IV Semester

DESIGN AND ANALYSIS OF ALGORITHMS			
Course Code	**21CS42**	CIE Marks	50
Teaching Hours/Week (L:T:P: S)	3:0:2:0	SEE Marks	50
Total Hours of Pedagogy	40 T + 20 P	Total Marks	100
Credits	04	Exam Hours	03

Course Learning Objectives:

CLO 1. Explain the methods of analysing the algorithms and to analyze performance of algorithms.
CLO 2. State algorithm's efficiencies using asymptotic notations.
CLO 3. Solve problems using algorithm design methods such as the brute force method, greedy method, divide and conquer, decrease and conquer, transform and conquer, dynamic programming, backtracking and branch and bound.
CLO 4. Choose the appropriate data structure and algorithm design method for a specified application.
CLO 5. Introduce P and NP classes.

Teaching-Learning Process (General Instructions)

These are sample Strategies, which teachers can use to accelerate the attainment of the various course outcomes.

1. Lecturer method (L) does not mean only traditional lecture method, but different type of teaching methods may be adopted to develop the outcomes.
2. Show Video/animation films to explain functioning of various concepts.
3. Encourage collaborative (Group Learning) Learning in the class.
4. Ask at least three HOT (Higher order Thinking) questions in the class, which promotes critical thinking.
5. Adopt Problem Based Learning (PBL), which fosters students' Analytical skills, develop thinking skills such as the ability to evaluate, generalize, and analyze information rather than simply recall it.
6. Topics will be introduced in a multiple representation.
7. Show the different ways to solve the same problem and encourage the students to come up with their own creative ways to solve them.
8. Discuss how every concept can be applied to the real world - and when that's possible, it helps improve the students' understanding.

Module-1

Introduction: What is an Algorithm? It's Properties. Algorithm Specification-using natural language, using Pseudo code convention, Fundamentals of Algorithmic Problem solving, Analysis Framework-Time efficiency and space efficiency, Worst-case, Best-case and Average case efficiency.

Performance Analysis: Estimating Space complexity and Time complexity of algorithms.

Asymptotic Notations: Big-Oh notation (O), Omega notation (Ω), Theta notation (Θ) with examples, Basic efficiency classes, Mathematical analysis of non-recursive and recursive algorithms with examples.

Brute force design technique: Selection sort, sequential search, string matching algorithm with complexity Analysis.

Textbook 1: Chapter 1 (Sections 1.1,1.2), Chapter 2(Sections 2.1,2.2,2.3,2.4), Chapter 3(Section 3.1,3.2)

Textbook 2: Chapter 1(section 1.1,1.2,1.3)

Laboratory Component:

1. Sort a given set of n integer elements using Selection Sort method and compute its time complexity. Run the program for varied values of n> 5000 and record the time taken to sort. Plot a graph of the time taken versus n. The elements can be read from a file or can be generated using the random number generator. Demonstrate using C++/Java how the brute force method works along with its time complexity analysis: worst case, average case and best case.

Teaching-Learning Process	1. Problem based Learning. 2. Chalk & board, Active Learning. 3. Laboratory Demonstration.

Module-2

Divide and Conquer: General method, Recurrence equation for divide and conquer, solving it using Master's theorem. , Divide and Conquer algorithms and complexity Analysis of Finding the maximum & minimum, Binary search, Merge sort, Quick sort.

Decrease and Conquer Approach: Introduction, Insertion sort, Graph searching algorithms, Topological Sorting. It's efficiency analysis.

Textbook 2: Chapter 3(Sections 3.1,3.3,3.4,3.5,3.6)

Textbook 1: Chapter 4 (Sections 4.1,4.2,4.3), Chapter 5(Section 5.1,5.2,5.3)

Laboratory Component:

1. Sort a given set of n integer elements using Quick Sort method and compute its time complexity. Run the program for varied values of n> 5000 and record the time taken to sort. Plot a graph of the time taken versus n. The elements can be read from a file or can be generated using the random number generator. Demonstrate using C++/Java how the divide-and-conquer method works along with its time complexity analysis: worst case, average case and best case.
2. Sort a given set of n integer elements using Merge Sort method and compute its time complexity. Run the program for varied values of n> 5000 and record the time taken to sort. Plot a graph of the time taken versus n. The elements can be read from a file or can be generated using the random number generator. Demonstrate using C++/Java how the divide-and-conquer method works along with its time complexity analysis: worst case, average case and best case.

Teaching-Learning Process	1. Chalk & board, Active Learning, MOOC, Problem based Learning. 2. Laboratory Demonstration.

Module-3

Greedy Method: General method, Coin Change Problem, Knapsack Problem, solving Job sequencing with deadlines Problems.

Minimum cost spanning trees: Prim's Algorithm, Kruskal's Algorithm with performance analysis.

Single source shortest paths: Dijkstra's Algorithm.

Optimal Tree problem: Huffman Trees and Codes.

Transform and Conquer Approach: Introduction, Heaps and Heap Sort.

Textbook 2: Chapter 4(Sections 4.1,4.3,4.5)

Textbook 1: Chapter 9(Section 9.1,9.2,9.3,9.4), Chapter 6(section 6.4)

Laboratory Component:

Write & Execute C++/Java Program

1. To solve Knapsack problem using Greedy method.
2. To find shortest paths to other vertices from a given vertex in a weighted connected graph, using Dijkstra's algorithm.
3. To find Minimum Cost Spanning Tree of a given connected undirected graph using Kruskal's algorithm. Use Union-Find algorithms in your program.
4. To find Minimum Cost Spanning Tree of a given connected undirected graph using Prim's algorithm.

Teaching-Learning Process	1. Chalk & board, Active Learning, MOOC, Problem based Learning. 2. Laboratory Demonstration.

Module-4

Dynamic Programming: General method with Examples, Multistage Graphs.

Transitive Closure: Warshall's Algorithm. **All Pairs Shortest Paths**: Floyd's Algorithm,

Knapsack problem, Bellman-Ford Algorithm, Travelling Sales Person problem.

Space-Time Tradeoffs: Introduction, Sorting by Counting, Input Enhancement in String Matching-Harspool's algorithm.

Textbook 2: Chapter 5 (Sections 5.1,5.2,5.4,5.9)

Textbook 1: Chapter 8(Sections 8.2,8.4), Chapter 7 (Sections 7.1,7.2)

Laboratory Component:

Write C++/ Java programs to

1. Solve All-Pairs shortest Paths problem using Floyd's algorithm.
2. Solve Travelling Sales Person problem using Dynamic programming.
3. Solve 0/1 Knapsack problem using Dynamic Programming method.

Teaching-Learning Process	1. Chalk & board, Active Learning, MOOC, Problem based Learning. 2. Laboratory Demonstration.

Module-5

Backtracking: General method, solution using back tracking to N-Queens problem, Sum of subsets problem, Graph coloring, Hamiltonian cycles Problems.

Branch and Bound: Assignment Problem, Travelling Sales Person problem, 0/1 Knapsack problem

NP-Complete and NP-Hard problems: Basic concepts, non- deterministic algorithms, P, NP, NP-Complete, and NP-Hard classes.

Textbook 1: Chapter 12 (Sections 12.1,12.2) Chapter 11(11.3)

Textbook 2: Chapter 7 (Sections 7.1,7.2,7.3,7.4,7.5) Chapter 11 (Section 11.1)

Laboratory Component:

1. Design and implement C++/Java Program to find a subset of a given set S = {Sl, S2,..., Sn} of n positive integers whose SUM is equal to a given positive integer d. For example, if S = {1, 2, 5, 6, 8} and d= 9, there are two solutions {1, 2, 6} and {1, 8}. Display a suitable message, if the given problem instance doesn't have a solution.

2. Design and implement C++/Java Program to find all Hamiltonian Cycles in a connected undirected Graph G of n vertices using backtracking principle.

Teaching-Learning Process	1. Chalk & board, Active Learning, MOOC, Problem based learning. 2. Laboratory Demonstration.

Course outcome (Course Skill Set)

At the end of the course the student will be able to:

CO 1. Analyze the performance of the algorithms, state the efficiency using asymptotic notations and analyze mathematically the complexity of the algorithm.

CO 2. Apply divide and conquer approaches and decrease and conquer approaches in solving the problems analyze the same

CO 3. Apply the appropriate algorithmic design technique like greedy method, transform and conquer approaches and compute the efficiency of algorithms to solve the given problem.

CO 4. Apply and analyze dynamic programming approaches to solve some problems. and improve an algorithm time efficiency by sacrificing space.

CO 5. Apply and analyze backtracking, branch and bound methods and to describe P, NP and NP-Complete problems.

Assessment Details (both CIE and SEE)

The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%. The minimum passing mark for the CIE is 40% of the maximum marks (20 marks). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each subject/ course if the student secures not less than 35% (18 Marks out of 50) in the semester-end examination (SEE), and a minimum of 40% (40 marks out of 100) in the sum total of the CIE (Continuous Internal Evaluation) and SEE (Semester End Examination) taken together

Continuous Internal Evaluation:

Three Unit Tests each of **20 Marks (duration 01 hour)**

1. First test at the end of 5th week of the semester
2. Second test at the end of the 10th week of the semester
3. Third test at the end of the 15th week of the semester

Two assignments each of **10 Marks**

4. First assignment at the end of 4th week of the semester
5. Second assignment at the end of 9th week of the semester

Practical Sessions need to be assessed by appropriate rubrics and viva-voce method. This will contribute to **20 marks**.

- Rubrics for each Experiment taken average for all Lab components – 15 Marks.
- Viva-voce– 5 Marks (more emphasized on demonstration topics)

The sum of three tests, two assignments, and practical sessions will be out of 100 marks and will be **scaled down to 50 marks**
(to have a less stressed CIE, the portion of the syllabus should not be common /repeated for any of the methods of the CIE. Each method of CIE should have a different syllabus portion of the course).

CIE methods /question paper has to be designed to attain the different levels of Bloom's taxonomy