Design and Analysis of Algorithms

---

1. Understand and apply asymptotic analysis to determine upper, lower, and average complexity bounds of algorithms.
2. Compare and contrast best, average, and worst-case behaviors of algorithms to determine their efficiency.
3. Determine the big-Oh, big-Omega, and big-Theta notation for algorithms to express their time and space complexity.
4. Identify and classify algorithms into standard complexity classes based on their efficiency and performance.
5. Use empirical measurements to analyze and compare the performance of algorithms in practice.
6. Analyze and evaluate time and space trade-offs in algorithms to optimize their efficiency.
7. Solve problems using the divide and conquer strategy with algorithms like Merge Sort, Quick Sort, and Strassen's Matrix Multiplication.
8. Solve optimization problems using greedy algorithms such as Knapsack algorithm and Huffman Codes.
9. Apply dynamic programming techniques to solve problems efficiently and optimally, like Chained matrix multiplication.
10. Solve constraint satisfaction problems using backtracking algorithms like the 8 queens' problem.
11. Apply branch and bound techniques to optimize the solution for problems like the Travelling Salesperson problem.
12. Implement and analyze graph and tree algorithms such as BFS, DFS, Topological Sort, Minimum Spanning Tree algorithms, Dijkstra's Algorithm, and Bellman Ford Algorithm.
13. Design and analyze advanced data structures including Binary Search Tree, AVL tree, 2-3 Tree, Red Black Tree, and Binomial Heaps.
14. Understand the complexity theory including tractable and intractable problems, computable functions, and the concepts of P and NP classes.
15. Identify and analyze NP-complete problems using polynomial reducibility and Cook's theorem.