Data Structures and Algorithms Interview Questions

---

1. Arrays

- **Question:** *What is the time complexity of accessing an element in an array?*
    - **Answer: O(1)**
- **Question:** *How would you reverse an array?*
    - **Answer:**
        - Iterate through the array and swap elements from both ends towards the middle.
- **Question:** *What is the difference between a static array and a dynamic array?*
    - **Answer:**
        - A **static array** has a fixed size, while a **dynamic array** can resize automatically as needed.

---

2. Linked Lists

- **Question:** *What is a singly linked list?*
    - **Answer:**
        - A **singly linked list** is a collection of nodes where each node has a data field and a pointer to the next node.
- **Question:** *What is the time complexity of inserting an element in the middle of a linked list?*
    - **Answer:**
        - **O(n)** (because you have to traverse the list to find the insertion point).
- **Question:** *How do you detect a cycle in a linked list?*
    - **Answer:**
        - Use **Floyd's Tortoise and Hare algorithm** (two pointers moving at different speeds).

---

3. Stacks

- **Question:** *What is a stack?*
    - **Answer:**
        - A **stack** is a data structure that follows the **Last-In-First-Out (LIFO)** principle.
- **Question:** *What are the basic operations of a stack?*
    - **Answer:**

        - **Push** (add an element),
        - **Pop** (remove an element),
        - **Peek** (view the top element),
        - **isEmpty** (check if the stack is empty).
- **Question:** *How do you implement a stack using an array or a linked list?*
    - **Answer:**
        - Using an **array**, you can use an index pointer to represent the top of the stack.
        - With a **linked list**, each new element is inserted at the head.

---

4. Queues

- **Question:** *What is a queue?*
    - **Answer:**
        - A **queue** is a data structure that follows the **First-In-First-Out (FIFO)** principle.
- **Question:** *How do you implement a queue using two stacks?*
    - **Answer:**
        - Use one stack for **enqueueing** and another for **dequeueing**. When the dequeue stack is empty, transfer elements from the enqueue stack.
- **Question:** *What is the time complexity of enqueue and dequeue operations in a queue?*
    - **Answer:**
        - Both operations have a time complexity of **O(1)** in a simple queue.

---

5. Trees

- **Question:** *What is a binary tree?*
    - **Answer:**
        - A **binary tree** is a tree data structure where each node has at most two children, usually referred to as the left and right child.
- **Question:** *What is the difference between a binary tree and a binary search tree (BST)?*
    - **Answer:**
        - In a **binary tree**, there are no specific rules for the arrangement of nodes.
        - In a **binary search tree**, the left child must be **less** than the parent node, and the right child must be **greater**.
- **Question:** *How do you perform a level-order traversal of a binary tree?*
    - **Answer:**
        - Use a **queue** to store nodes at each level and process them in a **FIFO** order.

---

6. Heaps

- **Question:** *What is a heap?*
    - **Answer:**
        - A **heap** is a complete binary tree where the parent node is greater (**max heap**) or smaller (**min heap**) than its children.
- **Question:** *What are the applications of heaps?*
    - **Answer:**
        - Heaps are commonly used in:
            - **Priority queues**,
            - **Heap sort**,
            - Finding the **kth largest or smallest element** in an array.

---

7. Hashing

- **Question:** *What is a hash table?*
    - **Answer:**
        - A **hash table** is a data structure that stores key-value pairs, where the **key** is hashed to produce an index to access the **value**.
- **Question:** *What are common issues with hash tables?*
    - **Answer:**
        - **Collisions** (when two keys hash to the same index), which can be resolved using:
            - **Chaining** (using linked lists for each index),
            - **Open addressing** (finding another available index).

---

8. Graphs

- **Question:** *What is a graph?*
    - **Answer:**
        - A **graph** is a collection of nodes (**vertices**) connected by edges. It can be **directed** or **undirected**.
- **Question:** *How do you represent a graph in memory?*
    - **Answer:**
        - A graph can be represented using:
            - **Adjacency matrix** (a 2D array where each element indicates an edge),

        - **Adjacency list** (a list of lists where each node points to its neighbors).
- **Question:** *What is the difference between a depth-first search (DFS) and breadth-first search (BFS)?*
    - **Answer:**
        - **DFS** explores as far as possible along each branch before backtracking.
        - **BFS** explores the neighbor nodes level by level.

---

9. Sorting Algorithms

- **Question:** *What is the time complexity of QuickSort?*
    - **Answer:**
        - **Average case: O(n log n)**,
        - **Worst case: O(n²)**
- **Question:** *How does MergeSort work?*
    - **Answer:**
        - **MergeSort** divides the array into halves, sorts each half, and then merges them back together.
- **Question:** *What is the time complexity of BubbleSort?*
    - **Answer:**
        - **O(n²)**

---

10. Searching Algorithms

- **Question:** *What is the time complexity of binary search?*
    - **Answer:**
        - **O(log n)**
- **Question:** *What is the difference between linear search and binary search?*
    - **Answer:**
        - **Linear search** checks each element one by one, whereas **binary search** repeatedly divides the search interval in half.