VIVA QUESTIONS

Introduction to Data Structures

No	Content
Q1	What is the concept of Abstract Data Types (ADT)?
A1	ADT refers to a mathematical model that defines the set of operations that can be performed on a data structure, without specifying the implementation details. It provides a high-level description of a data structure's behavior and properties. ADTs focus on what operations can be performed, not on how they are implemented.
Q2	Differentiate between linear and nonlinear data structures.
A2	Linear data structures store elements sequentially, while nonlinear data structures allow elements to be interconnected in various ways.
Q3	What are the common operations performed on data structures?
A3	The common operations on data structures include insertion, deletion, traversal, searching, and sorting.
Q4	Give examples of linear data structures.
A4	Examples of linear data structures include arrays, linked lists, stacks, and queues.
Q5	How can you define a data structure?
A5	A data structure can be defined as a way of organizing and storing data in a specific format to facilitate efficient operations.
Q6	Explain the concept of well-formedness of parentheses.
A6	Well-formedness of parentheses refers to the balanced arrangement of opening and closing parentheses in an expression.
Q7	What is the postfix notation? How is it useful in evaluating expressions?

A7	Postfix notation, also known as Reverse Polish Notation (RPN), is a way of representing mathematical expressions without the use of parentheses. It simplifies expression evaluation by eliminating the need for operator precedence rules.
Q8	Define recursion and its significance in data structures.
A8	Recursion is a programming technique where a function calls itself during its execution. It is useful in solving problems that can be divided into smaller subproblems. Recursion is commonly used in data structures for tasks such as tree traversal, graph traversal, and divide-and-conquer algorithms.
Q9	What are the types of data structures based on memory allocation?
A9	Data structures can be classified into static data structures (allocated at compile-time) and dynamic data structures (allocated at runtime).
Q10	Explain the concept of an Abstract Data Type (ADT).
A10	An Abstract Data Type (ADT) is a high-level description of a data structure that defines the operations that can be performed on it, without specifying the implementation details. ADTs provide a way to encapsulate data and operations, allowing the implementation details to be hidden and providing modularity and reusability in code.

Linked List

No	Content
Q1	What is a linked list? How does it differ from an array?
A1	A linked list is a linear data structure where elements are stored in separate nodes, each containing a reference to the next node. Unlike an array, linked lists can dynamically grow and shrink in size.
Q2	Explain the concept of singly linked list and doubly linked list.
A2	In a singly linked list, each node contains a reference to the next node. In a doubly linked list, each node contains references to both the next node and the previous node.
Q3	What are the common operations performed on a singly linked list and a doubly linked list?

A3	Common operations on a singly linked list include insertion, deletion, traversal, and searching. Common operations on a doubly linked list also include backward traversal and insertion/deletion at the tail.
Q4	How can you implement a stack and a queue using a singly linked list?
A4	A stack can be implemented using a singly linked list by performing push and pop operations at the head. A queue can be implemented by performing enqueue at the tail and dequeue at the head.
Q5	What is the application of a singly linked list for polynomial representation and addition?
A5	Singly linked lists can be used to represent polynomials, with each node containing a coefficient and an exponent. Addition of polynomials can be performed by adding corresponding terms of the linked lists.

Stack and Queues

No	Content
Q1	What is a stack? Explain its Last-In-First-Out (LIFO) property.
A1	A stack is a linear data structure that follows the Last-In-First-Out (LIFO) principle, where the last element inserted is the first one to be removed.
Q2	What are the common operations performed on a stack?
A2	The common operations performed on a stack are push (inserting an element onto the stack) and pop (removing the top element from the stack).
Q3	How can you implement a stack using an array?
A3	A stack can be implemented using an array by maintaining a variable to keep track of the top element and using array operations to insert and remove elements.
Q4	Explain the applications of a stack in real-life scenarios.
A4	Some applications of a stack include function calls and recursion, expression evaluation and conversion, undo/redo functionality, and backtracking in algorithms.
Q5	Describe the circular queue and its advantages.

A5	A circular queue is a variation of a regular queue where the rear and front elements are connected. It allows efficient space utilization and avoids unnecessary shifting.
Q6	What is a priority queue? How is it different from a regular queue?
A6	A priority queue is a variation of a queue where each element has a priority assigned to it. Higher priority elements are dequeued first compared to lower priority elements.
Q7	How can you implement a queue using an array?
A7	A queue can be implemented using an array by maintaining two pointers, one for the front and one for the rear, to keep track of the elements.
Q8	Describe the double-ended queue (deque) and its applications.
A8	A double-ended queue, or deque, is a linear data structure that allows insertion and deletion of elements from both ends. It can be used in scenarios requiring both stack and queue operations.
Q9	Explain the concept of linear search and binary search.
A9	Linear search involves iterating through the elements of a collection until the target element is found. Binary search is a more efficient search algorithm that requires a sorted collection and repeatedly divides the search space in half.
Q10	What is hashing and what are collision resolution techniques?
A10	Hashing is a technique to map data to a fixed-size array index using a hash function. Collision resolution techniques are used to handle situations where two elements map to the same index, such as separate chaining and open addressing.

Trees

No	Content
Q1	Explain the concept of a tree and its terminologies.
A1	A tree is a nonlinear data structure composed of nodes connected by edges. Terminologies include root, parent, child, sibling, leaf, depth, height, and level of a tree.

Q2	**How can a binary tree be represented?**
A2	**A binary tree can be represented using linked representations such as a structure with left and right pointers or an array-based representation using indices.**
Q3	**What are the common tree traversal methods?**
A3	**Common tree traversal methods include in-order, pre-order, and post-order traversals, which define the order in which the nodes are visited.**
Q4	**What is a binary search tree (BST)? How does it differ from a regular binary tree?**
A4	**A binary search tree is a binary tree where the value of each node in the left subtree is less than the value of the node, and the value of each node in the right subtree is greater than the value of the node. A regular binary tree has no specific ordering criteria for its nodes.**
Q5	**What are the applications of binary trees, such as expression trees, Huffman encoding, and AVL trees?**
A5	**Binary trees have applications in expression evaluation, Huffman encoding for data compression, and AVL trees for efficient searching and sorting.**
Q6	**Explain the concept of an expression tree and its use in evaluating arithmetic expressions.**
A6	**An expression tree is a binary tree where the operators are stored at internal nodes, and the operands are stored at the leaf nodes. Expression trees can be used to evaluate arithmetic expressions by traversing the tree and performing the corresponding operations.**
Q7	**What is Huffman encoding? How does it work?**
A7	**Huffman encoding is a data compression technique that assigns shorter codes to more frequently occurring characters and longer codes to less frequently occurring characters. It works by constructing a Huffman tree based on the frequency of characters in the input data.**
Q8	**What are AVL trees? How are rotations used to maintain balance in AVL trees?**

A8	**AVL trees are self-balancing binary search trees where the heights of the left and right subtrees differ by at most one. Rotations are used to maintain balance in AVL trees by reorganizing the tree structure based on certain conditions.**
Q9	**What is a B-tree? How does it differ from a binary search tree?**
A9	**A B-tree is a self-balancing search tree that can have multiple keys per node and multiple child nodes. It differs from a binary search tree by allowing a variable number of keys per node and supporting efficient disk-based operations.**
Q10	**What is a B+ tree? How is it useful in database systems?**
A10	**A B+ tree is a variation of a B-tree where only keys are stored in internal nodes, and data records are stored in the leaf nodes. It is useful in database systems for efficient range queries, sequential access, and indexing.**

Q6	**Explain the concept of a tree and its terminologies.**
A6	**A tree is a nonlinear data structure composed of nodes connected by edges. Terminologies include root, parent, child, sibling, leaf, depth, height, and level of a tree.**
Q7	**How can a binary tree be represented?**
A7	**A binary tree can be represented using linked representations such as a structure with left and right pointers or an array-based representation using indices.**
Q8	**What are the common tree traversal methods?**
A8	**Common tree traversal methods include in-order, pre-order, and post-order traversals, which define the order in which the nodes are visited.**
Q9	**What is a binary search tree (BST)? How does it differ from a regular binary tree?**

A9	A binary search tree is a binary tree where the value of each node in the left subtree is less than the value of the node, and the value of each node in the right subtree is greater than the value of the node. A regular binary tree has no specific ordering criteria for its nodes.
Q10	What are the applications of binary trees, such as expression trees, Huffman encoding, and AVL trees?
A10	Binary trees have applications in expression evaluation, Huffman encoding for data compression, and AVL trees for efficient searching and sorting.

Graphs

No	Content
Q1	What is a graph? Explain its terminologies.
A1	A graph is a nonlinear data structure consisting of a set of nodes (vertices) and a set of edges. Terminologies include vertices, edges, degree, adjacency, path, and cycle.
Q2	How can a graph be represented?
A2	A graph can be represented using various methods, such as an adjacency matrix, an adjacency list, or an edge list.
Q3	What are the common graph traversal methods?
A3	Common graph traversal methods include Depth-First Search (DFS) and Breadth-First Search (BFS).
Q4	Explain the topological sorting of a directed acyclic graph (DAG).
A4	Topological sorting of a DAG is an ordering of its vertices such that for every directed edge (u, v), vertex u comes before v in the ordering. It is useful in scheduling tasks with dependencies.
Q5	What are the applications of graphs?
A5	Graphs have applications in various domains, including social networks, computer networks, routing algorithms, recommendation systems, and optimization problems.
Q6	How can you detect cycles in a graph?

A6	**Cycles in a graph can be detected using algorithms such as Depth-First Search (DFS) or by applying cycle detection algorithms like Tarjan's algorithm or Kosaraju's algorithm.**
Q7	**What is a minimum spanning tree (MST)?**
A7	**A minimum spanning tree is a tree that spans all the vertices of a connected, weighted graph with the minimum total weight. It is useful in network design and optimizing costs.**
Q8	**How can you find the shortest path between two vertices in a graph?**
A8	**Shortest paths between two vertices in a graph can be found using algorithms such as Dijkstra's algorithm or the Bellman-Ford algorithm.**
Q9	**Explain the concept of a directed graph and an undirected graph.**
A9	**In a directed graph, edges have a direction, allowing traversal in only one direction. In an undirected graph, edges have no direction, allowing traversal in both directions.**
Q10	**What is the difference between a graph and a tree?**
A10	**A graph is a collection of nodes connected by edges, whereas a tree is a specific type of graph with no cycles and a hierarchical structure. Trees are acyclic graphs.**