

Design and Analysis of Algorithms

Chapter-1 (Algorithms and Program Performance)

- Designing and analyzing algorithms,
- Time and Space complexity,
- Average and worst case Analysis,
- Asymptotic notations,

Recurrence equations and their solution:

- substitution method,
- recursion-tree method,
- master method.

Chapter-2 (Review of Data Structures)

- Arrays,
- Stacks,
- Queues,
- Pointers,
- Linked Lists (One –way, Two-way and circular Two-way),
- Hashing,
- Trees (BST, B Tree, balanced trees (AVL, Red black trees)),
- Heaps,
- Graphs

Chapter-3 (Sorting algorithm)

Sorting in linear time:

- counting sort,
- radix sort,
- bucket sort

UNIT-II

[15h]

Chapter-4 (Divide and conquer & Greedy algorithms)

Divide and conquer:

- The General method,
- Binary search,
- Finding maximum and minimum of a sequence of numbers,
- 2 way Merge sort,
- Quick sort,
- Selection sort,
- Strassen's matrix multiplication.

Greedy algorithms:

- The general method,
- Fractional Knapsack problem,
- job sequence,
- Min spanning tree,
- Optimal merge pattern,
- Huffman coding,
- Dijkstra algo

Minimum cost spanning tree:

- Prim's Algorithm,
- Kruskal Algorithm;

Others

- Huffman coding,
- Optimal merge patterns.

Chapter-5 (Dynamic programming)

- The general method, multistage graph
- 0/1 knapsack,
- Subset Sum problem,
- Change making problem,
- optimal binary search tree,
- Matrix-chain Multiplication,
- Longest common Subsequence Problem,
- Travelling salesman problem.

Comparison of Divide & Conquer and Dynamic Programming techniques.

Chapter-6 (Backtracking & Branch and Bound)

Backtracking:

- The general method,
- N-queen's problem,
- sum-of-subsets,
- Hamiltonian cycles.

Branch and Bound:

- Branch and Bound method,
- 0/1 Knapsack problem,
- Travelling salesperson problem.

UNIT-III

[15h]

Chapter-7 (Graph Algorithms)

- Representation of Graphs,
- Depth First Search,
- Breadth First search,
- Topological sort,

Single source shortest path:

- Dijkstra Algorithm &
- Bellman Ford Algorithm.

All-pair shortest paths:

- Floyd Warshall Algorithm,

← DP

Minimum Spanning Tree:

- Sollin's algorithm.

↖

Not much important. it is mix of prim's and kruskal

Chapter-9 (Miscellaneous topics)

- Euclid Algorithm for GCD of 2 numbers,
- Modulo arithmetic,
- Chinese remainder theorem,

String manipulation/matching algorithms:

- Rabin Karp algorithm,
- KMP (Knuth-Morris-Pratt) algorithm,
- Boyer-Moore algorithm;
- Convex Hull.

TEXT BOOKS

1. Cormen, Leiserson, Rivest, Stein, “*Introduction to Algorithms*”, Prentice Hall of India, 3rd edition 2012. problem, Graph coloring.
2. Horowitz, Sahni and Rajasekaran, “*Fundamentals of Computer Algorithms*”, University Press (India), 2nd edition.

REFERENCE BOOKS

1. Tanenbaum, Augenstein, &Langsam, “*Data Structures using C and C++*”, Prentice Hall of India.
2. Brassard, Bratley, “*Fundamentals of Algorithms*”, Prentice Hall of India.
3. Knuth “*The Art of Computer Programming, Volume 1: Fundamental Algorithms*” (Addison-Wesley, Third Edition).
4. Lipschutz, S., “*Data Structures, Schaum's Outline Series*”, Tata McGraw Hill.
5. Kruse, “*Data Structures & Program Design*”, Prentice Hall of India.
6. Aho, Haperoft and Ullman, “*The Design and analysis of Computer Algorithms*”, Pearson Education India.

List of Experiments

UNIT-I

1. Code and analyze to compute the greatest common divisor (GCD) of two numbers
2. Code implement power function in $O(\log n)$ time complexity
3. Code to find frequency of elements in a given array in $O(n)$ time complexity.
4. (i) Code to Insert and Delete an element at the beginning and at end in Doubly and Circular Linked List.
(ii) Code to push & pop and check Iseempty, Isfull and Return top element in stacks using templates.

UNIT-II

5. Code and analyze to find an optimal solution to matrix chain multiplication using dynamic programming.
6. To implement subset-sum problem using Dynamic Programming
7. Code to implement 0-1 Knapsack using Dynamic Programming

UNIT-III

8. Code and analyze to do a depth-first search (DFS) on an undirected graph. Implementing an application of DFS such as (i) to find the topological sort of a directed acyclic graph, OR (ii) to find a path from source to goal in a maze.
9. Code and analyze to find shortest paths in a graph with positive edge weights using Dijkstra's algorithm.
10. Code and analyze to find all occurrences of a pattern P in a given string S.