**Huffman Coding:** Huffman coding is a compression algorithm that assigns shorter codes to more frequently occurring characters and longer codes to less frequent ones. This results in efficient data compression by reducing the overall size of the data.

**=============================UNIT-01===================================**

**Stack:** A stack is an ordered list in which all Insertions and deletions are made at one end, called the top. Stack follows the LIFO operations.

A real-world example of a stack is a stack of plates in a cafeteria.

**Queue:** A queue is an ordered list in which all insertions take place at one end, the rear, whereas all deletions take place at the other end, the front. It follows the FIFO operations

A real-world example of a queue is a line at a checkout counter in a store.

**Tree:** It is a hierarchical structure as elements in a Tree are arranged in multiple levels. In the Tree data structure, the topmost node is known as a root node. Each node contains some data, and data can be of any type.

**Binary Tree:** A tree whose elements can have at most 2 children is called a binary tree.

**Binary Search Tree:** It is a data structure to maintain a sorted set of elements, allowing efficient insertion, deletion, and lookup operations. Values which are greater than parent node will be on right side and values which are less than the parent node will be on left side.

**Graph:** A graph is a data structure consisting of nodes (vertices) and edges connecting pairs of nodes. It is used to represent relationships between objects, such as networks, maps, and social connections.

• An undirected graph is a type of graph where the edges have no specified direction assigned to them.

• A weighted graph or a network is a graph in which a number (the weight) is assigned to each edge.

**Computational Model:** Computational modeling is the use of computers to simulate and study complex systems using mathematics, physics and computer science.

**Algorithm:** An algorithm is a well-defined sequence of steps or instructions designed to perform a specific task or solve a particular problem.

**Program**: A specific implementation of an algorithm in code that can be executed by a computer. It includes syntax and structure specific to a programming language.

**Space Complexity:** The amount of memory required by an algorithm to run to completion.

• Fixed part: The size required to store certain data/variables, that is independent of the size

of the problem: Such as int a (2 bytes), float b (4 bytes) etc.

• Variable part: Space needed by variables, whose size is dependent on the size of the

problem: Dynamic array a[ ].

**The Random-Access Machine** is a simplified model of a modern computer used in theoretical

computer science to analyze algorithms.

**Random Access Memory** is the primary memory device in which the CPU stores data in a real-life computer.

**Time complexity** is a measure of how long an algorithm takes to run as the size of the input increases.

1) **O (Big – Oh notation):** Big-O notation represents the upper bound of the running time of an algorithm.

Therefore, it gives the worst-case complexity of an algorithm.

2) **Ω (Big – Omega notation):** Omega notation represents the lower bound of the running time of an algorithm.

Thus, it provides the best case complexity of an algorithm.

3) **Θ (Theta notation):** Theta notation encloses the function from above and below. Since it represents the upper and the lower bound of the running time of an algorithm, it is used for analyzing the average-case complexity of an algorithm

**Recursion:** 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. Examples of such problems are Towers of Hanoi (TOH), Inorder/Preorder/Postorder Tree Traversals, DFS of Graph, etc.

A real-world example of a recursive function is calculating the factorial of a number.

**Use of Recursion:** a recursive function can visit each node or element by calling itself on the child nodes or the next element in the list.

**Divide-and-Conquer Strategy:**

This technique can be divided into the following three parts:

1. Divide: This involves dividing the problem into smaller sub-problems.

2. Conquer: Solve sub-problems by calling recursively until solved.

3. Combine: Combine the sub-problems to get the final solution of the whole problem.

**Greedy Algorithm:** A greedy algorithm is an approach for solving a problem by selecting the best option available at the moment without considering the long-term consequences. It doesn't worry whether the current best result will bring the overall optimal result.

The steps to define a greedy algorithm are:

1. Define the problem: Clearly state the problem to be solved and the objective to be optimized.

2. Identify the greedy choice: Determine the locally optimal choice at each step based on the current state.

3. Make the greedy choice: Select the greedy choice and update the current state.

4. Repeat: Continue making greedy choices until a solution is reached.

A real-world **example** of a greedy algorithm is the **Change-Making Problem**: When giving change to a customer, the goal is to use the fewest coins possible.

**Dynamic programming:** Dynamic programming is used where we have problems, which can be divided into similar sub-problems so that their results can be re-used. it avoids re solving a sub program if repeated by storing it's result.

**Top-Down Approach**: Solve a problem by breaking it into smaller sub-problems and solving them recursively. Use memoization to store results of sub-problems to avoid redundant calculations.

**Bottom-Up Approach**: Solve the smallest sub-problems first and use their solutions to build up to the overall solution. This approach iterates from the base cases up to the final problem, typically using a table to store intermediate results.

**Backtracking algorithms**: Backtracking is a method of solving problems by trying solutions, and if a solution doesn’t work, it backtracks to try a different option.

**The Branch and Bound Algorithm:** It works by dividing the problem into smaller subproblems, or branches, and then eliminating certain branches based on bounds on the optimal solution. This process continues until the best solution is found or all branches have been explored.

**Approximation Algorithm:** An approximation algorithm is a method used to find a solution that is close to the best possible answer for a complex problem, especially when finding the exact solution is impractical. It provides a solution that is good enough and usually within a known ratio of the optimal solution.

**Selection Sort Algorithm:** Selection sort is a sorting algorithm that repeatedly finds the smallest (or largest) element from the unsorted portion of the list and swaps it with the first unsorted element. This process continues until the entire list is sorted.

The main drawback of selection sort is its inefficiency for large lists. It has a time complexity of

O(n2) because it requires scanning the remaining unsorted elements to find the minimum value for each position. This makes it slower compared to more efficient algorithms like quicksort or mergesort for large datasets.

**Traveling salesman problem**, an optimization problem in graph theory in which the nodes (cities) of a graph are connected by directed edges (routes), where the weight of an edge indicates the distance between two cities. The problem is to find a path that visits each city once, returns to the starting city, and minimizes the distance traveled.

**=============================UNIT-02===================================**

**Linear Search:** Linear search is a method of finding a target value in a list by checking each element one by one from the beginning to the end until the target is found or the list is exhausted.

**Binary Search:** Binary search is an efficient algorithm used to find the position of a target value within a sorted array. It works by repeatedly dividing in half the portion of the array that could contain the target value, thereby reducing the time complexity to O(log n).

**Comparison Tree:** A comparison tree is a binary tree used to visualize the process of comparisons made by an algorithm, typically sorting algorithms. Each node represents a comparison between two elements, and the tree structure shows how the algorithm's decisions split the problem into sub-problems. The depth of the tree indicates the number of comparisons required.

**Breadth-First Search (BFS)** explores nodes level by level using a queue.

**Depth-First Search (DFS)** explores as deep as possible along each branch before backtracking, using a stack or recursion.

**Merging:** It is a process of combining two sorted lists into a single sorted list.

**Sorting:** Itis the process of arranging items in a specific order, typically in numerical or alphabetical order. The goal is to organize data to make which is easier to search, analyze, and visualize.

**Merge Sort**: Merging is a key operation in the merge sort algorithm, which is a divide-and-conquer algorithm. Merge sort divides the input array into halves, recursively sorts them, and then merges the sorted halves.

**Quick Sort:** Quick sort works on the idea that an element is in the sorted position, if all the elements on the left side of that element are smaller than that and all the elements on the right side of that element are greater than that. Rest of the elements may or may not be sorted. It follows the divide-and-conquer strategy.It is recursive in nature.

**Bubble Sort:** Bubble sort is a simple sorting algorithm that repeatedly steps through a list, compares adjacent elements, and swaps them if they are in the wrong order. This process is repeated until the list is sorted.

**Binary Tree:** A binary tree is a tree data structure in which each node has at most two children, which are referred to as the left child and the right child.

**Full Binary Tree:** A full binary tree is a tree in which every node other than the leaves has two childrens.

**Complete Binary Tree:** A binary tree in which every level (depth), except possibly the deepest, is completely filled.

**Heap:** Heap is a complete binary tree.

Types of heap:

• Max heap: Max heap is a complete binary tree in which every node is having the element

greater than or equal to its child node.

• Min heap: Min heap is a complete binary tree in which every node is having the element

smaller than or equal to its child node.

**=============================UNIT-03===================================**

**knapsack problem:** The knapsack problem is an optimization problem where you have a set of items, each with a weight and a value, and a knapsack with a weight limit. The goal is to maximize the total value of the items in the knapsack without exceeding the weight limit.

**[It is a kind of container which has some maximum capacity so we need to utilize that capacity in such a way so that you can get maximum benefit without violating max capacity.]**

**Spanning Tree:** A spanning tree is a subset of Graph G, which has all the vertices covered with minimum possiblenumber of edges. Hence, a spanning tree does not have cycles and it cannot be disconnected.

**Prim’s Algorithm:** Prim's algorithm (also known as Jarník's algorithm) is a greedy algorithm that finds a minimum spanning tree for a weighted undirected graph.

**Kruskal’s Algorithm:** The Kruskal’s algorithm follows the greedy approach. It says that always select a minimum cost edge. Kruskal's Algorithm is used to find the minimum spanning tree for a connected weighted graph.

**The Single-Source Shortest Path (SSSP)** problem consists of finding the shortest paths between a given vertex v and all other vertices in the graph. Algorithm such as Dijkstra solve this problem

**Dijkstra Algorithm:** Dijkstra’s algorithm always searches for the shortest path. So, it selects a vertex with the shortest path and then find the shortest path to the vertices.

**=============================UNIT-06===================================**

**Backtracking algorithms**: Backtracking is a method of solving problems by trying solutions, and if a solution doesn’t work, it backtracks to try a different option.

**N Queens Problem:** In N Queens problem, the problem is to place n queens on an n \* n chessboard, so that no two queens are attacking each other.

**Graph coloring:** It is the procedure of assignment of colors to each vertex of a graph G such that no adjacent vertices get same color

**Hamiltonian Cycle Problem:** Hamiltonian Cycle Problem Given a graph, we start from some starting vertex and visit all the vertices exactly once and return to the starting vertex, so that it forms a cycle.

**=============================UNIT-08===================================**

**First fit alogorithm:** First Fit algorithm packs each item into the first bin where it fits, possibly opening a new bin if the item cannot fit into any currently open bin.

**Best Fit Alogrithm:** We have to use maximum space and leave less space not save more space!

**Pattern matching algorithm:** String or pattern matching algorithm try to find a place where

one or several string (called a pattern) are found within a text.

**Brute Force :** Brute Force Algorithms works by searching each element sequentially until the desired result is found or all options are exhausted.

**Knuth–Morris–Pratt algorithm:** In computer science, the Knuth–Morris–Pratt algorithm (or KMP algorithm) is a string-searching algorithm that searches for occurrences of a "word" within a main "text string"

**Boyer-Moore:** The Boyer-Moore algorithm is an efficient string-searching algorithm that finds the occurrence of a pattern within a text. It skips sections of the text by using preprocessed information from the pattern, allowing it to quickly bypass parts of the text that cannot contain the pattern.

**=============================UNIT-12===================================**

**Machine Learning:** Machine learning is a field of artificial intelligence where computers learn from data to make decisions or predictions without being explicitly programmed for specific tasks.

**Supervised Learning:** Trains on labeled data (input-output pairs) to predict outcomes for new data. Example: Predicting house prices based on historical data.

**Unsupervised Learning:** Trains on unlabeled data to find patterns or groupings. Example: Grouping customers by purchasing behavior without predefined categories.

**Inductive Learning:** Inductive learning is a process where the learner discovers rules by observing examples. The inductive learning is also known as discovery learning.

**Rule-Based Learning:** Rule-based learnings just another type of learning which makes the class decision depending by using various “if..else” rules.

**Reinforcement Learning:** Trains an agent to make decisions by rewarding it for good actions and penalizing it for bad ones. The goal is to maximize cumulative rewards over time. Example: Teaching a robot to navigate a maze by rewarding it when it reaches the goal.

**Huffman Coding:** Huffman coding is a compression algorithm that assigns shorter codes to more frequently occurring characters and longer codes to less frequent ones. This results in efficient data compression by reducing the overall size of the data.

**Data Compression:** It means restructuring or modifying data in order to reduce its size.

**=============================UNIT-13===================================**

**NP-complete problem**, any of a class of computational problems for which no efficient solution algorithm has been found.

**Traveling salesman problem**, an optimization problem in graph theory in which the nodes (cities) of a graph are connected by directed edges (routes), where the weight of an edge indicates the distance between two cities. The problem is to find a path that visits each city once, returns to the starting city, and minimizes the distance traveled.

**Deterministic algorithm:** A deterministic algorithm is an algorithm that, given a particular input, will always produce the same output.

**Non-Deterministic algorithm:** A non-deterministic algorithm can provide different outputs for the same input on different executions.

A **polynomial-time algorithm** is one whose running time grows as a polynomial function of the size of its input.

**=============================UNIT-14===================================**

**Cook’s Theorem:** In computational complexity theory, the Cook–Levin Theorem, also known as Cook's Theorem, states that the Boolean Satisfiability problem is NP-complete. That is, it is in NP, and any problem in NP can be reduced in polynomial time by a deterministic Turing machine to the Boolean Satisfiability problem.

**===============================LIVE VIVA====================================**