COURSE CONTENT

ADVANCED ALGORITHMS								
II Semester: CSE								
Course Code	**Category**	**Hours / Week**			**Credits**	**Maximum Marks**		
		L	**T**	**P**	**C**	**CIA**	**SEE**	**Total**
2525803	**Foundation**	3	0	0	3	40	60	100
Contact Classes: 45	**Tutorial Classes: Nil**	**Practical Classes: Nil**				**Total Classes: 45**		
Prerequisites: UG level course in Algorithm Design and Analysis								

Course Overview:

This course explores advanced algorithmic design techniques including divide-and-conquer, dynamic programming, greedy methods, and randomized algorithms. It covers graph algorithms, network flows, string matching, and computational geometry with rigorous complexity analysis

Course Objectives:

1. Introduce students to the advanced methods of designing and analyzing algorithms.
2. The student should be able to choose appropriate algorithms and use it for a specific problem.
3. To familiarize students with basic paradigms and data structures used to solve advanced algorithmic problems.
4. Students should be able to understand different classes of problems concerning their computation difficulties.
5. To introduce the students to recent developments in the area of algorithmic design.

Course Outcomes: After Completion of the Course, Students should be able to

1. Apply sorting, graph traversal, and shortest path algorithms to solve optimization problems in networks and real-time systems.
2. Analyze matroids, greedy algorithms, and graph matching techniques for applications in resource allocation and scheduling.
3. Implement flow networks, matrix computations, and divide-and-conquer strategies for solving large-scale scientific and engineering problems.
4. Evaluate dynamic programming, modular arithmetic, and fast polynomial/DFT algorithms for secure communications and cryptography.
5. Design solutions for NP-complete problems using advanced paradigms and recent data structures to support multidisciplinary research.

UNIT - I:

Sorting:

Review of various sorting algorithms, topological sorting Graph:

Definitions and Elementary Algorithms: Shortest path by BFS, shortest path in edge-weighted case (Dijkstra's), depth-first search and computation of strongly connected components, emphasis on correctness proof of the algorithm and time/space analysis, example of

amortized analysis.

UNIT - II:

Matroids:

Introduction to greedy paradigm, algorithm to compute a maximum weight maximal independent set. Application to MST.

Graph Matching:

Algorithm to compute maximum matching. Characterization of maximum matching by augmenting paths, Edmond's Blossom algorithm to compute augmenting path.

UNIT - III:

Flow-Networks:

Maxflow-mincut theorem, Ford-Fulkerson Method to compute maximum flow, Edmond-Karp maximum- flow algorithm.

Matrix Computations:

Strassen's algorithm and introduction to divide and conquer paradigm, inverse of a triangular matrix, relation between the time complexities of basic matrix operations, LUP-decomposition

UNIT - IV:

Shortest Path in Graphs:

Floyd-Warshall algorithm and introduction to dynamic programming paradigm. More examples of dynamic programming.

Modulo Representation of integers/polynomials:

Chinese Remainder Theorem, Conversion between base-representation and modulo-representation. Extension to polynomials. Application: Interpolation problem.

Discrete Fourier Transform (DFT):

In complex field, DFT in modulo ring. Fast Fourier Transform algorithm. Schonhage-Strassen Integer Multiplication algorithm.

UNIT - V:

Linear Programming: Geometry of the feasibility region and Simplex algorithm

NP-completeness: Examples, proof of NP-hardness and NP-completeness.

Recent Trends in problem solving paradigms using recent searching and sorting techniques by applying recently proposed data structures

TEXT BOOKS:

1. Introduction to Algorithms — Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, 4th Edition, MIT Press, 2022.
2. Algorithm Design — Jon Kleinberg and Éva Tardos**,** 1st Indian Reprint**,** Pearson, 2014.
3. The Algorithm Design Manual — Steven S. Skiena, 2nd Edition**,** Springer, 2008.

REFERENCE BOOKS:

1. Cormen, Leiserson, Rivest, Stein, "Introduction to Algorithms".
2. Aho, Hopcroft, Ullman "The Design and Analysis of Computer Algorithms".
3. Kleinberg and Tardos."Algorithm Design".

ELECTRONIC RESOURCES:

1. https://www.geeksforgeeks.org/fundamentals-of-algorithms/
2. https://www.tutorialspoint.com/algorithm_design/index.htm
3. https://www.coursera.org/specializations/algorithms
4. https://www.edx.org/learn/algorithms

MATERIALS ONLINE:

1. Course template
2. Tutorial question bank
3. Tech talk and Concept Video topics
4. Open-ended experiments
5. Definitions and terminology
6. Assignments
7. Model question paper – I
8. Model question paper – II
9. Lecture notes
10. E-Learning Readiness Videos (ELRV)