Design and Analysis of Algorithms

UNIT-I: Sorting

1. Selection Sort: Iteratively selects the smallest element from the unsorted part and places it at the beginning.

   Example: Sorting [3, 1, 4, 2] results in [1, 2, 3, 4].

   Time Complexity: O(n^2), Space Complexity: O(1)

2. Insertion Sort: Builds the sorted array one element at a time.

   Example: Sorting [3, 1, 4, 2] results in [1, 2, 3, 4].

   Time Complexity: O(n^2), Space Complexity: O(1)

3. Bubble Sort: Repeatedly swaps adjacent elements if they are in the wrong order.

   Time Complexity: O(n^2), Space Complexity: O(1)

4. Heap Sort: Builds a max-heap from the array and repeatedly extracts the maximum element.

   Time Complexity: O(n log n), Space Complexity: O(1)

5. Linear Time Sorting: Algorithms like Counting Sort sort integers in O(n).

   Time Complexity: O(n + k), Space Complexity: O(k)

UNIT-II: Graphs

1. Graph Traversals: Includes Depth-First Search (DFS) and Breadth-First Search (BFS).

   Time Complexity: O(V + E), Space Complexity: O(V)

2. Testing Bipartiteness: Check if a graph can be colored using 2 colors without conflicts.

   Time Complexity: O(V + E), Space Complexity: O(V)

3. Directed Acyclic Graphs (DAGs): Useful for task dependencies like Topological Ordering.

   Time Complexity: O(V + E), Space Complexity: O(V)

UNIT-III: Divide and Conquer

1. Merge Sort: Divides the array into halves, sorts, and merges.

   Time Complexity: O(n log n), Space Complexity: O(n)

2. Quick Sort: Picks a pivot, partitions, and sorts recursively.

   Time Complexity: O(n log n), worst-case O(n^2). Space Complexity: O(log n)

3. Maximum Subarray Problem: Finds the subarray with the maximum sum.

   Time Complexity: O(n log n)

4. NP-Completeness: Hard-to-solve problems like Traveling Salesman Problem (TSP).

UNIT-IV: Greedy and Dynamic Programming

1. Greedy Algorithms: Make locally optimal choices at each step.

   Example: Fractional Knapsack Problem.

   Time Complexity: O(n log n)

2. Minimum Spanning Tree: Finds edges connecting all vertices with minimum weight.

   Time Complexity: O(E log V)

3. Dynamic Programming: Solves problems by breaking into overlapping subproblems.

   Example: Fibonacci sequence.

   Time Complexity: O(n), Space Complexity: O(n)

4. Subset Sum Problem: Determines if a subset sums to a target.

   Time Complexity: O(n * W), Space Complexity: O(W)