What Are Algorithms?

- An algorithm is a clearly defined sequence of instructions used to solve problems or perform tasks. Algorithms are the foundation of all programming and computing operations.
- Time Complexity: This measures the time it takes for an algorithm to run, depending on the size of the input data. Common examples include:
    - O(1): Constant time
    - O(n): Linear time
    - $O(n^2)$: Quadratic time
- Space Complexity: This measures how much memory the algorithm requires as the input size increases.

Searching Methods

- Searching algorithms help find specific elements in a data structure. Two common methods include:
    - Linear Search: Iterates through each element one by one to find the target. Time complexity: O(n).
    - Binary Search: Efficiently finds elements in a sorted array by halving the search space at each step. Time complexity: O(log n).

Types of Data Structures

- Data structures refer to various ways of organizing and storing data, allowing efficient access and modification of the data.
    - Arrays: A linear collection of elements, where each element is stored in contiguous memory. Arrays allow fast access to elements by their index.
    - Linked Lists: A sequence of nodes, where each node holds data and a reference to the next node. They allow dynamic resizing and efficient insertions or deletions.

Common Sorting Techniques

- Sorting algorithms are methods to rearrange data in a specific order, such as ascending or descending. Two popular sorting algorithms are:
    - Bubble Sort: Repeatedly swaps adjacent elements if they are in the wrong order. Time complexity: $O(n^2)$.
    - Merge Sort: A divide-and-conquer algorithm that splits the array into halves, recursively sorts them, and merges them back together. Time complexity: O(n log n).