Data Structures

So, what exactly are data structures? Well, a data structure is a ***storage mechanism*** used to organize and store data on a computer. It provides a ***systematic way*** of arranging data so that it can be ***easily accessed and updated*** as needed.

But remember, data structures and data types are slightly different. Data structure refers to the ***collection of data types arranged in a specific order***. It's crucial to choose the right data structure for your project based on its ***requirements*** and ***complexity***. For example, if you need to store data sequentially in memory, an Array data structure would be a suitable choice.

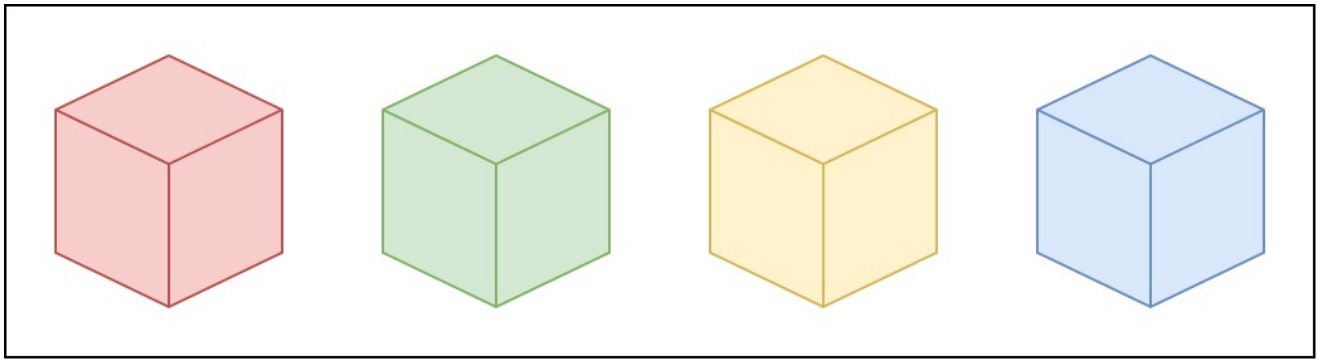
Now, let's explore the two main categories of data structures: ***Linear*** and ***Non-linear***.

Linear Data Structures

In linear data structures, elements are arranged in a ***sequential order***, one after the other. This makes them relatively ***easy to implement and understand***. However, as the complexity of your program grows, linear data structures might not always be the most efficient choice due to operational complexities.

Let's take a closer look at some popular linear data structures:

1. **Array Data Structure:** In an array, elements are stored in ***continuous*** memory locations. All elements in an array are of the ***same type***, and the programming language determines the allowable types.

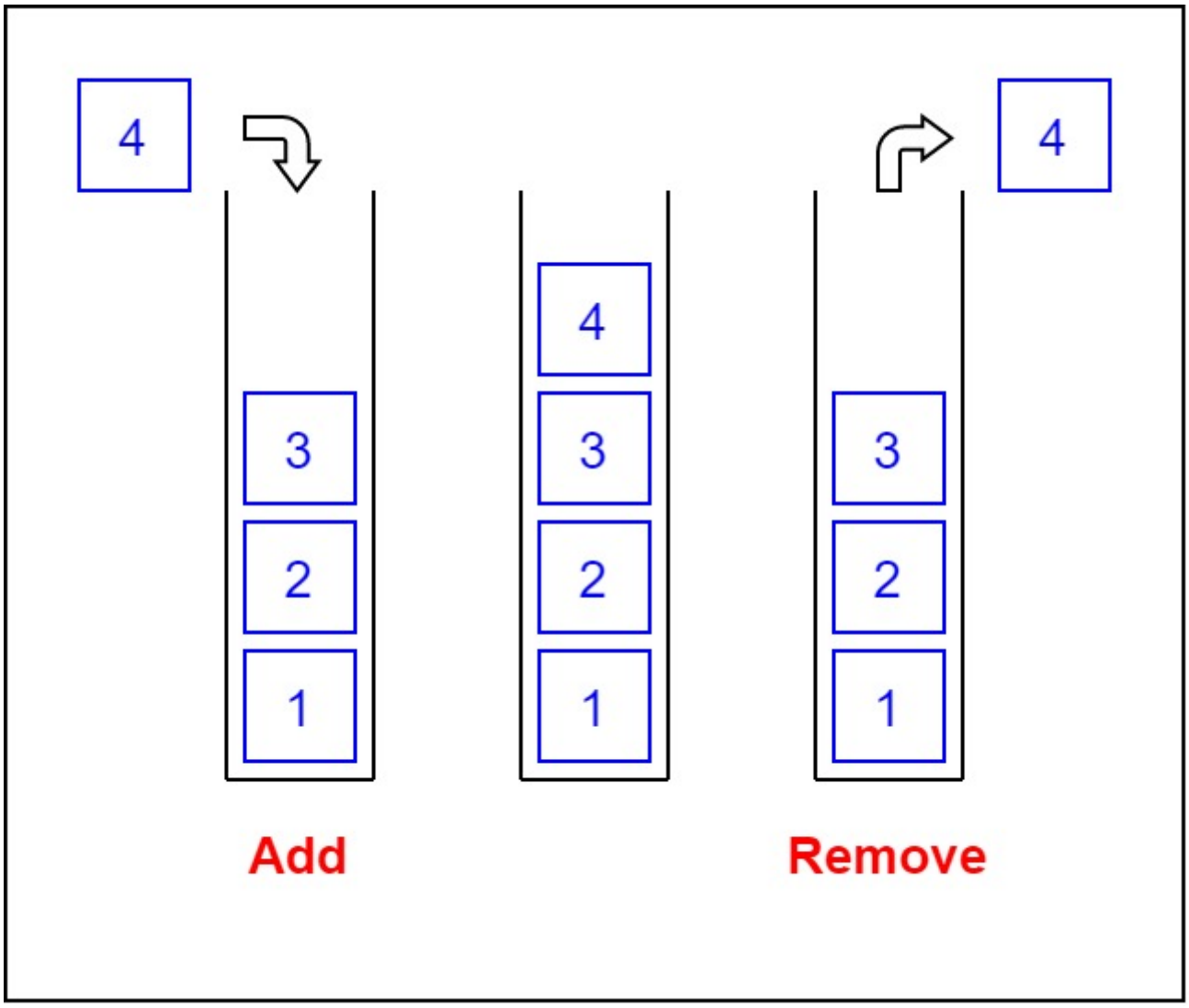


Memory Addresses

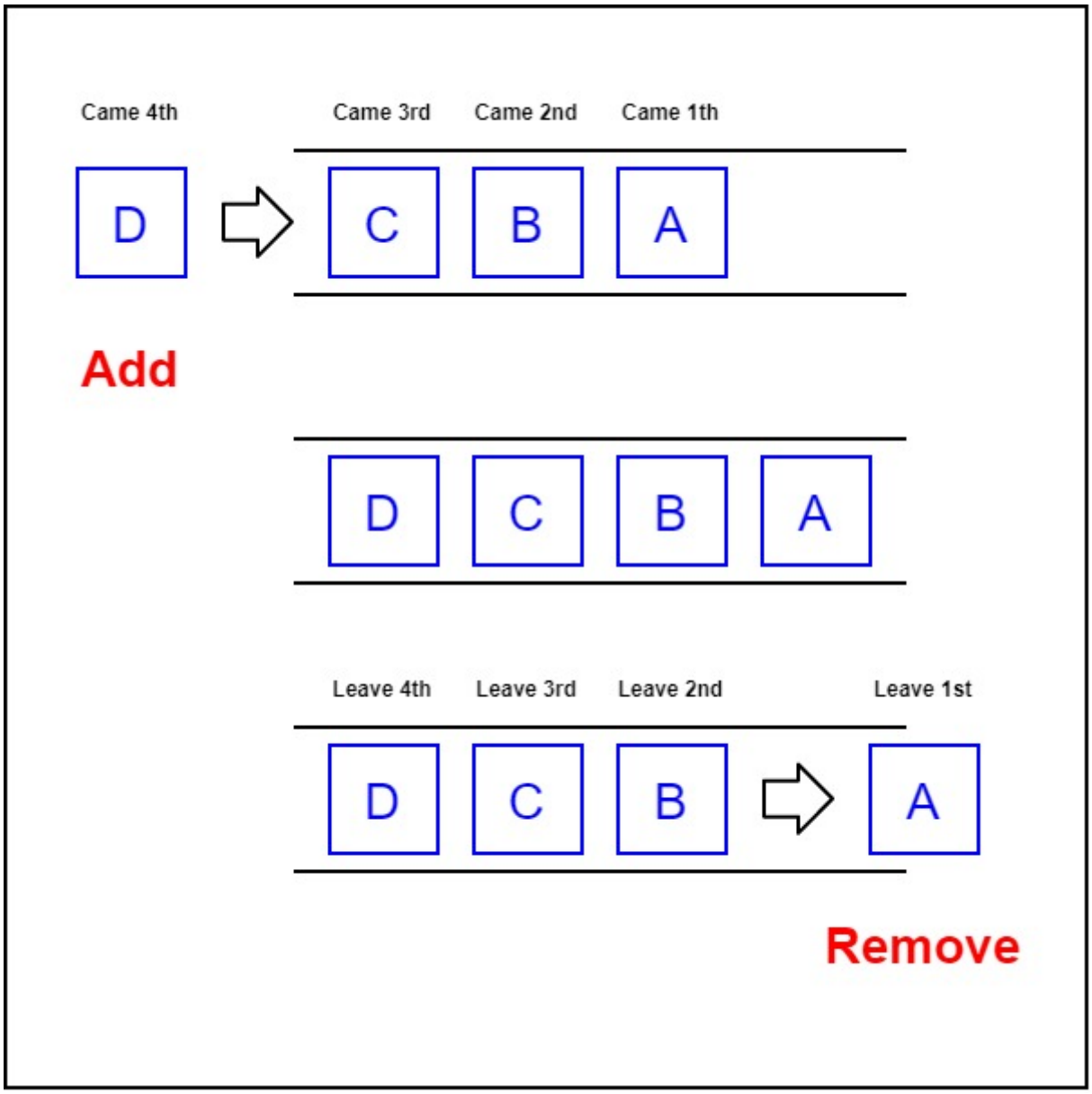
1120	1121	1122	1123	1124	1125	1126	1127
29	86	44	62	18	79	32	54
0	1	2	3	4	5	6	7

Indexes

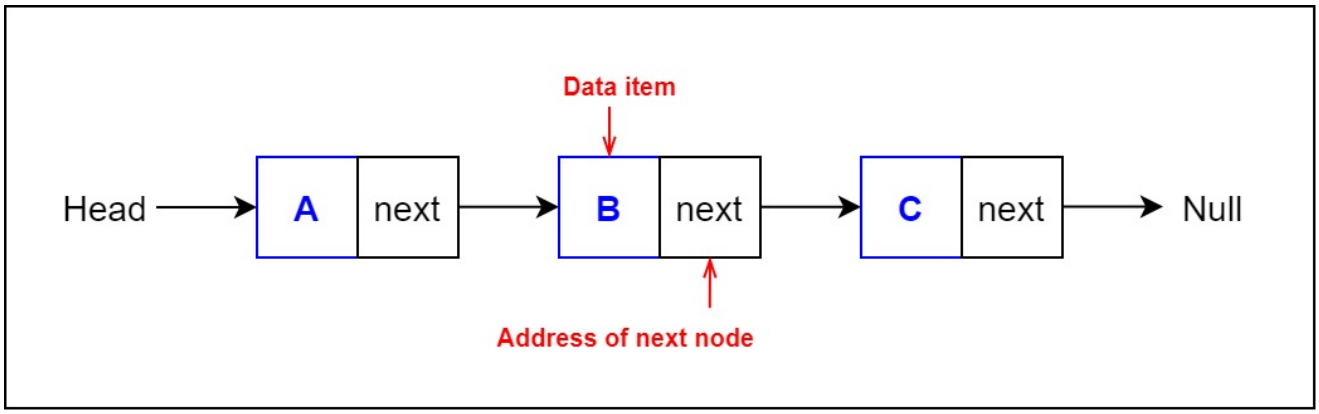
2. **Stack Data Structure:** A stack follows the ***Last In, First Out (LIFO)*** principle. Imagine a pile of plates; the last plate added to the pile will be the first one removed.



3. **Queue Data Structure:** In contrast to a stack, a queue operates on the ***First In, First Out (FIFO)*** principle. Think of a queue of people waiting at a ticket counter; the first person in line will be the first to receive a ticket.



4. **Linked List Data Structure:** In a linked list, data elements are connected through a series of ***nodes***. Each node contains the data item and a ***reference*** to the next node.

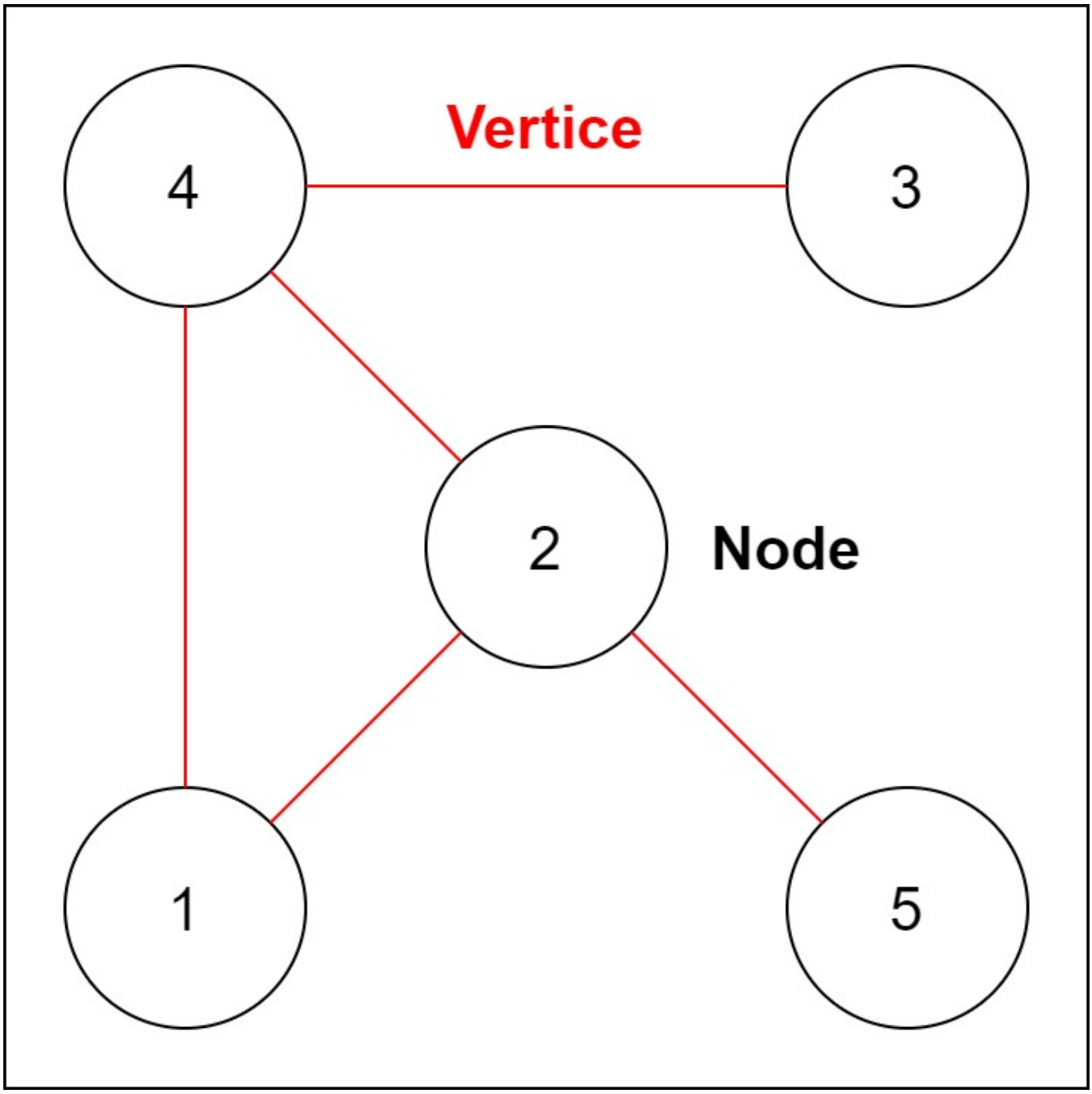


Non-linear Data Structures

In non-linear data structures, elements are not arranged in a sequential order. Instead, they form a ***hierarchical relationship***, where one element can be connected to ***one*** or ***more*** elements.

Here are some examples of non-linear data structures:

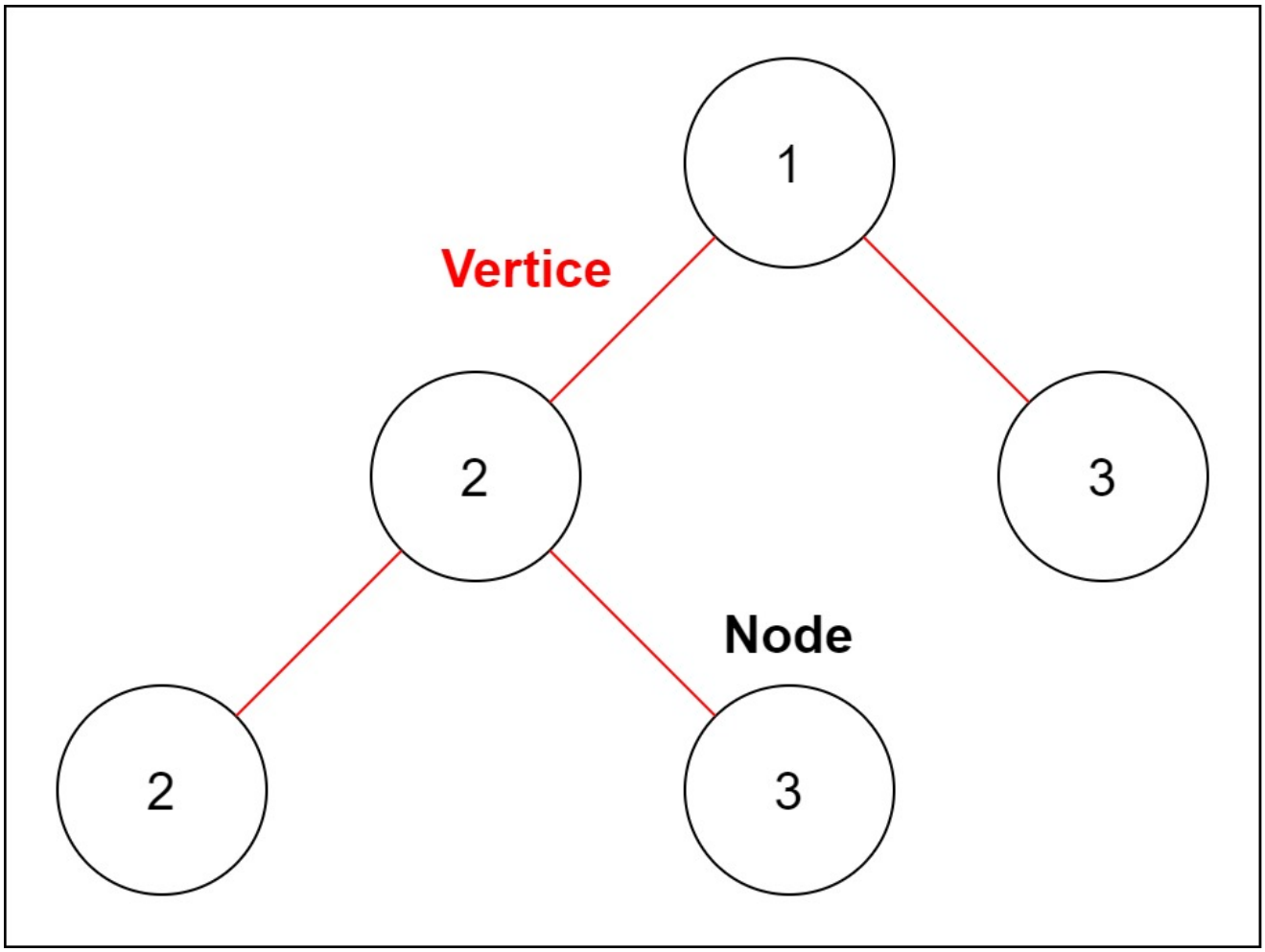
1. **Graph Data Structure:** In a graph, each node is called a ***vertex***, and these vertices are connected to one another through ***edges***.



Some popular graph-based data structures include:

- Spanning Tree and Minimum Spanning Tree,
- Strongly Connected Components,
- Adjacency Matrix,
- Adjacency List.

2. **Trees Data Structure:** A tree is also a collection of ***vertices*** and ***edges*** like a graph. However, in a tree data structure, there can only be ***one edge between two vertices***.



Familiar tree-based data structures include:

- Binary Tree,
- Binary Search Tree,
- AVL Tree,
- B-Tree,
- B+ Tree,
- Red-Black Tree.

Linear vs Non-linear Data Structures

Linear Data Structures	Non-linear Data Structures
The data items are arranged in a ***sequential order***, one after the other.	The data items are arranged in a ***non-sequential***, ***hierarchical*** manner.
All the items are present on the ***single layer***.	The data items are present at ***different layers***.
It can be traversed in a ***single run***, sequentially passing through all elements.	It requires ***multiple runs*** to traverse all elements.

Linear Data Structures	Non-linear Data Structures
The ***memory utilization*** might not be efficient.	Different structures ***utilize memory*** in different efficient ways based on the need.
The ***time complexity*** increases with data size.	***Time complexity*** remains the same, regardless of data size.

Understanding these differences will help you make informed decisions when choosing the right data structure for your projects. So, let's move on to our next lesson and explore each data structure in detail!