DESIGN AND ANALYSIS OF ALGORITHMS

III Semester: **CSE/IT**

Course Code	Category	Hours / Week			Credits	Maximum Marks		
		L	T	P	C	CIA	SEE	Total
AIT001	Core	3	1	-	4	30	70	100
Contact Classes: 45	Tutorial Classes: 15	Practical Classes: Nil				Total Classes: 60		

OBJECTIVES:

The course should enable the students to:

1. Calculate performance of algorithms with respect to time and space complexity.
2. Illustrate the graph traversals and tree traversals to solve the problems
3. Demonstrate the concepts greedy method and dynamic programming for several applications like knapsack problem, job sequencing with deadlines, and optimal binary search tree, TSP.
4. Illustrating the methods of backtracking and branch bound techniques to solve the problems like n-queens problem, graph coloring and TSP respectively

COURSE LEARNING OUTCOMES(CLO'S):

1. Use big O-notation formally to give asymptotic upper bounds on time and space complexity of algorithms Describe the performance of hybrid and electric vehicles.
2. Explain the use of big-Omega, big-Theta, and little-o notations to describe the amount of work done by an algorithm. Discuss the basic concepts of electric traction.
3. Use recurrence relations to determine the time complexity of recursive algorithms.
4. Evaluate and compare different algorithms using worst, average, and best-case analysis.
5. Solve elementary recurrence relations, e.g., using some forms of a Master Theorem. Give examples that illustrate time-space trade-offs of algorithms.
6. Demonstrate the ability to evaluate algorithms, to select from a range of possible options, to provide justification for that selection, and explain an implementation of the algorithm in a particular context.
7. Describe and use major algorithmic techniques (brute-force, greedy, divide-and-conquer, dynamic programming, and graph explorations).
8. Use a divide-and-conquer algorithm to solve an appropriate problem
9. Use a greedy approach to solve an appropriate problem and determine if the greedy rule chosen leads to an optimal solution.
10. Use dynamic programming to develop the recurrence relations and to solve an appropriate problem.
11. Use recursive backtracking to solve a problem such as navigating a maze
12. Explain the major graph algorithms and their analysis and employ graphs to model application Problems
13. Determine appropriate algorithmic approaches to apply to a given problem.
14. Describe heuristic problem-solving methods.
15. Understand the mapping of real-world problems to algorithmic solutions
16. Define the classes P and NP.
17. Explain the significance of NP-completeness.

18. Provide examples of NP-complete problems
19. Explain the impact of NP-complete problems to different application domains.
20. Explain the difference between NP-complete and NP-hard.
21. Prove that a problem is NP-complete.
22. Use reduction techniques between problems.
23. Demonstrate the use of approximation algorithms for NP-hard problems
24. Explain the Halting problem and other un-decidable problems.
25. Possess the knowledge and skills for employability and to succeed in national and international level competitive examinations.

UNIT - I	INTRODUCTION	Classes: 08

Introduction: Algorithm, Pseudo code for expressing algorithms, Performance Analysis-Space complexity, Time complexity, Asymptotic Notation-Big oh notation, Omega notation, Theta notation and Little oh notation, Probabilistic analysis, Amortized complexity Divide and conquer: General method, applications-Binary search, Quick sort, Merge sort, Strassen's matrix multiplication.

UNIT - II	SEARCHING AND TRAVERSAL TECHNIQUES	Classes: 10

Searching and traversal techniques, Disjoint set operations, union and find algorithms, Efficient non-recursive binary tree traversal algorithms, spanning trees, Graph traversals-Breadth first search, Depth first search, connected components, Bi-connected components.

UNIT - III	GREEDY METHOD AND DYNAMIC PROGRAMMING	Classes: 10

General method, applications-Job sequencing with deadlines, 0/1 knapsack problem, Minimum cost spanning trees, Single source shortest path problem. Dynamic Programming: General method, applications-Matrix chain multiplication, Optimal binary search trees, 0/1 knapsack problem, All pairs shortest path problem, Single source shortest path problem, Travelling sales person problem.

UNIT - IV	BACKTRACKING AND BRANCH AND BOUND	Classes: 08

Backtracking: General method, applications-8-queens problem, sum of subsets problem, graph coloring, Hamiltonian cycles. Branch and Bound: General method, applications-0/1 knapsack problem-LC Branch and Bound solution, FIFO Branch and Bound solution, Travelling sales person problem.

UNIT - V	NP-HARD AND NP-COMPLETE PROBLEMS	Classes: 09

NP-Hard and NP-Complete problems: Basic concepts, Non-deterministic algorithms, the classes NP -Hard and NP, NP Hard problems, clique decision problem, chromatic number decision problem, Cook's theorem.

Text Books:

1. Horowitz, Satraj Sahni, Sanguthevar Rajasekharan, "Fundamentals of Computer Algorithms", Universities Press, 2nd Edition, 2015.
2. Alfred V. Aho, John E. Hop croft, Jeffrey D, "The Design And Analysis Of Computer Algorithms", Pearson India, 1st Edition, 2013.

Reference Books:

1. Levi tin A, "Introduction to the Design and Analysis of Algorithms", Pearson Education, 3rdEdition, 2012.
2. Goodrich, M. T. R Tamassia, "Algorithm Design Foundations Analysis and Internet Examples", John Wiley and Sons, 1stEdition, 2001.
3. Base Sara Allen Vangelder, "Computer Algorithms Introduction to Design and Analysis", Pearson Education, 3rdEdition ,1999

Web References:

1. http://www.personal.kent.edu/~rmuhamma/Algorithms/algorithm.html
2. http://openclassroom.stanford.edu/MainFolder/CoursePage.php?course=IntroToAlgorithms
3. http://www.facweb.iitkgp.ernet.in/~sourav/daa.html

E-Text Books:

1. http://ebook/com/item/introduction_to_the_design_and_analysis_of_algorithms_3rd_edition_anany_le vitin/
2. https://drive.google.com/file/d/0B_Y1VbyboEDBTDVxVXpVbnk4TVE/edit?pref=2&pli=1
3. http://www.amazon.com/Computer-Algorithms-Introduction-Design-Analysis/dp/0201612445