# MsDS - Data Structures and Algorithms

## Module No: 4

**Revision from Last Lecture**

* Understanding time complexity and space complexity helps predict algorithm behavior and optimise performance and resource usage.
* Big-O notation provides a concise measure for comparing and evaluating the efficiency of different algorithms.
* Recursive algorithms use recurrence relations to analyse computational efficiency and problem resolution. The divide-and-conquer technique simplifies complex problems and is essential in efficient data structure operations.

**Expected Learning Outcomes**

* Recognise and contrast the approaches and efficacy of Binary Search and Linear Search on sorted and unsorted lists.
* Understand the fundamental ideas that guided the creation of different sorting algorithms.
* Understand the various application contexts for sorting algorithms, taking into account elements like storage type and list properties.
* Analyse and determine the most suitable sorting technique based on specific use-cases, such as the nature and size of data lists.

**Linear Search (Unsorted Lists) vs. Binary Search (Sorted Lists) – Cost Comparison**

* Linear search algorithm examines each member in a list until the desired element is found, with a time complexity of O(n).
* Binary search algorithm operates based on divide-and-conquer, requiring a sorted list, and has a time complexity of O(log n).
* Binary search is more efficient for extensive datasets, but requires the list to be sorted, while linear search may be more practical for dynamic settings.

**Design of Sorting Algorithms**

* Sorting algorithms are essential in computer science for organising data efficiently.
* Factors that influence sorting algorithm efficiency include comparison-based vs. non-comparison-based sorting, in-place vs. out-of-place sorting, stability, and adaptivity.
* Different sorting algorithms, such as Bubble Sort, Quick Sort, Merge Sort, and Heap Sort, have their own methodologies and characteristics.

**Application Context of Sorting Algorithms**

* In-memory sorting is efficient for datasets that can fit in RAM, allowing for rapid access and manipulation of data.
* Sequential storage sorting is necessary for datasets that exceed the capacity of RAM, utilising techniques like External MergeSort to minimise disc input/output operations.
* The size of the dataset influences the choice of sorting algorithm, with simpler algorithms like Insertion Sort being more effective for short lists, and advanced algorithms like QuickSort and MergeSort being preferred for long lists.

**Semi-sorted vs. Random Lists**

* The initial order or arrangement of pieces in a dataset can impact the effectiveness of a sorting algorithm.
* Semi-sorted lists have components that are already arranged in a certain order, making incremental modification algorithms more efficient.
* Random lists lack any obvious order, requiring adaptive sorting algorithms like MergeSort and QuickSort.

**Important Terminologies**

* Linear Search: Linear search is a technique used to locate a certain value inside a list by examining each member in a consecutive manner.
* Binary Search: The proposed approach aims to efficiently locate an item inside a sorted list by iteratively partitioning the search interval into two equal halves.
* Sorting Algorithms: This discourse concerns the methodologies used to systematically organise items inside a list or array in a predetermined sequence.
* Recurrence Relations: Equations that establish a correlation between distinct components of a sequence, often used to comprehend the intricacy of recursive functions.
* Divide-and-Conquer: The algorithmic methodology used involves the decomposition of issues into smaller sub-problems, the resolution of each sub-problem, and the subsequent integration of their respective answers.
* Recursive Algorithms: Recursive techniques are algorithms that use fewer inputs to solve bigger issues by repeatedly using the same logical process.
* In-memory Sorting: Sorting algorithms that use main memory, such as RAM, for their operations, hence providing quicker access times.
* Sequential Storage: Refers to data storage methods like disk drives where data is accessed in a sequence, often slower than in-memory operations.
* Semi-sorted Lists: Lists where a portion or majority of elements are already in order, potentially allowing certain sorting algorithms to operate more efficiently.

**Summary**

* Linear search and binary search are essential algorithms for finding data, with binary search offering a significant reduction in time complexity.
* Sorting algorithms have different designs tailored to specific requirements, including factors like stability, adaptivity, and in-place functioning.
* The choice of sorting algorithm depends on the physical storage location of the data and the size of the list, with in-memory sorting being faster but limited by RAM capacity, while sequential storage sorting can handle larger datasets but with potentially longer access times.

THANK YOU