Algorithm/Data Structure	Time Complexity (Big O)
Mergesort	O(n log n)
Quicksort	O(n^2) - worst case, O(n log n) - average case
Heapsort	O(n log n)
Trie	O(m), where m is the length of the key
Bellman-Ford	O(V * E), where V is the number of vertices and E is the number of edges
Dijkstra	O((V + E) log V) - using binary heap or Fibonacci heap
Matrix Chain Multiplication (MCM)	O(2^n) - naive approach(recursive) , O(n^3) - optimized approach(memoization)
0-1 Knapsack	DP: O(n*W) n=no.of items W=capacity Branch and Bound: O(2^n) Backtracking: O(2^n) Greedy: O(nlogn)
Topological Sort	O(V+E)
Activity Selection (Greedy method)	O(nlogn); O(nlogn) for sorting and O(n) for greedy selection
LCS	O(m*n) if DP applied;m,n are string lengths Exponential if brute force
Job Sequencing	O(n^2) ; nested loops used to iterate through each job and find the latest available time slot
Travelling Salesman	O(n!) (Branch and Bound)

1. In a min-heap, what is the relationship between a parent node and its children?

A parent is less than both children

2. Which of the following is a valid application of a heap data structure?

   Priority queue

3. To build max-heap from an unordered array, it takes-

   Linear time O(n)

4. Heap deletion-

   O(logn)

5. Heap insertion-

   O(logn)

6. Heap sort-

   O(nlogn)

7. What traversal over trie gives the lexicographical sorting of the set of the strings?

   Inorder

8. What does the Single Source Shortest Path (SSSP) problem aim to find?

   Shortest path from one source to all other vertices.

9. Which algorithm is commonly used for solving the Single Source Shortest Path problem on graphs with non-negative edge weights?

   Dijkstra's.

10. What type of graphs can Dijkstra's algorithm handle efficiently?

    Weighted graphs with non-negative weights.

11. Which data structure is essential for implementing Dijkstra's algorithm efficiently?

    Priority Queue.

12. What is the primary limitation of Dijkstra's algorithm?

    It cannot handle graphs with negative edge weights.

13. In Bellman-Ford algorithm, how many iterations are required to guarantee the shortest paths in a graph with V vertices?

    V - 1.

14. Which algorithm is suitable for finding the shortest paths in a graph with negative edge weights?

    Bellman-Ford.

15. **What is the time complexity of the Bellman-Ford algorithm?**

    O(V*E).

16. In the context of single-source shortest path algorithms, what does "relaxation" refer to?

    Updating the distance estimate to a vertex.

17. What is a heap in the context of data structures?

    A binary tree with a specific order property

18. In a max-heap, what is the relationship between a parent node and its children?

    Parent is greater than both children

19. Which operation ensures the heap property is maintained in a max-heap after an insertion?

    Bottom Up

20. What is the worst-case time complexity of Heapsort?

    O(n log n)

21. In Heapsort, what is the role of the max-heap or min-heap?

    To repeatedly extract the minimum or maximum element

22. Which heap operation is used in the extraction step of Heapsort?

    Delete Max (or Min)

23. What is the primary advantage of Heapsort over other sorting algorithms, such as Quicksort?

    In-place sorting

24. What is the LCS of the following string: X = ABCBDAB Y= BDCABA **BCAB**
25. Print LCS**- O(m*n)**
26. What is the fourth step of dynamic programming- **Construct an optimal solution from computed information**
27. The bottom-up approach of dp is also called- **Tabulation**
28. The top-down(recursive) approach is used with- **memoization**
29. characteristics that a problem must have to apply dp- **Optimal substructure property and overlapping subproblems property**
30. Backtracing traverses the state space tree by- **DFS**
31. Branch and bound traverses the tree by - **DFS or BFS**
32. Backtracking is used to solve- **Decision problems(problems of a combinatorial nature)**
33. Which is more efficient, backtracking or branch and bound? - **Backtracking**
34. Complexity of back tracking- **O(P^N)** Here, P= number of probabilities associated with each state, for 0-1 Knapsack P=2(taken or not taken) N= number of states or depth of the recursive tree (no. of items in case of knapsack)
35. Complexity of backtracking to solve -N Queen problem: **O(N!)**

    Subset Sum: **O(2^n)**

    Graph coloring: **O(m^n); m= no. of colors, n=no of vertices**

36. Branch and bound is used to solve- **Optimization problems**
37. why is Branch and Bound inefficient? **The entire state-space tree is searched in order to find the optimal solution**
38. LIFO(stack) branch and bound- **DFS**
39. FIFO(queue) branch and bound- **BFS**
40. NP-hard problems can be solved by- **Branch and Bound**
41. k-ary tree means- **each node has at most k children**

Operation	Time Complexity	Description
Insertion	O(L)	Inserting a key of length L into the Trie.
Search	O(L)	Searching for a key of length Lin the Trie.
Deletion	O(L)	Deleting a key of length L from the Trie.
Prefix Search	O(P)	Finding all keys with a given prefix of length P.
Counting Words with Prefix	O(P)	Counting the number of words with a given prefix of length P.
Space Complexity	O(N)	The overall space complexity of the Trie, where N is the total number of characters stored.
Lexicographical Order	O(N+M)	Where N is the total number of nodes in the TRIE and M is the total length of all strings

41. The number of scalar multiplications required to multiply two matrices A (of dimensions m x n) and B (of dimensions n x p) is given by the formula:

Number of multiplications=m×n×p

42. Worst case time complexity of quicksort is $O(n^2)$ - **When the array is already sorted, and if Quicksort consistently chooses one of the ends (e.g., always selecting the first or last element) as the pivot**

43. Sorting algorithm stability- **In a stable sorting algorithm, if two elements have equal keys (values), and they appear in a particular order in the input, they will also appear in the same order in the sorted output.**

44.

Cost matrix

$$\begin{bmatrix} \inf & 20 & 30 & 10 & 11 \\ 15 & \inf & 16 & 4 & 2 \\ 3 & 5 & \inf & 2 & 4 \\ 19 & 6 & 18 & \inf & 3 \\ 16 & 4 & 7 & 16 & \inf \end{bmatrix}$$

How to Compute the Reduced Matrix:

The reduced matrix is obtained after the row and column reductions.

The reduced cost for each cell is calculated by subtracting first the minimum row cost and then the minimum column cost from the original cost in that cell.

Cost (1) = 25

$$\begin{bmatrix} \inf & 10 & 17 & 0 & 1 \\ 12 & \inf & 11 & 2 & 0 \\ 0 & 3 & \inf & 0 & 2 \\ 15 & 3 & 12 & \inf & 0 \\ 11 & 0 & 0 & 12 & \inf \end{bmatrix}$$

45. The drawback of using the earliest start time in activity selection problem - **it may not always lead to the selection of the maximum-sized set of non-overlapping activities**

46**. Earliest finish time- always gives optimal solution**

47. **Job Sequence -**

i.Sort all jobs in decreasing order of profit.

ii. Iterate on jobs in decreasing order of profit. For each job, do the following :

iii.Find a time slot i, such that slot is empty and i <= deadline and i is greatest. Put the job in this slot and mark this slot filled. If no such i exists, then ignore the job.

48. Max heapify- **O(logn)**