Kadi Sarva Vishwavidyalaya

Faculty of Engineering and Technology

First Year Master of Engineering (Computer Engineering)

In Effect from Academic Year 2017-18

Subject Code: MECE102-N	**Subject Title: DESIGN AND ANALYSIS OF ALGORITHMS**

Teaching scheme				**Total Credit**	**Evaluation Scheme**					
L	**T**	**P**	**Total**		**Theory**		**Mid Sem Exam**	**CIA**	**Pract.**	**Total**
Hrs	**Hrs**	**Hrs**	**Hrs**		**Hrs**	**Marks**	**Marks**	**Marks**	**Marks**	**Marks**
04	**00**	**02**	**06**	**05**	**03**	**70**	**30**	**20**	**30**	**150**

LEARNING OBJECTIVES:

The objective of this course is:

- The complexity and performance analysis of different algorithms using asymptotic and empirical approaches
- Proper selection of the algorithms best suited for given applications
- To introduce various designing techniques and methods for algorithms
- To be familiar with various data structures and advanced data structures
- Applying efficient algorithms in engineering problems

OUTLINE OF THE COURSE:

Unit No	**Topics**	**Hours**
1	Fundamentals of Algorithms	10
2	Data Structures and Advanced Data Structures	12
3	Advanced Design and Analysis Techniques	10
4	Advanced Computer Algorithms	20
5	Computational Complexity	12

Total hours (Theory): 64
Total hours (Practical): 32
Total hours: 96

Kadi Sarva Vishwavidyalaya

Faculty of Engineering and Technology

First Year Master of Engineering (Computer Engineering)

In Effect from Academic Year 2017-18

DETAILED SYLLABUS:

Sr. No	Topic	Lecture Hours	Weight age (%)
1	**Fundamentals of Algorithms:** • Problem, algorithm definitions • Growth rates and asymptotic bounds (constant, logarithmic, linear, polynomial, exponential) • Amortized analysis • Sorting algorithms and Analysis (Compare-exchange, divide-conquer, linear time, tree sorting) • Applications of sorting and searching • Overview of Recurrence Relation	10	16
2	**Data Structures:** • Introduction of basic data structures like stack, queue, linked-list, binary tree, binary search tree, AVL tree • Red-Black tree, splay tree • Augmenting data structures **Advanced Data Structures:** • B-trees • Binomial heaps, Fibonacci heaps	12	19
3	**Advanced Design and Analysis Techniques:** • Dynamic programming (Sequence Alignment, Floyd-Warshall, Economic Optimization, Checkerboard, Egg Dropping Puzzle) • Greedy algorithms (Job Scheduling, MST - Prim and Kruskal, Malfatti's problem, Graph - Map coloring, Graph - Vertex Cover)	10	16
4	**Advanced Computer Algorithms:** • Network flow problems(max flow and min-cut) • Sorting networks (odd-even merge, bitonic, butterfly) • Parallel algorithms (Convex-hull, sorting, searching, merging) • Approximation algorithms (vertex cover, TSP, set covering, subset sum) • Randomized algorithms (classification, quick sort, min-cut, balls in bins)	20	30

5	**Computational Complexity:** • Resources for computation (time, space, non-determinism, randomness) and their associated complexity classes. • The complexity classes P, NP, NP-complete and their relationships • Probabilistic algorithms (Examples and complexity analysis)	12	19
	Total	64	100

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.
- Learn the graph algorithms and their analysis. 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.
- Be able to understand the applicability and analysis of parallel algorithms, approximation algorithms and randomized algorithms.
- Become familiar with computational complexity, complexity classes, their relationships and the applications of complexity classes.

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. Randomized Algorithms by Rajeev Motwani and Prabhakar Raghavan
6. Research papers

LIST OF PRACTICALS:

Sr. No	Name of Experiment
1	Implement insertion sort, bubble sort and selection sort along with the cost analysis and frequency counts.
2	Implement merge sort and quick sort along with the cost analysis and frequency counts.
3	Implement heap sort, radix sort and count sort along with the cost analysis and frequency counts
4	Implement AVL Tree.
5	Implement B Tree.
6	Implement job scheduling problem with Greedy Algorithm.
7	Implement any one algorithm with Dynamic Programming.
8	Implement Minimum Spanning Tree.
9	Implement any one approximation algorithm.
10	Implement any one randomized algorithm.