Q: What is an algorithm?

A: An algorithm is a finite set of step-by-step instructions designed to solve a specific problem or perform a required computation efficiently.

Q: What is time complexity?

A: Time complexity expresses how the running time of an algorithm grows relative to the input size (n). It is used to compare algorithm performance independent of hardware.

Q: What is space complexity?

A: Space complexity measures the amount of memory an algorithm uses relative to input size. It includes input space, auxiliary space, and recursion stack.

Q: Explain Big-O notation.

A: Big-O notation represents the upper bound of an algorithm's time or space requirement. It describes the worst-case performance, such as O(n), O(n log n), or O(n^2).

Q: Difference between array and linked list?

A: Arrays support random access but have fixed size. Linked lists support dynamic memory allocation but accessing elements takes O(n).

Q: What is a stack?

A: A stack is a LIFO (Last In First Out) data structure used for function calls, undo operations, and expression evaluation.

Q: What is a queue?

A: A queue is a FIFO (First In First Out) structure used in scheduling, buffering, and BFS traversal.

Q: What is a binary tree?

A: A binary tree is a hierarchical data structure where each node has at most two children: left and right.

Q: What is a binary search tree?

A: A BST is a binary tree where left child < root < right child. It allows O(log n) search on average.

Q: What is a hash table?

A: A hash table stores key–value pairs using a hash function to compute index. It offers average O(1) insert, delete, and search.

Q: Explain hashing and collision resolution.

A: Hashing maps keys to indexes. Collisions occur when two keys map to the same index. Common resolution techniques include chaining and open addressing.

Q: What is a heap?

A: A heap is a complete binary tree used for implementing priority queues. It supports efficient extraction of min/max.

Q: Difference between min-heap and max-heap?

A: In a min-heap, the smallest element is at the root; in a max-heap, the largest is at the root.

Q: Explain graph data structure.

A: A graph consists of vertices and edges. It can be directed or undirected, weighted or unweighted, cyclic or acyclic.

Q: Difference between BFS and DFS?

A: BFS uses a queue and explores level by level. DFS uses a stack/recursion and explores deep paths first.

Q: What is dynamic programming?

A: Dynamic programming solves problems by breaking them into overlapping subproblems and storing solutions to avoid recomputation.

Q: What is memoization?

A: Memoization is a top-down DP technique where results of recursive calls are cached.

Q: What is tabulation?

A: Tabulation is a bottom-up DP approach where solutions to smaller subproblems are filled in a table.

Q: Explain greedy algorithm.

A: Greedy algorithms make the optimal choice at each step without reconsideration, used in Kruskal, Prim, and Dijkstra algorithms.

Q: What is divide and conquer?

A: An approach where a problem is divided into smaller subproblems, solved independently, and combined. Examples: merge sort, quicksort.

Q: Explain recursion.

A: Recursion occurs when a function calls itself. It must have a base case to stop infinite calls.

Q: What is tail recursion?

A: Tail recursion is when the recursive call is the last operation in the function, allowing compiler optimization.

Q: What is backtracking?

A: Backtracking tries all possible solutions and removes invalid ones using recursion. Examples: N-Queens, permutations.

Q: What is a trie?

A: A trie is a prefix tree used for efficient retrieval of strings, autocomplete, and dictionary storage.

Q: Explain AVL tree.

A: An AVL tree is a self-balancing BST where height difference (balance factor) is maintained between -1 and +1.

Q: What is red-black tree?

A: A red-black tree is a balanced BST ensuring longest path is no more than twice the shortest. Used in maps and sets.

Q: What is topological sorting?

A: Topological sort orders vertices of a directed acyclic graph based on dependencies.

Q: Explain shortest path problem.

A: Shortest path algorithms find minimum cost paths in graphs. Dijkstra works for positive weights; Bellman–Ford supports negative weights.

Q: Explain Dijkstra’s algorithm.

A: Dijkstra finds shortest paths from a source to all nodes using a priority queue and greedy relaxation.

Q: Explain Floyd–Warshall algorithm.

A: Floyd–Warshall computes shortest paths between all pairs of vertices using dynamic programming.

Q: What is union–find?

A: A disjoint set data structure used for grouping elements and detecting cycles. Supports union and find operations.

Q: Explain cycle detection in graph.

A: DFS is used for cycle detection. Back edges indicate cycles in directed graphs; parent-tracking helps in undirected graphs.

Q: What is DAG?

A: A Directed Acyclic Graph is a directed graph with no cycles. Used in scheduling and dependency resolution.

Q: Explain KMP string algorithm.

A: KMP performs substring search in O(n + m) by precomputing longest prefix-suffix (LPS) table.

Q: Explain Rabin–Karp algorithm.

A: Rabin–Karp uses rolling hash to search patterns efficiently, especially for multiple pattern matching.

Q: What is a segment tree?

A: A segment tree is a binary tree used for range queries like sum, min, max with O(log n) updates.

Q: What is a Fenwick tree?

A: Fenwick tree (Binary Indexed Tree) supports prefix sums and updates in O(log n).

Q: Explain quicksort.

A: Quicksort uses a pivot to partition the array and recursively sort halves. Average time: O(n log n).

Q: Explain mergesort.

A: Mergesort divides the array, sorts subarrays, and merges them. Time: O(n log n), stable sorting.

Q: Explain heapsort.

A: Heapsort builds a heap and repeatedly extracts the root. Time: O(n log n), not stable.

Q: Stable vs unstable sorting?

A: Stable sorting preserves order of equal elements (e.g., merge sort). Unstable sorting does not (e.g., quicksort).

Q: Explain binary search.

A: Binary search divides the search interval in half. Works on sorted arrays. Time: O(log n).

Q: What is interpolation search?

A: An improved binary search for uniformly distributed data, using probing formula to predict position.

Q: Explain two-pointer technique.

A: Two pointers move through an array solving problems like pair sums, partitions, and sliding windows.

Q: What is sliding window technique?

A: Sliding window solves subarray problems by maintaining a dynamic range. Common in max/min subarray and frequency problems.

Q: Difference between BFS and level-order?

A: Both are same for trees. In graphs, BFS may revisit levels due to cycles but uses visited[] to handle.

Q: Explain strongly connected components.

A: SCCs are groups of vertices in directed graphs where each is reachable from another. Found using Kosaraju or Tarjan algorithm.

Q: Explain Kadane's algorithm.

A: Kadane computes maximum subarray sum in O(n) by keeping track of current and global maxima.

Q: Explain LCS algorithm.

A: LCS finds the longest common subsequence using DP table in O(n*m).

Q: Explain palindrome checking.

A: Use two pointers checking characters from both ends until they meet.

Q: What is bit manipulation?

A: Using bitwise operators to optimize problems such as subsets, masks, parity, or power of two checks.

Q: What is monotonic stack?

A: A stack where elements are kept in sorted order. Used in nearest greater/smaller problems.

Q: What is monotonic queue?

A: A queue that maintains elements in sorted order, used in sliding window max/min in O(n).

Q: Explain graph coloring.

A: Assigning colors to vertices such that adjacent vertices have different colors. Greedy or backtracking is used.