L	T	P	C
3	0	0	3

Course Code: CSEXXX
Semester: V

ALGORITHM DESIGN STRATEGIES & ANALYSIS

Course Objectives

This course will help the learner to acquire knowledge to develop efficient algorithm for a given application by selecting appropriate design technique and analyze its computational complexity

UNIT - I **11 Periods**

Introduction: Algorithm Specification - Performance Analysis - Space Complexity, Time Complexity - Asymptotic Notation (O, Ω, Θ) - Time and Space Trade-Offs - Analysis of Recursive Algorithms through Recurrence Relations - Substitution Method - Recursion Tree Method - Masters' Theorem **Fundamental Algorithmic Strategies:** Brute-Force Method - Heuristics Method - Travelling Salesman Problem - Greedy Method - General Method - Knapsack Problem - Job Sequencing with Deadlines

UNIT - II **11 Periods**

Advanced Strategies: Dynamic Programming Method - Optimal Binary Search Trees - String Editing - 0/1 Knapsack - Travelling Salesman problem - Branch and Bound Method - 0/1 Knapsack Problem - Travelling Salesman Problem - Backtracking Method - 8-Queens Problem - Sum of Subsets - Hamiltonian Cycles - Knapsack Problem

UNIT - III **11 Periods**

Graph and Tree Algorithms: Traversal algorithms - Breadth First Search - Depth First Search - Topological sort - Minimum Spanning Tree - Kruskal's and Prim's - Shortest path algorithms - Bellman Ford Algorithm - Dijkstra's Algorithm - Floyd-Warshall Algorithm -Flow Networks - Ford-Fulkerson Method

UNIT - IV **12 Periods**

Tractable and Intractable Problems: Nondeterministic Algorithms - The classes NP-Hard and NP-Complete - Cook's Theorem - Clique Decision Problem - Node Cover Decision Problem - Travelling Salesperson Decision Problem. **Advanced Topics:** Approximation algorithms - Scheduling Independent Tasks - Bin Packing - Interval Partitioning - Randomized algorithms - Class of problems beyond NP - P SPACE - Introduction to Quantum Algorithms

TEXTBOOK

1. Ellis Horowitz, Sartaj Sahni, Sanguthevar Rajasekaran, Fundamental of Computer Algorithms, Computer Science Press, Second Edition, 2008.
2. T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. Introduction to Algorithms, Prentice Hall of India, Third Edition, 2009. (Paperback-2011)

3. A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *The Design and Analysis of Computer Algorithms*, Pearson Education, 2003.

REFERENCES

1. Anany Levitin. Introduction to the Design and Analysis of Algorithm, Pearson Education, Third Edition, 2012.
2. Sara Baase and Allen Van Gelder. Computer Algorithms - Introduction to Design and Analysis, Pearson Education, Third Edition, 2008.
3. Jon Kleinberg and Éva Tardos. Algorithm Design, Pearson Education, First Edition, 2013.

ONLINE MATERIALS

1. https://onlinecourses.nptel.ac.in/noc21_cs68/preview
2. https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-046j-design-and-analysis-of-algorithms-spring-2015/

UNITWISE LEARNING OUTCOMES

Upon successful completion of each unit, the learner will be able to

Unit I	• Define asymptotic notations for time complexity analysis • Employ the techniques for solving recurrences to find the computational complexity of recursive algorithms • Develop algorithms using Brute-Force, Heuristic and Greedy Strategies
Unit II	• Develop algorithms using Dynamic Programming, Branch and Bound and Backtracking strategies for a given application
Unit III	• Judge and Select appropriate graph algorithms for a given application
Unit IV	• Define computability classes - P, NP, NP-Complete and NP-Hard • Employ approximation algorithms for solving NP-Complete and NP-Hard problems.

COURSE LEARNING OUTCOMES

Upon successful completion of this course, the learner will be able to

- Define asymptotic notations for time complexity analysis
- Employ the techniques for solving recurrences to find the computational complexity of recursive algorithms
- Judge and Select appropriate design strategy (Brute-Force, Heuristic, Greedy, Dynamic Programming, Branch & Bound and Backtracking) for solving a given application
- Judge and Select appropriate graph algorithms for a given application
- Define computability classes - P, NP, NP-Complete and NP-Hard
- Employ approximation algorithms for solving NP-Complete and NP-Hard problems