DATA STRUCTURES

Data structures consists of arrays, linked lists, stacks, and queues is crucial for algorithm design. The system life cycle involves various stages from conception to retirement, where data structures play a vital role in system design and implementation. Performance analysis involves evaluating the time and space complexity of algorithms, which are crucial for understanding their efficiency.

Big O is used to describe the upper bound or worst-case scenario of the growth rates of functions. Big Theta indicates tight bounds or average-case scenarios. Big Omega represents the lower bound or best-case scenario.

An algorithm is a step-by-step procedure for solving a problem. Algorithm characterstics include input, output, definiteness, finiteness, effectiveness, and generality.

Performance analysis involves evaluating the efficiency and effectiveness of algorithms in terms of time complexity and space complexity. Time complexity measures the amount of time an algorithm takes to run as a function of the length of its input.

Space complexity measures the amount of memory space an algorithm requires as a function of the length of its input. Big O notation, also known as asymptotic notation, describes the upper bound of the time complexity or space complexity of an algorithm.

Big Omega notation represents the lower bound, while Big Theta notation denotes tight bound of the complexity.

Arrays:

Arrays are fundamental data structures used to store homogeneous elements in contiguous memory locations. Arrays. Sparse matrices, where most elements are zero, require efficient representations for optimization.

Sorting:

Sorting algorithms such as bubble sort, heap sort, and divide and conquer approaches play crucial roles in arranging elements in arrays. Bubble sort compares adjacent elements, and swaps them if they are wrong. Heap sort builds a heap data structure and repeatedly extracts the maximum element to achieve sorting. Divide and conquer approaches, like merge sort and quicksort, break the array into smaller parts, sort them individually, and then merge them back together.

Searching:

Search algorithms are fundamental algorithms used to find elements in arrays.

Linked List:

Linked lists are linear data structures consisting of nodes connected by pointers. They offer dynamic memory allocation and efficient insertion and deletion operations. Self-referential structures enable nodes to reference other nodes, facilitating the creation of complex data structures such as trees and graphs. Singly linked lists, doubly linked lists, and circular linked lists are variations of linked lists with different properties and operations. Doubly linked list is a type of linked list where each node contains a reference not only to the next node but also to the previous node. Circular linked list is a type of linked list where the last node points back to the first node, forming a circular structure.

Stacks:

Stack is a fundamental data structure that follows the Last-In-First-Out (LIFO) principle. It operates on two main stack operations: push and pop. Push operation adds an element to the top of the stack. The pop operation is to remove an element from top.

Queues:

Queue follows the First-In-First-Out (FIFO) principle. The queue supports operations i,e two primary queue operations: enqueue and dequeue. The enqueue operation adds an element to the rear of the queue. The dequeue operation removes the front element. Circular Queue- A circular queue addresses the limitation of linear queues by implementing a circular buffer. Priority Queue: A priority queue is a type of queue where each element has an associated priority. Double-Ended Queue supports insertion and deletion of elements from both ends.

Trees and Graphs:

Trees -. Trees are hierarchical data structures composed of nodes connected by edges, having root nodes and child nodes. Binary trees are another type of trees, having maximum of two children. Binary search trees maintain the property that the left subtree contains nodes with values less than the root, and the right subtree contains nodes with values greater than the root.

Graphs are used to store data. Graphs are non-linear data structures consisting of vertices and edges that represent relationships between objects. Graph traversal algorithms are the Depth-first search and breadth-first search which are used to explore and analyze graphs efficiently.

Hashing:

Hashing - is used for organizing and retrieving information efficiently. Collision resolution , methods that handle situations where multiple keys map to values that are same, ensuring efficient retrieval and storage of data. Hashing functions like mid-square, division, folding, and digit analysis are used to compute hash values which are based on the hashkeys properties.