DIVIDE AND CONQUER

Min Max algorithm:- Time complexity:- O(n) but number of comparisons are less than naive approach

Space complexity:- O(n) due to recursion stack used

Quick sort:- Time Complexity:- O(nlogn) in best case and average case , O(n^2) in worst case

Worst case when array is already sorted in increasing or decreasing order

Space Complexity:- O(logn) in best case or average case and O(n) in worst case due to recursion stack

Merge sort:- Time Complexity:-O(nlogn) both in best case and worst case

Space Complexity:-O(n) for the extra array

GREEDY ALGORITHM

Fractional Knapsack:- time- O(nlogn) for sorting +O(n) for traversing the array so overall O(nlogn)

space- O(1)

Job Sequencing:- time- O(nlogn) for sorting + O(n^2) for traversing and searching free slot so overall O(n^2)

space- O(n) for extra array used

Prim's Algorithm:- time- O(nlogn+elogn) using priority queue and adjacency list

space- O(n) for extra arrays

Dijkstra's Algorithm:- time- O(n+elogn) using priority queue and adjacency list

space- O(n) for extra arrays

Kruskal's Algorithm:- Time complexity:- O(eloge+elogn)

Space complexity:- O(n) for extra arrays

BACKTRACKING ALGORITHM

M-Coloring:- Time Complexity:- O(m^n) where n is number of nodes of graph and m is number of colors

Space Complexity:- O(n)+O(m) due to color array and recursion stack apart for graph storage

N-Queen:- Time Complexity:- O(n^n) or O(n!)
Space Complexity:- O(n^2) for grid array

GRAPH TRAVERSAL ALGORITHM

BFS:- Time Complexity:- The Time complexity of BFS is O(n + e) when Adjacency List is used and O(n^2) when Adjacency Matrix is used
where n stands for vertices and e stands for edges.
Space Complexity:- O(n) due to visited array and queue

DFS Iterative:- Time Complexity:- O(n+e), where n is the number of vertices and e is the number of edges in the graph
Space Complexity:- O(n) due to visited array and stack

DYNAMIC PROGRAMMING

Matrix Chain Multiplication:- Time complexity:- O(n^3)
Space complexity:- O(n^2) due to DP array

Floyd Warshall:- Time complexity:- O(n^3) where n is number of vertices
Space complexity:- O(n^2) for extra distance matrix

Bellman Ford:- Time Complexity:- O(ne) where n is number of vertices and e is number of edges
Space Complexity:- O(n) due to distance array

Heap Algorithm:-

Heap sort:- Time complexity:- O(nlogn) where heapify works at O(logn)
Space complexity:- O(logn) due to recursion stack