DAA Short Questions

Q1: Algorithm vs Data Structure

Ans: Data Structure is about organising and managing data effectively such that we can perform specific operation efficiently, while Algorithm is a step-by-step procedure to be followed to reach the desired output.

Q2: Linear vs Non-linear search ?

Ans:

1.	In a linear data structure, data elements are arranged in a linear order where each and every elements are attached to its previous and next adjacent.	In a non-linear data structure, data elements are attached in hierarchically manner.
2.	In linear data structure, single level is involved.	Whereas in non-linear data structure, multiple levels are involved.
3.	Its implementation is easy in comparison to non-linear data structure.	While its implementation is complex in comparison to linear data structure.
4.	In linear data structure, data elements can be traversed in a single run only.	While in non-linear data structure, data elements can't be traversed in a single run only.
5.	In a linear data structure, memory is not utilized in an efficient way.	While in a non-linear data structure, memory is utilized in an efficient way.

Q3: Recursive Algorithm ?

Ans: A recursive algorithm is an **algorithm which calls itself with "smaller (or simpler)" input values**, and which obtains the result for the current input by applying simple operations to the returned value for the smaller (or simpler) input.

Q4: Inplace sorting ?

Ans: **A sort algorithm in which the sorted items occupy the same storage as the original ones**. These algorithms may use o(n) additional memory for bookkeeping, but at most a constant number of items are kept in auxiliary memory at any time. Also known as sort in place.

Q5: Stable Sorting ?

Ans: A sorting algorithm is said to be stable if **two objects with equal keys appear in the same order in sorted output** as they appear in the input array to be sorted.

Q6: Big oh (*O*) notation ?

Ans: An asymptotic notation that measures the performance of an algorithm by simply providing the order of growth of the function.

Q7: Problem of optimality ?

Ans: A problem is said to satisfy the principle of optimality of the sub-solutions of an optimal solution of the problem are themselves optimal solutions for the sub-problem.

e.g: The Shortest path problem satisfies the principle of optimality.

Q8: Criteria to improve the effectiveness of algorithm ?

Ans:

1. Zero or more input values.
2. One or more output values.
3. Clear and unambiguous instructions.
4. Atomic steps that take constant time.
5. No infinite sequence of steps (help, the halting problem)
6. Feasible with specified computational device.

Q9: Complexity of Prims & kruskal Algo. ?

Ans:

The time complexity of the **Prim's Algorithm** is **O ((V + E) l o g V)** because each vertex is inserted in the priority queue only once and insertion in priority queue take logarithmic time.

Kruskal's algorithm's time complexity is **O(E log V)**, V being the number of vertices.

Q10: Greedy Method vs Dynamic Programming ?

Ans:

Dynamic Programming	Greedy Method
1. Dynamic Programming is used to obtain the optimal solution.	1. Greedy Method is also used to get the optimal solution.
2. In Dynamic Programming, we choose at each step, but the choice may depend on the solution to sub-problems.	2. In a greedy Algorithm, we make whatever choice seems best at the moment and then solve the sub-problems arising after the choice is made.
3. Less efficient as compared to a greedy approach	3. More efficient as compared to a greedy approach
4. Example: 0/1 Knapsack	4. Example: Fractional Knapsack
5. It is guaranteed that Dynamic Programming will generate an optimal solution using Principle of Optimality.	5. In Greedy Method, there is no such guarantee of getting Optimal Solution.

Q11: Minimum Spanning Tree ?

Ans: The Minimum Spanning Tree is the one whose cumulative edge weights have the smallest value, however. Think of it as the least cost path that goes through the entire graph and touches every vertex.

Q12: Divide And Conquer ?

Ans: Divide and conquer is a recursive problem. Solving approach which break a problem into smaller sub problems, and finally combines the solution to the subproblem to solve the original problem.

This method usually allows us to reduce the time complexity to a large extent.

Q13: Dense & Sparse Graph ?

Ans:

Dense Graph:	**Sparse Graph:**
Dense graph is a graph in which the number of edges is close to the maximal number of edges q	Sparse graph is a graph in which the number of edges is close to the minimal number of edges

Q14: BFS vs DFS ?

Ans:

Breadth First Search:	**Depth First Search:**
Breadth first search is a **graph traversal algorithm that starts traversing the graph from root node and explores all the neighbouring nodes**. Then, it selects the nearest node and explore all the unexplored nodes. The algorithm follows the same process for each of the nearest node until it finds the goal.	The **Depth First Search** (DFS) is an algorithm for traversing or searching tree or graph data structures which uses the idea of backtracking. It explores all the nodes by going forward if possible or uses backtracking. **Note:** It can be implemented using a *stack.*

Q15: Digraph ?

Ans: A directed graph (or digraph) is a set of vertices and a collection of directed edges that each connects an ordered pair of vertices. We say that a directed edge points from the first vertex in the pair and points to the second vertex in the pair

Q16: Prims vs Kruskal ?

Ans:

Prims:	**Kruskal:**
• Prim's Algorithm is faster for dense graphs. • Prim's algorithm works by choosing the adjacent vertices from the selected set of vertices • The time complexity of Prim's algorithm is O(V2)	• Kruskal's Algorithm is faster for sparse graphs. • Kruskal's algorithm selects least weight edges instead of using adjacency list, it organizes the edges by their weights. • Kruskal's algorithm runs in O(log V) time.

Q17:Dijkstra vs Bellman Ford ?

Ans:

Dijkstra:	**Bellman:**
• Dijkstra's Algorithm doesn't work when there is negative weight edge. • It is less time consuming. • Greedy approach is taken to implement the algorithm.	• Bellman Ford's Algorithm works when there is negative weight edge, it also detects the negative weight cycle. • It is more time consuming than Dijkstra's algorithm. • Dynamic Programming approach is taken to implement the algorithm.

Q18: String Matching with finite automata ?

Ans: The string-matching automaton is a very useful tool which is used in string matching algorithm. It **examines every character in the text exactly once** and reports all the valid shifts in O (n) time.

Q19: What is KMP Algo. ?

Ans: The **KMP algorithm** is an efficient string matching algorithm. It is a linear time algorithm for the string matching problem. It is used to find a pattern in the text.

Q20:What is NAÏVE string matching Algorithm ?

Ans: A naïve algorithm is an algo. That behaves in a very simple way.

For example: A naïve algo. For sorting numbers scan all numbers to find the smallest one, puts it aside, and so on. But it is not an efficient algorithm.

Q21: What is Divide and Conquer approach ?

Ans: In divide and conquer algorithm recursively breaks down a problem into two or more sub-problems of the same or related type, then each problem is solved independently.

Q22: Define the term complexity of an algorithm ?

Ans: Complexity of an algorithm is a measure of the amount of time and/or space by an algorithm for an input of a given size (n).

Q23: What do you know about Floyd-Warshall algorithm ?

Ans: The Floyd-warshall algorithm is an algorithm for finding shortest paths in a directed weighted graph with positive or negative edge weights.

Q24: What are different types of notation used for representation of complexities?

Ans: Big-O Notation (O) – Describes worst case scenario.

Omega Notation (Ω) – Describes best case scenario.

Theta Notation (θ) – Represents the average complexity of an algorithm.

Q25: Shortly describe the binary search ?

Ans: Binary search is **an efficient algorithm for finding a value from a sorted array**. Binary search compares the target value to the middle element of the array, until the value is found or the array is completely searched.

Q26: What will be the complexity of two N-times nested loops ?

Ans: The Complexity of two N-times nested loops is $O(n^2)$.

Q27: What is recursion ?

Ans: The process in which a function calls itself directly or indirectly is called recursion and the corresponding function is called as recursive function. Using recursive algorithm, certain problems can be solved quite easily.

Q28: What are the different complexities according to which we analyze any algorithm?

Ans: The complexity of an algorithm computes the amount of time and spaces required by an algorithm for an input of size (n).

The complexity of an algorithm can be divided into two types. The **time complexity** and the **space complexity.**

Q29: What is Merge Sort ? and is insertion sort better than the merge sort ?

Ans: Merge Sort is a Divide and Conquer algorithm. It divides the input array into two halves, calls itself for the two halves, and then merges the two sorted halves.

Insertion Sort is preferred for fewer elements. It becomes fast when data is already sorted or nearly sorted because it skips the sorted values. Efficiency: Considering average time complexity of both algorithm we can say that Merge Sort is efficient in terms of time and Insertion Sort is efficient in terms of space.

Q30: What is NP hard and NP complex problems ?
Ans: A problem is in the class NPC if it is in NP and is as hard as any problem in NP. A problem is NP-hard if all problems in NP are **polynomial time reducible to it**, even though it may not be in NP itself. If a polynomial time algorithm exists for any of these problems, all problems in NP would be polynomial time solvable.

Q31: What is Feasible and optimal solution ?

Ans: A **feasible solution** is a set of values for the decision variables that satisfies all of the constraints in an optimization problem.

An optimal solution is a **feasible solution where the objective function reaches its maximum (or minimum) value** – for example, the most profit or the least cost.

Q32: Write down three cases of Master Method for solving recurrences. ?
Ans:

There are 3 cases for the master theorem:

Case 1: $\mathbf{n^{\log_b a} > f(n)}$ Time Complexity: $\mathbf{\theta(n^{\log_b a})}$
Case 2: $\mathbf{n^{\log_b a} = f(n)}$ Time Complexity: $\mathbf{\theta(n^{\log_b a} \log n)}$
Case 3: $\mathbf{n^{\log_b a} < f(n)}$ Time Complexity: $\mathbf{\theta(f(n))}$

Q33: Which sorting algorithm will perform better if array is already sorted?
Ans: **Insertion sort** runs much more efficiently if the array is already sorted or "close to sorted." Insertion sort performs O(n2) swaps in the average and worst case.

Q34: What is the Time complexity of Prim's Algorithm ?
Ans: The time complexity of the **Prim's Algorithm** is **O ((V + E) l o g V)** because each vertex is inserted in the priority queue only once and insertion in priority queue take logarithmic time.

Q35: How Problems are solved using divide and conquer approach ?

Ans: A divide-and-conquer algorithm recursively breaks down a problem into two or more **sub-problems** of the same or related type, until these become simple enough to be solved directly. The solutions to the sub-problems are then combined to give a solution to the original problem.

Q36: Define all pair shortest path problem ?

Ans: The **all pair shortest path algorithm** is also known as Floyd-Warshall **algorithm**.

The Floyd-warshall algorithm is an algorithm for finding shortest paths in a directed weighted graph with positive or negative edge weights.

Q37: Write an algorithm using Recursive function to find the sum of n numbers ?

Ans:

```
int recurSum(int n)
{
   if (n <= 1)
      return n;
   return n + recurSum(n - 1);
}
```

Q38: Differentiate between P and NP Problem ?

Ans:

P PROBLEMS	**NP PROBLEMS**
P problems are set of problems which can be solved in polynomial time by deterministic algorithms.	NP problems are the problems which can be solved in non-deterministic polynomial time.
The problem belongs to class P if it's easy to find a solution for the problem.	The problem belongs to NP, if it's easy to check a solution that may have been very tedious to find.
P Problems can be solved and verified in polynomial time.	Solution to NP problems cannot be obtained in polynomial time, but if the solution is given, it can be verified in polynomial time.
P problems are subset of NP problems.	NP problems are superset of P problems.
It is not known whether P=NP. However, many problems are known in NP with the	If P≠NP, there are problems in NP that are neither in P nor in NP-Complete.

property that if they belong to P, then it can be proved that P=NP.	
All P problems are deterministic in nature.	All the NP problems are non-deterministic in nature.
Selection sort, linear search	TSP, Knapsack problem.

Q39: What is non-deterministic computer ?

Ans: In computer programming, a **nondeterministic algorithm** is an algorithm that, even for the same input, can exhibit different behaviors on different runs, as opposed to a deterministic algorithm.

Q40: What is Average Case efficiency ?

Ans: Average Case Efficiency - **average comparisons between minimum no. of comparisons and maximum no.** is called average case efficiency.