Ashwini Vadkar

ADS

Day 1 – 25-03-2025

Homework: Solve the following Interview questions

Algorithms

1. **What is an algorithm?**

   An algorithm is a step-by-step procedure or a set of rules for solving a specific problem or performing a computation.

2. **Why is algorithm analysis important?**

Algorithm analysis helps determine the efficiency of an algorithm in terms of time and space complexity, ensuring optimal performance for large inputs.

3. **What are the key criteria for analyzing an algorithm?**

The key criteria for analyzing an algorithm include time complexity, space complexity, correctness, scalability, and optimality.

4. **What are the different approaches to developing algorithms?**

The different approaches to developing algorithms include the brute force method, divide and conquer, dynamic programming, greedy approach, and backtracking.

5. **What are the characteristics of a good algorithm?**

A good algorithm should be correct, efficient, simple, well-defined, finite, and should provide the expected output for all valid inputs.

Data Structures

1. **What are different types of data structures?**

2. The different types of data structures include linear data structures (arrays, linked lists, stacks, and queues) and non-linear data structures (trees, graphs, and hash tables).

3. **What is the difference between an array and a linked list?**

An array is a fixed-size, contiguous memory data structure where elements are stored sequentially, whereas a linked list consists of nodes where each node contains data and a reference to the next node, allowing dynamic memory allocation.

4. **How does a stack work? Provide a real-time example.**

A stack follows the Last In, First Out (LIFO) principle, where the last element added is the first to be removed. A real-time example is the undo-redo feature in text editors, where the most recent action is undone first.

5. **What are the operations on a queue? Explain different types of queues.**

The main operations on a queue are **enqueue** (inserting an element), **dequeue** (removing an element), **peek** (viewing the front element), and **isEmpty** (checking if the queue is empty).

a. **Simple Queue**: Follows First In, First Out (FIFO) order.
b. **Circular Queue**: The last position is connected to the first position, making efficient use of space.
c. **Deque (Double-Ended Queue)**: Allows insertion and deletion from both ends.
d. **Priority Queue**: Elements are dequeued based on priority instead of arrival order.

6. **What is a graph? Explain different types of graphs.**

A graph is a non-linear data structure consisting of **vertices (nodes)** and **edges (connections)** that represent relationships between entities.

a. **Directed Graph (Digraph)**: Edges have a direction (A → B).
b. **Undirected Graph**: Edges do not have a direction (A — B).
c. **Weighted Graph**: Edges have weights or costs assigned to them.

d. **Unweighted Graph**: All edges have equal weight or no weight.
e. **Cyclic Graph**: Contains at least one cycle (a path where a node is revisited).
f. **Acyclic Graph**: Has no cycles, such as a Tree.

Recursion

1. **What is recursion, and how does it work?**

2. Recursion is a programming technique where a function calls itself to solve a problem by breaking it down into smaller subproblems until a base case is reached.

3. **Why is recursion used in programming?**

Recursion is used to simplify complex problems, especially those that have a natural recursive structure like tree traversal, backtracking, and divide-and-conquer algorithms.

4. **What are the advantages and disadvantages of recursion?**

Advantages:

a. Simplifies code for problems like tree traversal and backtracking.
b. Reduces the need for explicit loops, making code more readable.

Disadvantages:

c. Higher memory usage due to function call stack.
d. Risk of stack overflow if the base case is not well defined.
e. Usually less efficient than iteration due to repeated function calls.

5. **Differentiate between recursion and iteration.**
   a. **Recursion:** A function calls itself, and termination depends on a base case. Uses the call stack and can lead to higher memory usage.

b. **Iteration:** Uses loops (for, while) to repeat execution without extra memory overhead. More efficient in terms of execution speed and memory.

6. **What are base cases in recursion, and why are they important?**

A base case is a condition that stops the recursion by returning a value without making further recursive calls. Base cases are crucial to prevent infinite recursion and stack overflow errors.

7. **Which problems can be solved using recursion?**

Recursion is useful for problems like:

a. Factorial calculation
b. Fibonacci series
c. Tower of Hanoi
d. Tree and graph traversal (DFS)
e. Sorting algorithms like QuickSort and MergeSort
f. Backtracking problems like N-Queens and Sudoku solver