COURSEPACK

SCHEME

The scheme is an overview of work-integrated learning opportunities and gets students out into the real world. This will give what a course entails.

<table>
<tr><td>Course Title</td><td colspan="3">Advanced Design and Analysis of Algorithms</td><td colspan="2">Course Type</td><td colspan="4">Theory</td></tr>
<tr><td>Course Code</td><td colspan="3">E2PV101T</td><td colspan="2">Class</td><td colspan="4">M Tech (CSE&AI) Sem I</td></tr>
<tr><td rowspan="6">Instruction delivery</td><td>Activity</td><td>Credits</td><td>Weekly Hours</td><td colspan="4" rowspan="2">Total Number of Classes per Semester</td><td colspan="2" rowspan="2">Assessment in Weightage</td></tr>
<tr><td>Lecture</td><td>3</td><td>3</td></tr>
<tr><td>Tutorial</td><td>0</td><td>0</td><td rowspan="4">Theory</td><td rowspan="4">Tutorial</td><td rowspan="4">Practical</td><td rowspan="4">Self-study</td><td rowspan="4">CIE</td><td rowspan="4">SEE</td></tr>
<tr><td>Practical</td><td>0</td><td>0</td></tr>
<tr><td>Self-study</td><td>0</td><td>6</td></tr>
<tr><td>Total</td><td>3</td><td>9</td></tr>
<tr><td>Course Lead</td><td colspan="2"></td><td>Course Coordinator</td><td colspan="6"></td></tr>
<tr><td rowspan="2">Names Course Instructors</td><td colspan="3">Theory</td><td colspan="6">Practical</td></tr>
<tr><td colspan="3">Dr S Srinivasan</td><td colspan="6">Dr S Srinivasan</td></tr>
</table>

COURSE OVERVIEW

This Advanced Algorithm Design and Analysis course covers both general design and analysis methodologies as well as particular algorithms for a range of issues. Among the specific subjects covered are sorting, searching, lower limits, trees, graph problem algorithms, amortized analysis, greedy algorithms, and dynamic programming

PREREQUISITE COURSE

PREREQUISITE COURSE REQUIRED	No

COURSE OBJECTIVE

Provides insight into why students learn what they learn before beginning a course.

1. To teach basic graph algorithms and their efficiency analysis using different algorithm design paradigms with illustrative problems.
2. Guide them to designing algorithms for the classical network flow problems and computationally hard problems using approximation algorithms.

COURSE OUTCOMES (COs)

After the completion of the course, the student will be able to:

CO No.	Course Outcomes
E2PV101T.1	Analyze algorithms and examine their worst case and average case behavior
E2PV101T.2	Model the problems using graph and write programs for solving complex problems.
E2PV101T.3	Apply dynamic programming and greedy approach to solve the problem
E2PV101T.4	Apply randomization and approximation Technique for solving NP hard problems

BLOOM'S LEVEL OF THE COURSE OUTCOMES

THEORY CO No.	Remember **KL1**	Understand **KL 2**	Apply **KL 3**	Analyse **KL 4**	Evaluate **KL 2**	Create **KL 6**
E2PV101T.1				√		
E2PV101T.2			√	√		
E2PV101T.3			√			
E2PV101T.4			√			

PROGRAM OUTCOMES (POs):

PO1 Computing Science knowledge: Apply the knowledge of mathematics, statistics, computing science, and information science fundamentals to the solution of computer application problems.

PO2 Problem analysis: Identity, formulate, review research literature, and analyze computing science problems reaching substantiated conclusions using the first principles of mathematics, natural sciences, and computer sciences.

PO3 Design/development of solutions: Design solutions for complex computing problems and design system components or processes that meet the specified needs with appropriate consideration

for public health and safety, and the cultural, societal, and environmental considerations.

PO4 Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5 Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern computing science and IT tools including prediction and modeling to complex computing activities with an understanding of the limitations.

PO6 IT specialist and society: Apply reason informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional computing science and information science practice.

PO7 Environment and sustainability: Understand the impact of professional computing science solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8 Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the computing science practice.

PO9 Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10 Communication: Communicate effectively on complex engineering activities with the IT analyst community and with society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11 Project management and finance: Demonstrate knowledge and understanding of computing science and management principles and apply these to one’s work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12 Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs):

PSO1 Have the ability to work with emerging technologies in computing requisite to Industry 4.0.

PSO2 Demonstrate Engineering Practice learned through industry internship and research project to solve live problems in various domains.

COURSE ARTICULATIONMATRIX

COs#/ POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
E2PV101T.1	2	2												
E2PV101T.2	3		2	2	2							2	1	
E2PV101T.3	3	2	2		2							2	1	
E2PV101T.4		2		2	2							2		

Note: 1-Low, 2-Medium, 3-High

COURSE ASSESSMENT

Assessment Tools	CIE							Total CIE marks	SEE
	QUIZ1 /AAT	CAT1	QUIZ2 AAT	CAT2	LAB	LAB EXAM	Course-based Project		
Theory	20	30	20	30	0	0	0	100	100

* Assignment, Quiz, Class test, SWAYAM/NPTEL/MOOCs and etc.

COURSE CONTENT

THEORY

Introduction - Overview of algorithmic design**,** asymptotic notation and its properties, Growth of Functions, Time complexity and Analysis of algorithms, Recurrence Relations.

Graph Algorithms- BFS, DFS, Topological sort, strongly connected components, The Bellman-Ford algorithm, Single-source shortest paths in directed acyclic graphs, Dijkstra's algorithm, shortest paths and matrix multiplication, Johnson's algorithm for sparse graphs, Flow networks

Dynamic programming and Greedy Technique – Principle of optimality - Coin changing problem, Computing a Binomial Coefficient – Floyd's algorithm – Multi stage graph - Optimal Binary Search Trees – Knapsack Problem and Memory functions. Greedy Technique – Container loading problem - 0/1 Knapsack problem, Optimal Merge pattern - Huffman Trees.

String Matching and Computational Geometry- naive string-matching, Rabin-Karp, String matching with finite automata, Knuth-Morris-Pratt algorithm Line-segment properties– Closest-Pair and Convex - Hull Problems.

NP Completeness- Lower - Bound Arguments - P, NP NP- Complete and NP Hard Problems. Backtracking – n-Queen problem - Hamiltonian Circuit Problem – Subset Sum Problem. Branch and Bound – LIFO Search and FIFO search - Assignment problem – Knapsack Problem – Travelling Salesman Problem - Approximation Algorithms for NP-Hard Problems

LESSON PLAN FOR THEORY COURSES (THEORY AND TUTORIAL CLASSES)

FOR THEORY 15 weeks * 3 Hours = 45 Classes (1credit = 1 Lecture Hour)

L-No	Topic for Delivery	Tutorial/ Practical Plan	Skill	Competency
1	Introduction	Lecture with interaction	Analyze and determine the running time and space of an algorithm	CO1
2	Overview of algorithmic design	Lecture with interaction		
3	asymptotic notation and its properties	Lecture with interaction		
4	Growth of Functions	Lecture with interaction		
5	Time complexity and Analysis of algorithms1	Lecture with interaction		
6	Time complexity and Analysis of algorithms2	Lecture with interaction		
7	Time complexity and Analysis of algorithms3	Lecture with interaction		
8	Recurrence Relations1	Lecture with interaction		
9	Recurrence Relations2	Lecture with interaction		
10	Revision1	Lecture with interaction		
11	BFS, DFS, Topological sort	Lecture with interaction	Implement suitable graphs to model engineering problems	CO2
12	strongly connected components	Lecture with interaction		
13	The Bellman-Ford algorithm	Lecture with interaction		
14	Single-source shortest paths in directed acyclic graphs	Lecture with interaction		
15	Dijkstra's algorithm	Lecture with interaction		
16	shortest paths and matrix multiplication	Lecture with interaction		
17	Johnson's algorithm for sparse graphs	Lecture with interaction		
18	Flow networks	Lecture with interaction		
19	Revision2	Lecture with interaction		
20	Dynamic programming and Greedy Technique	Lecture with interaction	Solve simple optimization problems using recursion and	CO3
21	Principle of optimality	Lecture with interaction		
22	Coin changing problem	Lecture with interaction		

23	Computing a Binomial Coefficient	Lecture with interaction	Optimal substructure techniques	
24	Floyd‘s algorithm	Lecture with interaction		
25	Multi stage graph	Lecture with interaction		
26	Optimal Binary Search	Lecture with interaction		
27	Knapsack Problem and Memory functions	Lecture with interaction		
28	Greedy Technique	Lecture with interaction		
29	Container loading problem	Lecture with interaction		
30	Optimal Merge pattern	Lecture with interaction		
31	Revision 3	Lecture with interaction		
32	String Matching naive string-matching	Lecture with interaction	Select and use appropriate matching algorithms for pattern matching.	CO4
33	Rabin-Karp	Lecture with interaction		
34	String matching with finite automata	Lecture with interaction		
35	Knuth-Morris	Lecture with interaction		
36	Pratt algorithm Line-segment properties	Lecture with interaction		
37	Hull Problems	Lecture with interaction		
38	P, NP NP- Complete and NP Hard Problems	Lecture with interaction	Implement computationally hard problems and tackling them using approximation algorithms	
39	Backtracking	Lecture with interaction		
40	n-Queen problem	Lecture with interaction		
41	Hamiltonian Circuit Problem	Lecture with interaction		
42	LIFO Search and FIFO search	Lecture with interaction		
43	Assignment problem	Lecture with interaction		
44	Travelling Salesman Problem	Lecture with interaction		
45	Approximation Algorithms for NP-Hard Problems	Lecture with interaction		

BIBLIOGRAPHY

Text Books

1. Cormen, Leiserson, Rivest and Stein, “Introduction to Algorithms”, 2nd Edition, by, McGraw-Hill, 2000.
2. E. Horowitz, and S. Sahni, “Fundamentals of Computer Algorithms”, Computer Science Press (1978).

Reference Books

1. Jon Kleinberg and Eva Tardos. Algorithm Design. Pearson Education, 2007.
2. Sanjoy Das Gupta, Christos Papadimitriou, Umesh Vazirani, Algorithms 1st Edition, Mcgraw Higher Ed, 2006.
3. Alfred V. Aho, John E. Hopcroft, Jeffery D.Ulman, Data Structures and Algorithms, Pearson; 1st edition, 2001.

Online references

- https://www.udemy.com/course/design-and-analysis-of-algorithms/?utm_source=adwords&utm_medium=udemyads&utm_campaign=DSA_Catchall_la.EN_cc.INDIA&utm_content=deal4584&utm_term=_._ag_82569850245_._ad_533220805574_._kw__._de_c_._dm__._pl__._ti_dsa-406594358574_._li_9302611_._pd__._&matchtype=&gclid=CjwKCAjw7c2pBhAZEiwA88pOF1cL4EZK31403p2Jpp_yXHo3gK6Rm3Arl4Ua7VRhJ_Lbfr63TybvIhoCoYQQAvD_BwE
- https://onlinecourses.swayam2.ac.in/cec20_cs03/preview
- https://www.classcentral.com/course/swayam-design-and-analysis-of-algorithms-3984

Practice Problems

1	Write a java program to implement the dfs algorithm for a graph.
2	Write a java program to implement the bfs algorithm for a graph.
3	Write a java program to implement backtracking algorithm for the N-queens problem.
4	Write a java program to implement the backtracking algorithm for the sum of subsets problem.
5	Write a java program to implement the backtracking algorithm for the Hamiltonian Circuits problem.
6	Write a java program to implement greedy algorithm for job sequencing with deadlines.
7	Write a java program to implement Dijkstra's algorithm for the Single source shortest path problem.
8	Write a java program that implements Prim's algorithm to generate minimum cost spanning tree.
9	Write a java program that implements Kruskal's algorithm to generate minimum cost spanning tree
10	Write a java program to implement Floyd's algorithm for the all-pairs shortest path problem.
11	Write a java program to implement Dynamic Programming algorithm for the 0/1 Knapsack problem.
12	Write a java program to implement Dynamic Programming algorithm for the Optimal Binary Search Tree Problem.
13	Write a java program to implement Greedy algorithm for the 0/1 Knapsack problem.
14	Implement insertion, deletion, searching of a BST, also write a routine to draw the BST horizontally
15	From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra's algorithm. Write the program in Java
16	Implement Travelling Sales Person problem using Dynamic programming
17	Design and implement in Java to find all Hamiltonian Cycles in a connected undirected Graph G of n vertices using backtracking principle
18	Design and implement in Java 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
19	Implement binary search and linear search in a program
20	Given a set of weights, form a Huffman tree from the weight and also find out the code corresponding to each weight
21	Given a set of weight and an upper bound M – Find out a solution to the Knapsack problem
22	Implement Strasseem's matrix multiplication algorithm for matrices whose order is a power of two
23	Given two sequences of character, find out their longest common subsequence using dynamic programming

24	Implement Floyd-Warshal algorithm.
25	Implement recursive binary search.
26	WAP to implement matrix chain multiplication
27	Code and analyze to find all occurrences of a pattern P in a given string S
28	Code and analyze to do a depth-first search (DFS) on an undirected graph to find the topological sort of a directed acyclic graph
29	Code and analyze to do a depth-first search (DFS) on an undirected graph to find a path from source to goal in a maze
30	Code and analyze to do a breadth-first search (BFS) on an undirected graph to find connected components of an undirected graph
31	Code and analyze to do a breadth-first search (BFS) on an undirected graph to check whether a given graph is bipartite
32	Write a program to find the solution for job sequencing with deadlines problems
33	Write a program to solve Sum of subsets problem for a given set of distinct numbers using backtracking.
34	Design and implement in Java to find all Hamiltonian Cycles in a connected undirected Graph G of n vertices using backtracking principle
35	Write a program to implement Longest Common Subsequence
36	Write a program to implement Longest Common Subsequence
37	Write a program to implement Naive Based String Matching
38	Write a program to implement N-Queens problem
39	Write a program for all pairs shortest path.
40	Write a program to implement Knuth-Morris-Pratt algorithm

(Course Lead) (Program Chair) (Dean)