CO301: Design and Analysis of Algorithms			
Teaching Scheme		**Examination Scheme**	
Lectures:	**3Hrs. / Week**	**End-Sem Exam:**	**60 Marks**
Credits:	**3**	**Continuous Assessment:**	**40 Marks**
		Total:	**100 Marks**

==========

Prerequisite Course: Fundamentals of Data Structures Advanced Data Structures, Discrete Mathematics.

==========

Course Objectives:

1. To study and understand problem solving & basics of algorithm.
2. To study how to solve problems using greedy strategy.
3. To study how to solve problems using dynamic programming.
4. To study how to solve problems using backtracking and branch-n-bound strategies
5. To understand computational complexity theory.
6. To study parallel algorithms.

Course Outcome (COs): On completion of the course, students will be able to-

Course Outcomes	**Bloom's Taxonomy**	
	Level	**Descriptor**
1. **Understand** basics of problem solving and algorithm designing.	2	Understand
2. **Solve** problems using divide & conquer and greedy strategy.	3	Apply
3. **Solve** problems using dynamic programming strategy.	3	Apply
4. **Solve** problems using backtracking and branch-n-bound strategies.	3	Apply
5. **Understand** computational complexity theory.	2	Understand
6. **Understand** parallel algorithms.	2	Understand

Mapping of Course Outcomes to Program Outcomes (POs) & Program Specific Outcomes (PSOs):

	PO1	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**	**PSO1**	**PSO2**	**PSO3**
CO1	2	3	2	3	-	1	2	1	-	-	-	1	3		-
CO2	2	-	1	3	-	1	1	1	-	-	-	1	3	2	-
CO3	2	-	1	3	-	1	1	1	-	-	-	1	3	2	2
CO4	2	-	1	3	-	1	1	1	-	-	-	1	3	2	2
CO5	-	2	2	3	-	1	1	1	-	-	-	1	3		2
CO6	1	-	2	3	-	1	1	1	-	-	-	1	2	1	-

COURSE CONTENTS

Unit I	Problem Solving &Basic of Algorithm	No. of Hours	Cos
	Problem Solving: Definition of Problem, Problem solving principles, Classification & Strategies to solve problems, **Algorithm:** Definition, Asymptotic Notations, Time Complexities, Best, Worst & Average Case Analysis. **Types of algorithms:** Randomized, Approximate & Exact. **Case study**: Brute Force Method. **Application:** Medical Domain Problem and Algorithm.	6	CO1
Unit II	**Divide-&-Conquer and Greedy Strategy**	**No. of Hours**	**Cos**
	Divide and Conquer Strategy: Principle, Control Abstraction, Time complexity Analysis, Binary search algorithm. **Case study**: Merge Sort. **Application**: Google's Binary Search to Identify Malware. **Greedy Strategy:** Principle, Control Abstraction, Time Complexity Analysis, Knapsack Problem, **Case study**: Scheduling Algorithms-Job Scheduling. **Application**: Finding the Shortest Path on Google Map	6	CO2
Unit III	**Dynamic Programming**	**No. of Hours**	**Cos**
	Dynamic Programming: Principle, Control Abstraction, Time Complexity Analysis, Binomial Coefficients, 0/1 Knapsack, **Case study**: Optimal Binary Search Tree **Application** of DP: Path Finder GPS Application-Uber.	6	CO3
Unit IV	**Backtracking and Branch & Bound**	**No. of Hours**	**Cos**
	Backtracking: Principle, Control Abstraction, Time Complexity Analysis, 8-Queen Problem. **Case Study**: Sum of Subsets Problem. **Application** of BT: Sudoku Solver App **Branch-and-Bound:** Principle, Control Abstraction, Time Complexity Analysis, Knapsack Problem. **Case Study** :- Traveling Salesperson Problem, **Application:** Airline Crew Scheduling problem.	6	CO4
Unit V	**Complexity Theory**	**No. of Hours**	**Cos**
	Polynomial and Non-Polynomial Class Problems, Deterministic and Non-Deterministic Algorithms, P class problems, NP class problems. **NP complete class problems**- Vertex cover problem, 3-SAT problem **NP-Hard Problems:** Clique problem. **Case Study**:- Reduction problem (3SAT to Clique Problem). **Application of Complexity**: Visiting All the Cities in State, Country and Globe	6	CO5
Unit VI	**Parallel Algorithms**	**No. of Hours**	**Cos**
	Sequential and Parallel Computing, RAM & PRAM Models for Parallel Processing, Parallel Algorithm with Analysis. Optimal Parallel Algorithms, Multithreading. Machines Learning Algorithms: GMMs (Gaussian Mixture Model) Algorithm. **Case study:-** Analysis of Parallel Quick Sort. **Application**: Database and Data Mining for Banking Data	6	CO6

Books:

Text Books(T):

T1. Horowitz and Sahani, “Fundamentals of Computer Algorithms”, University Press.

T2.Gills Brassard and Paul Bartly, “Fundamentals of Algorithmic”, PHI, New Delhi.

T3. . A.V.Aho., “The Design and Analysis of Algorithms” Pearson Education, NewDelhi.

T4. K, Louden, “ Mastering Algorithms”, O‟ Reily Media Inc

Reference Books(R):

R1. Fayez Gebali, “Algorithms and Parallel Computing”, Willy Publication.

R2. Thomas H. Coreman and Charles R. L. Leiserson, “Introduction to Algorithm”, PHI Publications.

R3. M.R.Kabat, “Design and Analysis of Algorithms”, PHI Learning (p) Ltd.

R4. S. Sridhar ,““Design and Analysis of Algorithms”, Oxford University Press.

e-Resources(E):

E1:Robert Sedgewick and Kevin Wayne, ”algorithms” Princeton University. https://bank.engzenon.com/tmp/5e7f6ee5-d4dc-4aa8-9b0a-42d3c0feb99b/6062caf3-c600-4fc2-b413-4ab8c0feb99b/Algorithms-4th-Edition.pdf.

E2: Jeff Erickson, “algorithms”, a Creative Commons Attribution 4.0 International License https://jeffe.cs.illinois.edu/teaching/algorithms/book/Algorithms-JeffE.pdf.

E3: Junhui deng, “Data structures and algorithms specialization”, tsinghua University, Beijing. https://www.coursera.org/specializations/data-structures-algorithms-tsinghua

E4:Prof.Madhavan, ”Design and Analysis of Algorithms https://nptel.ac.in/courses/106106131

CO306: Design and Analysis of Algorithms Lab			
Teaching Scheme		**Examination Scheme**	
Practical:	**2 Hrs. / Week**	**Oral:**	**50 Marks**
Credits:	**1**	**Total:**	**50 Marks**

Prerequisite Course: Fundamentals of Data Structures, Advanced Data Structures, Discrete Mathematics

Course Objectives:

1. To study and implement application of divide and conquer algorithmic strategy
2. To study and implement application of greedy approach
3. To study and implement application of dynamic programming strategy
4. To study and implement application of backtracking approach
5. To identify and apply the suitable algorithmic strategy for the given problem.

Course Outcomes:

After successful completion of the course, students will able to:-

Course Outcome(s)		Bloom's Taxonomy	
		Level	**Descriptor**
CO1	Apply knowledge of divide and conquer technique to implement solution of problem statement.	3	Apply
CO2	Apply knowledge of greedy strategy implement solution of problem statement..	3	Apply
CO3	Apply the concept of dynamic programming to implement solution of problem statement.	3	Apply
CO4	Apply backtracking technique programming to implement solution of problem statement.	3	Apply
CO5	Apply the suitable algorithmic strategy to solve real world problem.	3	Apply

Mapping of Course Outcomes to Program Outcomes (POs) & Program Specific Outcomes (PSOs):

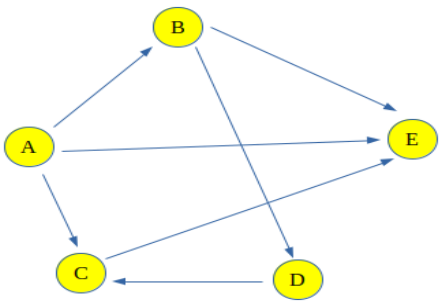
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	2	3	2	3	-	1	2	1	-	-	-	1	3		-
CO2	2	-	1	3	-	1	1	1	-	-	-	1	3	2	-
CO3	2	-	1	3	-	1	1	1	-	-	-	1	3	2	2
CO4	2	-	1	3	-	1	1	1	-	-	-	1	3	2	2
CO5	-	2	2	3	-	1	1	1	-	-	-	1	3		2

GENERAL INSTRUCTIONS:

1. Each student has to implement 5 assignment individually from set A to set E assigned by faculty members
2. Each student has to complete mini project in group of max 4 members based in CIA.

LIST OF EXPERIMENTS:

	Sr. No.	Assignment	CO
A	1.	**Implement a problem of number of zeroes.** **Statement**: Given an array of 1s and 0s which has all 1s first followed by all 0s? Find the number of 0s. Count the number of zeroes in the given array. **Input**: arr[] = {1, 1, 1, 1, 0, 0} **Output**: 2 **Input:** arr[] = {1, 0, 0, 0, 0} **Output**: 4	CO1
	2.	**Implement a problem of move all zeroes to end of array.** **Statement:** Given an array of random numbers, Push all the zero's of a given array to the end of the array. For example, if the given arrays is {1, 9, 8, 4, 0, 0, 2, 7, 0, 6, 0}, it should be changed to {1, 9, 8, 4, 2, 7, 6, 0, 0, 0, 0}. The order of all other elements should be same. **Input** : arr[] = {1, 2, 0, 4, 3, 0, 5, 0}; **Output** : arr[] = {1, 2, 4, 3, 5, 0, 0, 0};	CO1
	3.	**Implement a problem of smallest number with at least n trailing zeroes in factorial.** **Statement:** Given a number n. The task is to find the smallest number whose factorial contains at least n trailing zeroes. **Input** : n = 1 **Output** : 5 **Input** : n = 6 **Output** : 25	CO1
B	1.	**Implement a problem of activity selection problem with K persons.** **Statement**: Given two arrays S[] and E[] of size N denoting starting and closing time of the shops and an integer value K denoting the number of people, the task is to find out the maximum number of shops they can visit in total if they visit each shop optimally based on the following conditions: • A shop can be visited by only one person • A person cannot visit another shop if its timing collide with it **Input**: S[] = {1, 8, 3, 2, 6}, E[] = {5, 10, 6, 5, 9}, K = 2 **Output**: 4 **Input**: S[] = {1, 2, 3}, E[] = {3, 4, 5}, K = 2 **Output**: 3	CO2
	2.	**Implement a problem of maximize Profit by trading stocks based on given rate per day.** **Statement:** Given an array arr[] of N positive integers which denotes the cost of selling and buying a stock on each of the N days. The task is to find the maximum profit that can be earned by buying a stock on or selling all previously bought stocks on a particular day. **Input**: arr[] = {2, 3, 5} **Output**: 5 **Input**: arr[] = {8, 5, 1} **Output**: 0	CO2
	3.	**Implement a problem of minimum work to be done per day to**	CO2

		finish given tasks within D days problem. **Statement:** Given an array task[] of size N denoting amount of work to be done for each task, the problem is to find the minimum amount of work to be done on each day so that all the tasks can be completed in at most D days. Note: On one day work can be done for only one task. **Input**: task[] = [3, 4, 7, 15], D = 10 **Output**: 4 **Input**: task[] = [30, 20, 22, 4, 21], D = 6 **Output**: 22	
C	1.	**Implement Coin Change problem.** **Statement** Given an integer array of coins[ ] of size N representing different types of currency and an integer sum, The task is to find the number of ways to make sum by using different combinations from coins[]. Note: Assume that you have an infinite supply of each type of coin. ***Input**: sum = 4, coins[] = {1,2,3},* ***Output**: 4* ***Input**: sum = 10, coins[] = {2, 5, 3, 6}* ***Output**: 5*	CO3
	2.	**Implement Subset Sum Problem.** **Statement** Given a set of non-negative integers and a value sum, the task is to check if there is a subset of the given set whose sum is equal to the given sum. ***Input**: set[] = {3, 34, 4, 12, 5, 2}, sum = 9* ***Output**: True* ***Input**: set[] = {3, 34, 4, 12, 5, 2}, sum = 30* ***Output**: False*	CO3
	3.	**Implement Check if it is possible to transform one string to another.** **Statement** Given two strings s1 and s2 (all letters in uppercase). Check if it is possible to convert s1 to s2 by performing following operations. 1. Make some lowercase letters uppercase. 2. Delete all the lowercase letters. ***Input:** s1 = daBcd s2 = ABC* ***Output:** yes* ***Input:** s1 = argaju s2 = RAJ* ***Output:** yes*	CO3
D	1.	**Implement program to find all distinct subsets of a given set using Bit Masking Approach.** **Statement** Given an array of integers arr[], The task is to find all its subsets. The subset cannot contain duplicate elements, so any repeated subset should be considered only once in the output**.** ***Input**: S = {1, 2, 2}* ***Output**: {}, {1}, {2}, {1, 2}, {2, 2}, {1, 2,2}* ***Input**: S = {1, 2}* ***Output**: {}, {1}, {2}, {1, 2}*	CO4
	2.	**Implement program Count all possible Paths between two Vertices.** **Statement** Count the total number of ways or paths that exist between two vertices in a directed graph. These paths don't contain a cycle, the simple enough reason is that a cycle contains an infinite number of paths and hence they create a problem.  	CO4

		Input: *Count paths between A and E* ***Output:*** *Total paths between A and E are 4* ***Input:*** *Count paths between A and C* ***Output:*** *Total paths between A and C are 2*	
	3.	**Implement program to print all subsets of a given Set or Array** **Statement** Given a set of positive integers, find all its subsets. ***Input****: array = {1, 2, 3}* ***Output****: // this space denotes null element.* *1 1 2 1 2 3 1 3 2 2 3 3* ***Input****: 1 2* ***Output****: 1 2 1 2*	CO4
E		**Mini Project:-**Implement CIA assignment assigned in group as a CO301 (DAA theory subject) and store in source code in git repository.	CO5

Books:

Text Books(T):

T1. Horowitz and Sahani, "Fundamentals of Computer Algorithms", University Press.

T2.Gills Brassard and Paul Bartly, "Fundamentals of Algorithmic", PHI, New Delhi.

Reference Books(R):

R1. Fayez Gebali, "Algorithms and Parallel Computing", Willy Publication.

R2. Thomas H. Coreman and Charles R. L. Leiserson, "Introduction to Algorithm", PHI Publications.

e-Resources(E):

E1:Robert Sedgewick and Kevin Wayne, "algorithms" Princeton University. https://bank.engzenon.com/tmp/5e7f6ee5-d4dc-4aa8-9b0a-42d3c0feb99b/6062caf3-c600-4fc2-b413-4ab8c0feb99b/Algorithms-4th-Edition.pdf.

E2: Jeff Erickson, "algorithms", a Creative Commons Attribution 4.0 International License https://jeffe.cs.illinois.edu/teaching/algorithms/book/Algorithms-JeffE.pdf.

E3: https://www.geeksforgeeks.org/

E4: https://github.com/

E5: https://www.codechef.com/