Important DSA Questions on Trees, Graphs, Linked Lists, and Queues

1. Trees

- Binary Tree Traversals: Given preorder and postorder sequences, draw the binary tree.
- AVL Trees: Construct an AVL tree from a series of keys, showing rotations and adjustments to maintain balance.
- B-Trees: Create a B-Tree of a specific order from a given set of keys.
- Binary Search Trees: Convert a sorted doubly linked list into a balanced binary search tree.
- Tree Diameter: Calculate the diameter of a binary tree.

Important & Repeated Questions: AVL Tree construction, diameter calculation, and BST conversion from a linked list appear frequently.

2. Graphs

- Graph Traversals: Perform breadth-first search on a given weighted graph, presenting possible traversal orders.
- Minimum Spanning Trees: Differentiate between Prim's and Kruskal's algorithms; find the minimum spanning tree of a graph.
- Shortest Paths: Find all pairs of shortest paths using Floyd's algorithm.
- Dijkstra's Algorithm: Apply Dijkstra's algorithm starting from a specific vertex.

Important & Repeated Questions: Minimum spanning trees (Prim's vs. Kruskal's) and shortest path algorithms (Floyd's and Dijkstra's) are highly emphasized.

3. Linked Lists

- Linked List Operations: Merge two linked lists at an intersection point using stacks.
- Circular Doubly Linked List: Delete the third-last element and swap the first and last nodes without swapping data.
- Queue using Linked List: Implement a queue with a circular linked list and perform enqueue and dequeue in O(1) time.

- Binary Search in Linked List: Discuss if binary search can be applied to a doubly linked list sorted by ID, and give an insertion algorithm to keep the list sorted.

Important & Repeated Questions: Implementing various list operations, especially merging and polynomial addition, and linked list manipulations are common.

4. Queues

- Queue Implementation with Stack: Implement queue operations using two stacks and demonstrate the process.
- Queue and Stack Operations: Given a sequence of letters and symbols, simulate PUSH/POP (stack) and ENQUEUE/DEQUEUE (queue) operations.
- Priority Queue Design: Design a data structure to support operations for an application needing fast deletions of minimum and maximum values.

Important & Repeated Questions: Queue implementations using other data structures (like stacks) and advanced queue handling are notably repeated.