

DSC10/DSC07/GE7a: DESIGN AND ANALYSIS OF ALGORITHMS

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

| Course title & Code | Credits | Credit distribution of the course | | | Eligibility criteria | Pre-requisite of the course (if any) |
|-----------------------------------|---------|-----------------------------------|----------|---------------------|----------------------|--------------------------------------|
| | | Lecture | Tutorial | Practical/ Practice | | |
| Design and Analysis of Algorithms | 4 | 3 | 0 | 1 | Pass in Class XII | Data Structures |

Course Objectives

The course is designed to develop an understanding of various algorithm design techniques and apply them to problem-solving. The course shall also enable the students to verify the correctness of algorithms and analyze their time complexity.

Learning Outcomes

On successful completion of the course, students will be able to:

- Analyze and compare the asymptotic time and space complexity of algorithms to assess their efficiency.
- Design and implement algorithms using fundamental techniques such as divide and conquer, greedy, and dynamic programming.
- Apply and evaluate standard algorithms for searching, sorting, graph processing, and optimization problems.
- Demonstrate understanding of algorithm correctness and justify design choices through theoretical analysis.
- Construct and analyze hashing-based data structures using appropriate hash functions and collision resolution schemes.

Syllabus

Unit 1

(16 hours)

Linear Search, Binary Search, Insertion Sort, Selection Sort, Bubble Sort, Heapsort, Linear Time Sorting, running time analysis and correctness, Introduction to divide and conquer technique, Merge Sort, Quick Sort, Randomized quicksort, Maximum-subarray problem, Strassen's algorithm for matrix multiplication.

Unit 2

(16 hours)

Review of graph traversals, graph connectivity, testing bipartiteness, Directed Acyclic Graphs and Topological Ordering, Introduction to the Greedy algorithm design approach, Minimum Spanning Tree, Shortest Path Problem, Interval Scheduling, fractional knapsack problem, and analysis of time

complexity.

Unit 3

(9 hours)

Introduction to the Dynamic Programming approach, Weighted Interval Scheduling, Integer Knapsack problem, application to the subset sum problem and analysis of time complexity.

Unit 4

(4 hours)

Hash Tables, Hash Functions, Collision Resolution Schemes.

Essential/recommended readings

1. Cormen, T.H., Leiserson, C.E., Rivest, R. L., Stein C. Introduction to Algorithms, 4th edition, Prentice Hall of India, 2022.
2. Kleinberg, J., Tardos, E. Algorithm Design, 1st edition, Pearson, 2013.

Additional references

1. Basse, S., Gelder, A. V., Computer Algorithms: Introduction to Design and Analysis, 3rd edition, Pearson, 1999.

Practical List (If any): (30 Hours)

1. Write a program to sort the elements of an array using Insertion Sort (The program should report the number of comparisons).
2. Write a program to sort the elements of an array using Merge Sort (The program should report the number of comparisons).
3. Write a program to sort the elements of an array using Heap Sort (The program should report the number of comparisons).
4. Write a program to multiply two matrices using Strassen's algorithm for matrix multiplication
5. Write a program to sort the elements of an array using Radix Sort.
6. Write a program to sort the elements of an array using Bucket Sort.
7. Display the data stored in a given graph using the Breadth-First Search algorithm.
8. Display the data stored in a given graph using the Depth-First Search algorithm.
9. Write a program to determine a minimum spanning tree of a graph using Prim's algorithm.
10. Write a program to implement Dijkstra's algorithm to find the shortest paths from a given source node to all other nodes in a graph.
11. Write a program to solve the weighted interval scheduling problem.
12. Write a program to solve the 0-1 knapsack problem.

For the algorithms at S.No. 1, 2, and 3, test the algorithm on 100 different input sizes, varying from 30 to 1000. For each size, find the number of comparisons averaged on 10 different input instances; plot a graph for the average number of comparisons against each input size. Compare it with a graph of $n \log n$.