ADVANCED ALGORITHMS AND ANALYSIS

Semester I (Computer Engineering)
SUB CODE: MECE102

Teaching Scheme (Credits and Hours)

Teaching scheme				Total	Evaluation Scheme					
L	T	P	Total	Credit	Theory		Mid Sem Exam	CIA	Pract.	Total
Hrs	Hrs	Hrs	Hrs		Hrs	Marks	Marks	Marks	Marks	Marks
04	00	02	06	05	3	70	30	20	30	150

LEARNING OBJECTIVES:

The objective of this course is

- To Introduce various designing techniques and methods for algorithms
- Performance analysis of Algorithms using asymptotic and empirical approaches
- Demonstrate a familiarity with major algorithms and data structures.
- To give clear idea on algorithmic design paradigms like Divide-and-Conquer, Dynamic Programming, Greedy, Branch and Bound etc.
- Applying efficient algorithms in engineering problems

OUTLINE OF THE COURSE:

Unit No	Topics
1	Fundamentals of Algorithms
2	Analysis and Design Techniques
3	Advanced Data Structures
4	Graph Algorithms
5	Algorithms for parallel computers
6	Approximation algorithms
7	Complexity Theory

Total hours (Theory): 60

Total hours (Practical): 30

Total hours: 90

DETAILED SYLLABUS:

Sr. No	Topic	Lecture Hours	Weight age (%)
Module I : Basics of Algorithms			
1	**Fundamental of Algorithms** • Analysis of Algorithms (algorithm definitions, Orders of Magnitude, Growth rates, Arithmetic and geometric series, harmonic numbers, sets, relations, functions, combinations) • Recurrence Relations • Amortized analysis • Sorting algorithms and Analysis (Compare-exchange, divide-conquer, linear time, tree sorting) • Applications of sorting and searching	08	15
2	**Analysis and Design Techniques** • Dynamic programming • Greedy algorithms	07	5
3	**Data Structures:** • Introduction of basic data structures like stack, queue, linked-list, binary tree, binary search tree • Red-Black trees • Augmenting data structures **Advanced Data Structures:** • B-trees • Binomial heaps • Fibonacci heaps	10	15
Module II : Advanced Algorithms and Applications			
4	**Graph Algorithms:** • Elementary graph algorithms • Minimum spanning trees • Graph coloring • Single source shortest paths • All- pairs shortest paths • Maximum flow	10	15
5	**Algorithms for parallel computers** • Analysis of parallel algorithms • Sorting networks (Bitonic, odd-even merge, butterfly) • Parallel sorting algorithms • Parallel searching algorithms	10	20

	• Prefix sum computations • Matrix operations		
6	**Approximation algorithms** • The vertex-cover problem • The traveling-salesman problem • The set-covering problem • The subset-sum problem	10	20
7	**Complexity Theory** • NP-completeness - Complexity Classes • NP-Hard and NP-Complete problems	05	10

INSTRUCTIONAL METHOD AND PEDAGOGY (Continuous Internal Assessment (CIA) Scheme)

- At the start of course, the course delivery pattern, prerequisite of the subject will be discussed.
- Lectures will be conducted with the aid of multi-media projector, black board, OHP etc.
- Attendance is compulsory in lecture and laboratory which carries 10 marks in overall evaluation.
- One internal exam will be conducted as a part of internal theory evaluation.
- Assignments based on the course content will be given to the students for each unit and will be evaluated at regular interval evaluation.
- Surprise tests/Quizzes/Seminar/tutorial will be conducted having a share of five marks in the overall internal evaluation.
- The course includes a laboratory, where students have an opportunity to build an appreciation for the concepts being taught in lectures.
- Experiments shall be performed in the laboratory related to course contents.

STUDENTS LEARNING OUTCOMES:

On successful completion of the course, the student will:

- Be able to check the correctness of algorithms using inductive proofs and loop invariants.
- Be able to compare functions using asymptotic analysis and describe the relative merits of worst-, average-, and best-case analysis.
- Be able to solve recurrences using the master, the iteration/recursion tree, and the substitution method.
- Become familiar with a variety of sorting algorithms and their performance characteristics (eg, running time, stability, space usage) and be able to choose the best one under a variety of requirements.
- Be able to understand and identify the performance characteristics of fundamental algorithms and data structures and be able to trace their operations for problems such as sorting, searching, selection, operations on numbers, polynomials and matrices, and graphs.

- Explain the major graph algorithms and their analyses. Employ graphs to model engineering problems, when appropriate. Synthesize new graph algorithms and algorithms that employ graph computations as key components, and analyze them.
- Be able to use the design techniques introduced i.e. dynamic programming, greedy algorithm etc. to design algorithms for more complex problems and analyze their performance.
- Become familiar with the major graph algorithms and their analyses. Employ graphs to model engineering problems, when appropriate.
- Demonstrate a familiarity with applied algorithmic settings - such as computational geometry, operations research, security and cryptography, parallel and distributed computing, operating systems, and computer architecture - by reciting several algorithms of importance to different fields.

REFERENCE BOOKS:

1. Introduction to Algorithms by Coreman MIT Press
2. Design and Analysis of Computer Algorithms by Aho,Hopcroft and Ullman ,Pearson
3. The Algorithm Design Manual By Steve s. Skiena
4. Fundamental of Algorithms- Theory and Practice by Gilles Brassard and Paul Bratley
5. Data Structure and Algorithm by Hari Mohan Pandey, Laxmi Publication

LIST OF PRACTICALS:

Sr. No	Name of Experiment
1	Find out Big - Oh and Big – Omega of the function. Take necessary data like degree of the function, coefficients, etc… .
2	Implement insertion sort, bubble sort, merge sort, shell sort and quick sort.
3	Implement Heap Sort, Radix Sort, Count Sort
4	Implement AVL Tree
5	Implement B Tree and R tree
6	Implement following problem with Greedy Algorithm. • Making a Change • Fractional Knap Sack • Traveling salesman Problem (Take 10 city for solving problem).
7	Implement the following problem with Dynamic Programming. • Making Change • Knap Sack • Longest Common Subsequence problem • Chain Matrix multiplication Problem
8	Implement following problem with Branch and Bound Technique • Task Assignment Problem • Subset sum Problem • Hamiltonian Circuit
9	Implement Minimum Spanning Tree using Prim and Kruskal algorithms
10	Implement following problem with Back Tracking Technique • Eight Queen Problem