SEMESTER-II: PAPER-III

DATA STRUCTURES QUESTION BANK

Unit I: Introduction to Data Structures, Arrays, and Sorting

1 Mark Questions

1. Define a data structure.
2. What is a linear data structure?
3. Give an example of a non-linear data structure.
4. What is a primitive data structure?
5. Define a non-primitive data structure.
6. What is a one-dimensional array?
7. How is an array element accessed in memory?
8. What is row-major order representation?
9. What is column-major order representation?
10. Define the base address of an array.
11. What is the time complexity of linear search in the worst case?
12. What is the best-case time complexity of binary search?
13. State the primary condition required to perform binary search.
14. What is the core idea behind bubble sort?
15. Define the term 'Traversing' an array.
16. What does the operation 'Merge' mean in arrays?
17. What is a multi-dimensional array?
18. Which sorting algorithm iteratively selects the minimum element and places it at the beginning?
19. What is the space complexity of an in-place sorting algorithm like Selection Sort?
20. Why is an array called a static data structure?

2 Mark Questions

1. Differentiate between linear and non-linear data structures.
2. Explain the concept of memory representation of a 1D array.
3. Define the address formula for row-major order in a 2D array.
4. Define the address formula for column-major order in a 2D array.
5. Explain the 'Insertion' operation in a 1D array with its worst-case complexity.
6. Explain the 'Deletion' operation from a 1D array.

7. Compare Linear Search and Binary Search based on efficiency.

8. Describe how Selection Sort works in a single sentence.

9. Why is Bubble Sort called 'bubble' sort? Explain.

10. What are the limitations of a multi-dimensional array in terms of memory?

11. Explain the merging operation of two sorted arrays.

12. What is the classification of data structures? Draw a brief hierarchy.

13. How does an array handle homogeneous elements? Explain.

14. What is the time complexity of Selection Sort in best and worst cases?

15. What is the worst-case number of comparisons in a Bubble Sort for 'n' elements?

5 Mark Questions

1. Explain the classification of data structures with suitable examples for each category.

2. Derive the address calculation formula for an element A[i][j] in a 2D array stored in Row-Major and Column-Major order.

3. Write an algorithm to insert an element at a specific position in a one-dimensional array.

4. Write an algorithm to delete an element from a given index in a 1D array.

5. Explain the Binary Search algorithm with a step-by-step example on a sorted array of 7 elements.

6. Write a C/C++ function or clear algorithm to implement Linear Search and discuss its time complexity.

7. Illustrate the working of Bubble Sort with a complete trace on the array: [5, 1, 4, 2, 8].

8. Explain Selection Sort algorithm and sort the following array step-by-step: [29, 10, 14, 37, 13].

9. Write an algorithm to merge two sorted arrays into a third sorted array.

10. Discuss the concept of multi-dimensional arrays and how they are represented linearly in memory blocks.

8 Mark Questions

1. Provide an exhaustive comparison between Linear Search and Binary Search. Write complete algorithms for both and derive their best, average, and worst-case time complexities.

2. Explain the memory mapping of two-dimensional arrays. Derive the address generation formulas for both Row-Major and Column-Major orders, and calculate the address of A[15][20] if the base address is 1000, size of element is 4 bytes, and array bounds are A[0...30][0...40].

3. Design and detail the Bubble Sort algorithm. Write a full function to implement it, explain how it can be optimized using a swapped flag, and analyze its worst-case and best-case behaviors with tracking passes.

4. Develop the Selection Sort algorithm completely. Show its execution mechanism, provide the formal pseudocode, analyze its total number of comparisons and swaps, and explain why its time complexity remains O(n^2) across all cases.

5. Detail all major primitive and non-primitive array operations (Traversing, Searching, Insertion, Deletion, and Merging). Provide precise algorithmic representations and complexity boundaries for each operation.

Unit II: Linked Lists

1 Mark Questions

1. What is a linked list?
2. Name the two fields contained in a single linked list node.
3. What does the pointer field of the last node in a singly linked list contain?
4. What is a double linked list?
5. How many pointer fields are present in a node of a doubly linked list?
6. What is a circular linked list?
7. What is a double circular linked list?
8. Which data structure is efficient for representing sparse matrices dynamically?
9. What is a sparse matrix?
10. How is the end of a circular linked list identified?
11. What is the main advantage of a linked list over an array?
12. What is the time complexity to insert a node at the beginning of a singly linked list?
13. What is the time complexity to search an element in a singly linked list?
14. What does 'head' pointer store in a linked list?
15. What operation combines two linked lists into one?
16. How do you represent a polynomial term using a linked list node?
17. What is the structural drawback of a single linked list during backward traversal?
18. Define memory representation of a node in a double linked list.
19. What value is assigned to the 'prev' pointer of the first node in a double linked list?
20. Can we perform binary search directly on a linked list efficiently?

2 Mark Questions

1. Differentiate between an array and a linked list dynamic allocation.
2. Explain the node structure of a Doubly Linked List with a small code structure declaration.
3. What is a circular linked list? Mention its key advantage.
4. Explain how a polynomial is represented using a linked list.
5. Describe the deletion of the first node in a singly linked list.
6. How do you insert a node at the beginning of a singly linked list?
7. What is a sparse matrix? Explain its triplet representation scheme.
8. Explain the concept of a double circular linked list.
9. How does the merge operation function on two separate single linked lists?
10. Explain traversing operation in a circular linked list.
11. What are the disadvantages of using linked lists?
12. Why is memory allocation in linked lists considered dynamic?

13. Show how to delete a node from the end of a singly linked list.

14. Explain the role of the 'next' pointer field in a linked list.

15. How do you access the previous node of a current node in a double linked list?

5 Mark Questions

1. Write an algorithm to insert a new node at a given position in a Singly Linked List.

2. Write an algorithm to delete a node with a specific value from a Singly Linked List.

3. Explain the structures and pointer manipulations required to insert a node in a Doubly Linked List at the middle.

4. Explain the deletion operation in a Doubly Linked List from a specified position.

5. Discuss the traversal and search operations in a Circular Linked List with proper pseudocode.

6. Explain how a Sparse Matrix can be represented efficiently using linked lists instead of a 2D array.

7. Describe an algorithm to add two polynomials represented using linked lists.

8. Write an algorithm to merge two sorted singly linked lists into a single sorted linked list.

9. Explain the advantages and structural differences between a Circular Linked List and a Double Circular Linked List.

10. Write a function or detailed algorithmic steps to reverse a Singly Linked List by updating its pointers.

8 Mark Questions

1. Comprehensive Guide to Singly Linked Lists: Write complete, step-by-step algorithms for (a) Insertion at beginning, middle, and end, and (b) Deletion from beginning, middle, and end. Include precise pointer tracking commands.

2. Detail the architecture of a Doubly Linked List. Provide the full node design and write formal algorithms to insert a node before a given reference node and delete an arbitrary target node, explaining the updates to both 'next' and 'prev' links.

3. Discuss Polynomial representation and addition using linked lists. Write a complete, comprehensive algorithm to perform the addition of two multi-variable polynomials, illustrating node creation and coefficient/exponent mapping.

4. Explain Sparse Matrix representation. Contrast the traditional array-based representation with the linked list-based header node representation. Provide algorithms for inserting elements and analyze space-time efficiencies.

5. Analyze Circular and Double Circular Linked Lists thoroughly. Write algorithms to perform traversal, node insertion, and node deletion in a double circular linked list, highlighting how boundary conditions (empty list, single node) are handled safely.

Unit III: Stacks and Queues

1 Mark Questions

1. Define a stack data structure.

2. What acronym describes the entry/exit mechanism of a stack?
3. Name the pointer that tracks the top element of a stack.
4. What is the condition when a stack is full and an insertion is attempted?
5. What is stack underflow?
6. Name the two primary operations performed on a stack.
7. What does the STATUS operation of a stack check?
8. What is an infix expression notation?
9. What is a postfix expression notation?
10. Convert the infix expression 'A + B' to postfix form.
11. What data structure is inherently used to implement recursion?
12. What is the minimum number of moves required to solve the Tower of Hanoi problem with 'n' disks?
13. Define a queue data structure.
14. What acronym describes the entry/exit mechanism of a queue?
15. Name the two pointers used in a standard queue structure.
16. What is the Enqueue operation?
17. What is the Dequeue operation?
18. What is a Circular Queue?
19. What is a Double-Ended Queue (Deque)?
20. What is a Priority Queue?

2 Mark Questions

1. Explain PUSH and POP operations with their basic index updates.
2. Differentiate between Infix, Prefix, and Postfix notations.
3. Why is a circular queue preferred over a linear queue implemented via arrays?
4. Explain the STATUS operation in stack and queue structures.
5. Describe how a stack can be represented using a linked list structure.
6. Explain the basic rules of the Tower of Hanoi problem.
7. What is an input-restricted deque and an output-restricted deque?
8. How does a priority queue handle elements with identical priorities?
9. Write a recursive mathematical definition for the Fibonacci sequence.
10. Explain the application of stacks in function call management.
11. How are the Front and Rear pointers initialized in an empty array-based queue?
12. What happens during a Dequeue operation on an empty queue?
13. Convert the infix expression 'A * B + C' into its postfix equivalent manually.
14. Explain the concept of queue overflow in a static array representation.
15. List two distinct applications of a queue in computer operating systems.

5 Mark Questions

1. Write an algorithm to implement standard PUSH and POP operations on an array-based stack with proper boundary checks.

2. Explain the algorithm to convert an Infix expression to a Postfix expression using an evaluation stack.

3. Write an algorithm to evaluate a given Postfix expression using a stack, showing intermediate steps.

4. Explain how a Stack can be represented and implemented using a Singly Linked List, detailing push and pop transitions.

5. Describe the recursive solution for the Tower of Hanoi problem and write its algorithmic steps for 'n' disks.

6. Write an algorithm to perform Enqueue and Dequeue operations on a standard linear queue using arrays.

7. Explain the working and index wrap-around formulas for Enqueue and Dequeue operations in a Circular Queue.

8. Discuss the concept of Deque (Double-Ended Queue) and explain its types and variations.

9. Explain Priority Queues, detailing their memory representations and real-world execution scenarios.

10. Describe how a Queue can be implemented using a Singly Linked List, tracking front and rear updates.

8 Mark Questions

1. Detail the Infix-to-Postfix conversion algorithm using stacks. Trace the complete conversion step-by-step for the expression: 'A + (B * C - (D / E ^ F) * G) * H', displaying the stack state and partial outputs at every token change.

2. Analyze stack allocation models. Write complete, production-ready algorithms for array-based and linked list-based stacks. Critically evaluate both models regarding spatial complexity, operational speed, and dynamic flexibility.

3. Explain the Circular Queue data structure comprehensively. Provide the complete mathematical conditions for Queue Full and Queue Empty in a circular array, and write formal algorithms for both Enqueue and Dequeue operations.

4. Provide a comprehensive study on recursion using stacks. Explain stack frame allocation during recursive calls using the Factorial or Fibonacci sequence as an anchor, and write a complete non-recursive stack-driven alternative algorithm.

5. Develop the entire operational framework for Deque and Priority Queues. Write functional algorithms for insertion and deletion across both structures, contrast max-priority vs min-priority queues, and elaborate on three distinct real-world system applications.

Unit IV: Trees and Graphs

1 Mark Questions

1. Define a tree data structure.

2. What is a root node in a tree?

3. Define the degree of a tree node.

4. What is a leaf node?

5. What is a binary tree?
6. State the maximum number of nodes at level 'L' of a binary tree.
7. Define a Binary Search Tree (BST).
8. What is an AVL tree?
9. Define the balance factor of a node in an AVL tree.
10. What is an M-Way search tree?
11. In which tree traversal is the root node visited first?
12. What sequence does Inorder traversal follow?
13. Which traversal of a BST produces elements in a sorted ascending order?
14. Define a graph data structure.
15. What are the two core sets that define a graph?
16. What is a directed graph (digraph)?
17. What is an adjacency matrix?
18. Name the traversal technique for graphs that utilizes a queue structure.
19. Name the traversal technique for graphs that utilizes a stack structure.
20. What is a connected graph?

2 Mark Questions

1. Differentiate between a binary tree and a general tree.
2. List the properties of a strict or full binary tree.
3. Explain the linear (array) representation of a binary tree.
4. Explain the linked list representation of a binary tree node.
5. Define the terminologies: 'Path' and 'Ancestors' in a tree.
6. What is the primary structural purpose of an AVL tree? Explain.
7. Compare an AVL tree with a standard Binary Search Tree.
8. Differentiate between a directed graph and an undirected graph.
9. Explain the linked list (adjacency list) representation of a graph.
10. What is a cycle or loop inside a graph structure?
11. Describe the core operational difference between BFS and DFS graph traversals.
12. What are the standard applications of a tree data structure?
13. Define an out-degree and in-degree of a vertex in a directed graph.
14. What is an M-Way search tree? List one of its key advantages.
15. What does 'Searching' a key in a Binary Search Tree involve?

5 Mark Questions

1. Explain the common tree terminologies: Root, Child, Parent, Sibling, Leaf, Level, Depth, and Height.
2. Discuss the structural properties of a Binary Tree, including node bounds and depth properties.

3. Write algorithms for the three standard binary tree traversals: Inorder, Preorder, and Postorder.

4. Explain the Binary Search Tree (BST) insertion mechanism and construct a BST from the keys: [45, 15, 79, 90, 10, 55, 12].

5. Introduce AVL Trees, detail the balance factor computation, and list the four types of balancing rotations (LL, RR, LR, RL).

6. Explain the concept and structural parameters of M-Way Search Trees and B-Trees with a suitable layout diagram.

7. Describe the set, adjacency matrix, and adjacency list representation methods for graphs.

8. Write an algorithm to perform Breadth-First Search (BFS) traversal on a graph using a queue.

9. Write an algorithm to perform Depth-First Search (DFS) traversal on a graph using a stack or recursion.

10. Discuss three major real-world applications of Graph data structures in network routing or social networks.

8 Mark Questions

1. Explain Binary Search Trees thoroughly. Write full algorithms to perform (a) Key Insertion, (b) Key Searching, and (c) Key Deletion (covering all three cases: leaf node, single-child node, and two-children node).

2. Analyze Binary Tree Traversals. Write full recursive algorithms for Inorder, Preorder, and Postorder traversals. Given the Inorder traversal: [D, B, E, A, F, C, G] and Preorder traversal: [A, B, D, E, C, F, G], reconstruct the unique binary tree step-by-step.

3. Provide an exhaustive analysis of AVL Trees. Detail the balancing conditions, explain the explicit transformations for Single Rotations (LL, RR) and Double Rotations (LR, RL), and show structural state insertions for keys: [10, 20, 30, 40, 50, 25].

4. Develop the Breadth-First Search (BFS) and Depth-First Search (DFS) algorithms for graph traversals. Provide full procedural algorithms for both, analyze their time and space complexities, and trace their paths across an arbitrary 5-vertex connected graph.

5. Write a definitive essay on graph representations and structural topologies. Detail and code templates for Adjacency Matrices, Adjacency Lists, and Set structures. Critically compare them based on edge query execution speeds, memory usage overheads, and sparseness factors.