays, linked lists, stacks, queues, trees, and graphs. These structures help optimise computational operations like searching, sorting, and managing large sets of data.

From this learning material, you will learn how data structures differ from data types, the classification of linear and non-linear data structures, and learn the characteristics and applications of the most commonly used data structures. Finally, you will be able to select appropriate data structures based on specific problem-solving scenarios and real-world applications.

WHAT IS DATA STRUCTURE?

A data structure is a systematic way of organising, managing, and storing data to allow efficient access and modification. It defines the relationship between the data elements and the operations that can be performed on them, such as searching, insertion, deletion, and traversal.



Common examples of data structures include:

- Arrays
- Linked lists
- Stacks
- Queues
- Trees
- Graphs

How Does Data Structure Vary from Data Type?

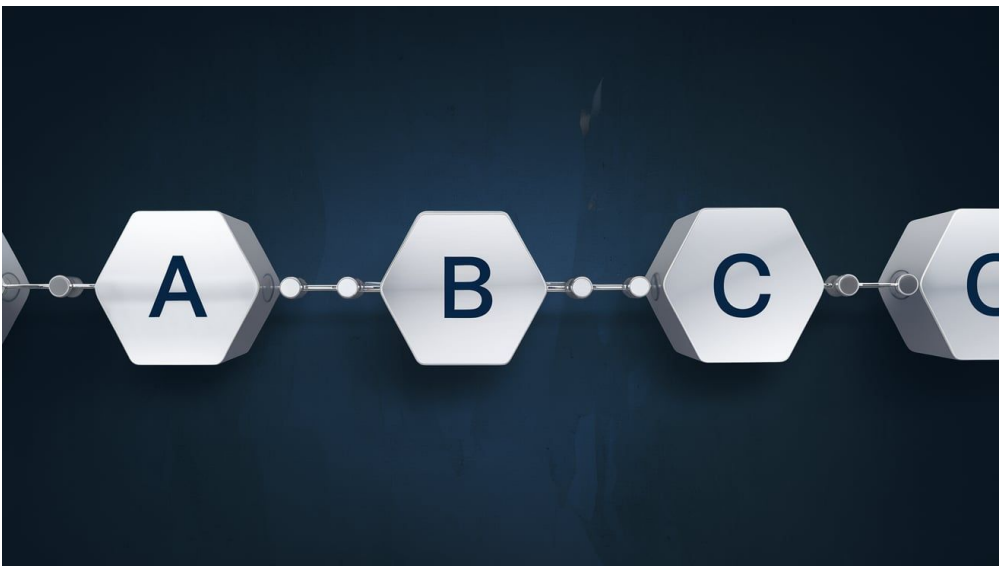
Data Type: Describes the type of data (e.g., integer, float, string).

A data type specifies the kind of data that a variable can hold (e.g., integers, floats, characters).

Data Structure: Defines the format and structure in which data is stored and manipulated.

Data structures, on the other hand, focus on how data is stored and organised in memory.

For example, an array is a data structure that can store multiple values of the same data type in consecutive memory locations.



CLASSIFICATION OF DATA STRUCTURES

Data structures can be broadly classified into two categories:

Linear data structure

In a linear data structure, elements are arranged in a sequential manner. Each element has a unique predecessor and successor, except the first and last elements.

- Static Data Structure: The size of the data structure is fixed (e.g., arrays).
- Dynamic Data Structure: The size of the data structure can change during runtime (e.g., linked lists).

Non-Linear Data Structure

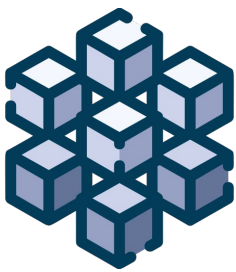
In non-linear data structures, elements are arranged in a hierarchical or graph-like structure. There is no sequential relationship between data elements (e.g., trees, graphs).

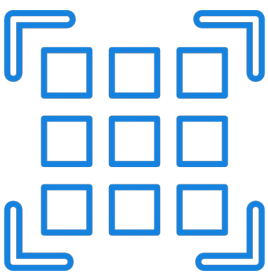
ARRAYS

An array is a collection of elements of the same data type stored in contiguous memory locations. Arrays allow random access to elements using indices, making them ideal for situations where fixed-size data storage is required.

Characteristics of an Array

- Fixed size
- Elements are stored in contiguous memory.
- Allows direct access to any element via its index.



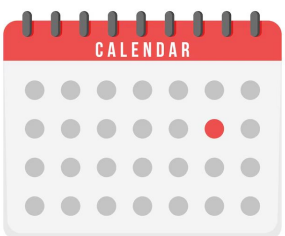


Applications of Array

- Storing lists of data like student scores or daily temperatures.
- Used in sorting and searching algorithms.
- Implementing matrices and tables.

Real-Life Applications of Array

- Storing pixel data in images.
- Representing the days of the month in calendars.



LINKED LIST

A linked list is a dynamic data structure where each element (node) contains data and a reference (or pointer) to the next node. Unlike arrays, linked lists do not require contiguous memory.

Characteristics of a linked list

- Dynamic size.
- Nodes are not stored in contiguous memory.
- Provides efficient insertion and deletion operations.

Applications of linked list

- Implementing stacks, queues, and graphs.
- Efficient memory management.
- Used in dynamic memory allocation.

Real-Life applications of linked list

- Music playlists where songs are linked in sequence.
- Implementing undo functionality in software like word processors.

MID-LESSON QUESTION

Explain the differences between linear and non-linear data structures. Provide examples of each type.

STACK

A stack is a linear data structure that follows the Last In, First Out (LIFO) principle. The last element inserted into the stack is the first one to be removed.

Characteristics of a Stack

- Follows LIFO order.
- Supports two primary operations: push (insert) and pop (remove).
- Only the top element is accessible at any time.

Applications of Stack

- Managing function calls in recursive algorithms.
- Expression evaluation and syntax parsing.
- Undo operations in software.

Real-Life Applications of Stack

- Call stack in programming languages for managing function calls.
- Stacking books, where the last one added is the first to be removed.

QUEUE

A queue is a linear data structure that follows the First In, First Out (FIFO) principle. The first element added to the queue is the first to be removed.

Characteristics of a Queue

- Follows FIFO order.
- Supports two main operations: enqueue (insert) and dequeue (remove).
- Only the front element is accessible for removal.

Applications of Queue

- Task scheduling in operating systems.
- Managing resources in network traffic.
- Handling requests in print queues.

Real-Life Applications of Queue

- Queue management in banks or supermarkets.
- Ticketing systems, where customers are served in the order they arrive.

ADVANCED DATA STRUCTURES

1. Tree

A tree is a non-linear data structure consisting of nodes, where each node has a parent-child relationship. A tree has a root node, and every node (except the root) has exactly one parent.

Characteristics of a Tree

- Hierarchical structure.
- Contains a root node and child nodes.
- Used to represent hierarchical relationships.

Applications of Tree

- Representing hierarchical data, such as file systems.
- Search algorithms like binary search trees.
- Parsing expressions in compilers.

Real-Life Applications of Tree

- Folder structure in operating systems.
- Organisation charts in companies.

2. Graph

A graph is a non-linear data structure made up of nodes (or vertices) connected by edges. Graphs are used to represent pairwise relationships between objects.

Characteristics of a Graph

- Can be directed or undirected.
- Consists of vertices connected by edges.
- Can be weighted or unweighted.

Applications of Graph

- Modelling networks like social media or transportation systems.
- Finding the shortest path in routing algorithms (e.g., Dijkstra's algorithm).
- Representing relationships in graphs of social networks.

Real-Life Applications of Graph

- Google Maps uses graphs to represent and navigate routes between locations.
- Representing and analysing relationships in social networks.

Conclusion

Data structures play an essential role in programming and algorithm design. Understanding the characteristics and applications of different data structures is crucial for solving complex computational problems efficiently. Whether you are working with arrays, linked lists, stacks, queues, trees, or graphs, choosing the right data structure can significantly improve the performance of your program.

SUMMARY OF THE LESSON

Some of the take-away from the lesson are:

- Data structures are systematic ways of organising and storing data.
- Data types define what kind of data can be held, while data structures focus on how data is organised.
- Linear data structures include arrays, linked lists, stacks, and queues.
- Non-linear data structures include trees and graphs.
- Each data structure has specific characteristics and applications that make it suitable for different tasks.

REFERENCES

- Pai, G. a. V. (2008). Data Structures and Algorithms.
- Books Davidson, 2004, Data Structures (Principles and Fundamentals), Dreamtech Press.
- Karthikeyan, Fundamentals, Data Structures and Problem Solving, PHI Learning Pvt. Ltd.
- Samir Kumar Bandyopadhyay, 2009, Data Structures using C, Pearson Education India.
- Sartaj Sahni, 1976, Fundamentals of Data Structures, Computer Science Press.

LESSON QUESTIONS

1. **Which data structure follows the Last In, First Out (LIFO) principle?**

A. Queue

B. Stack

C. Array

D. Linked List

2. **What type of data structure would be most suitable for modelling a social network where users are connected by relationships?**

A. Array

B. Queue

C. Graph

D. Stack

3. **Explain the key differences between arrays and linked lists in terms of memory allocation and access time. Provide a practical scenario where each would be preferable.**

4. **Describe how trees are used in search algorithms like binary search trees and explain their significance in optimising search operations.**

Yipee! You are at the end!

Description of Various Data Structures

MIVA OPEN UNIVERSITY

Study. Anywhere.
Anyone. Anytime.