Data Structures and Algorithms (DSA)

Data Structures and Algorithms form the foundation of efficient problem-solving in computer science. Arrays are a contiguous block of memory used to store elements of the same data type, while binary trees are hierarchical structures where each node has at most two children, facilitating operations like search and traversal. Graphs, on the other hand, represent a set of nodes (vertices) connected by edges and are widely used in networking and pathfinding problems. Dynamic Programming is a powerful algorithmic technique used to solve problems by breaking them into smaller overlapping subproblems, enabling efficient computation. The time complexity of algorithms is crucial to understanding their performance; for example, binary search operates in O(logn)O(\log n)O(logn), bubble sort in O(n2)O(n^2)O(n2), and merge sort in O(nlogn)O(n \log n)O(nlogn). Fundamental data structures like stacks (Last In, First Out), queues (First In, First Out), and linked lists play a pivotal role in problem-solving and system design.

Computer Networks

Computer networks enable communication between devices, structured according to the OSI model, which consists of seven layers. The physical layer manages the actual hardware connections, while the data link layer handles error detection and correction. The network layer, responsible for routing and forwarding, includes protocols like IP. The transport layer ensures reliable communication through protocols like TCP and UDP, while the session layer maintains session integrity between applications. Data formatting and encryption are handled by the presentation layer, with the application layer facilitating user-facing services such as HTTP and FTP. Common protocols like DNS (domain resolution), ARP (address resolution), and HTTP/HTTPS (web communication) form the backbone of the internet. The TCP/IP model, a simplified version of the OSI model, operates across four layers: application, transport, internet, and network access.

Operating Systems

Operating systems are critical for managing computer resources and providing services for software applications. The kernel, the core component of an operating system, oversees CPU usage, memory management, and device control. The scheduler allocates CPU time among processes, ensuring efficient multitasking. A file system organizes and stores data, while virtual memory extends RAM by using disk storage to handle larger workloads. Processes and threads are key execution units in an OS; processes operate independently with separate memory spaces, while threads share the memory space within a single process, enabling lightweight task execution. Scheduling algorithms, such as First-Come-First-Served (FCFS), Round Robin, and Priority Scheduling, determine the order in which processes are executed. Deadlocks, a critical issue in multitasking systems, occur under specific conditions: mutual exclusion, hold and wait, no preemption, and circular wait. Understanding these concepts is essential for designing robust and efficient operating systems.