K.S. SCHOOL OF ENGINEERING AND MANAGEMENT, BENGALURU - 560109

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

CO-PO Mapping

Course: DESIGN AND ANALYSIS OF ALGORITHMS	
Type: Integrated Professional Core Course	**Course Code: 21CS42**

No of Hours

Theory (Lecture Class)	Tutorials	Practical/Field Work/Allied Activities	Total/Week	Total hours of Pedagogy
4	0	3	7	40 T + 20 P

Marks

CIE	SEE	Total	Credits
50	50	100	4

Aim/Objectives of the Course

1. Explain the methods of analyzing the algorithms and to analyze performance of algorithms.
2. State algorithm’s efficiencies using asymptotic notations.
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.
4. Choose the appropriate data structure and algorithm design method for a specified application.
5. Introduce P and NP classes

Course Learning Outcomes

After completing the course, the students will be able to

CO1	Analyze the performance of the algorithms, state the efficiency using asymptotic notations and analyze mathematically the complexity of the algorithm.	**Applying (K3)**
CO2	Apply divide and conquer approaches and decrease and conquer approaches in solving the problems analyze the same	**Applying (K3)**
CO3	Apply the appropriate algorithmic design technique like greedy method, transform and conquer approaches and compare the efficiency of algorithms to solve the given problem.	**Applying (K3)**
CO4	Apply and analyze dynamic programming approaches to solve some problems. and improve an algorithm time efficiency by sacrificing space.	**Applying (K3)**
CO5	Apply and analyze backtracking, branch and bound methods and to describe P, NP and NP Complete problems.	**Applying (K3)**

Syllabus Content

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.	**CO1** 8 hrs PO1-3

Performance Analysis: Estimating Space complexity and Time complexity of algorithms. **Asymptotic Notations**: Big-Oh notation (O), Omega notation (Ω), Theta notation 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. **Laboratory Experiments:** 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. **LO:** At the end of this session the student will be able to 1. Understand what is algorithm. 2. Estimate Space complexity and Time complexity of algorithms. 3. Identify Asymptotic Notations.	PO2-3 PO3-3 PO4-3 PO6-1 PO7-1 PO12 -1 PSO1-1 PSO2-1

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. **Laboratory Experiments:** 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. **LO:** At the end of this session the student will be able to 1. Understand Divide and Conquer approach. 2. Understand Decrease and Conquer approach.	**CO2** 8 hrs. PO1-3 PO2-3 PO3-3 PO4-3 PO6-1 PO7-1 PO12-1 PSO1-1 PSO2-1
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.	**CO3** 8 hrs PO1-3 PO2-3 PO3-3 PO4-3

Laboratory Experiments: 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. **LO:** At the end of this session the student will be able to 1. Apply various Greedy methods. 2. Use Single Source shortest paths algorithm 3. Know about Heaps and Heap Sort	PO6-1 PO7-1 PO12-1 PSO1-1 PSO2-1
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. **Laboratory Experiments:** 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. **LO:** At the end of this session the student will be able to 1. Understand the Dynamic programming concepts and methods. 2. Solve all pair shortest paths using various algorithms. 3. Do String Matching using Harspool's algorithm.	**CO4** 8 hrs PO1-3 PO2-3 PO3-3 PO4-3 PO6-1 PO7-1 PO12-1 PSO1-1 PSO2-1
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. **Laboratory Experiments:** 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. **LO:** At the end of this session the student will be able to 1. Use backtracking method to solve many problems 2. Solve some problems using Branch and Bound 3. Identify NP-Complete and NP-Hard problems	**CO5** 8hrs PO1-3 PO2-3 PO3-3 PO4-3 PO6-1 PO7-1 PO12-1 PSO1-1 PSO2-1

Text Books

1. Introduction to the Design and Analysis of Algorithms, Anany Levitin: 2nd Edition, 2009. Pearson.
2. Computer Algorithms/C++, Ellis Horowitz, SatrajSahni and Rajasekaran, 2nd Edition, 2014, Universities Press.

Reference Books

1. Introduction to Algorithms, Thomas H. Cormen, Charles E. Leiserson, Ronal L. Rivest, Clifford Stein, 3rd Edition, PHI.
2. Design and Analysis of Algorithms, S. Sridhar, Oxford (Higher Education)

Useful Websites

- http://elearning.vtu.ac.in/econtent/courses/video/CSE/06CS43.html
- https://nptel.ac.in/courses/106/101/106101060/
- http://elearning.vtu.ac.in/econtent/courses/video/FEP/ADA.html
- http://cse01-iiith.vlabs.ac.in/
- http://openclassroom.stanford.edu/MainFolder/CoursePage.php?course=IntroToAlgorithms

Useful Journals

- IEEE TECHNOLOGY NAVIGATOR
- Journal of informatics and data mining
- Journal of computer and system sciences-Elsevier

Teaching and Learning Methods

1. Lecture class: 40 hrs
2. Tutorial classes: 23 hrs
3. Practical classes: 20hrs

Assessment

Type of test/examination: Written examination

Continuous Internal Evaluation(CIE) : 1) Three Tests each of 20 marks (duration 01 hour)
2) Two assignments each of 10 Marks
3) Practical Sessions for 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

Total CIE: 50 Marks

Semester End Exam (SEE) : 100 marks (students have to answer all main questions) which will be reduced to 50 Marks.

Test duration: 1 hr

Examination duration: 3 hrs

CO to PO Mapping

PO1: Science and engineering Knowledge **PO2:** Problem Analysis **PO3:** Design & Development **PO4:**Investigations of Complex Problems **PO5:** Modern Tool Usage **PO6:** Engineer & Society	**PO7:**Environment and Society **PO8:**Ethics **PO9:**Individual & Team Work **PO10:** Communication **PO11:**Project Mngmt & Finance **PO12:**Life long Learning

PSO1: An ability to design and develop Artificial Intelligence technology into innovative products for solving real world problems.

PSO2: An ability to design and develop Data Science methods for analyzing massive datasets to extract insights by applying AI as a tool.

CO	**PO**	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**	**PSO 1**	**PSO 2**
18CS 32	K-level														
CO1	K3	3	3	3	3	-	1	1	-	-	-	-	1	1	1
CO2	K3	3	3	3	3	-	1	1	-	-	-	-	1	1	1
CO3	K3	3	3	3	3	-	1	1	-	-	-	-	1	1	1
CO4	K3	3	3	3	3	-	1	1	-	-	-	-	1	1	1
CO5	K3	3	3	3	3	-	1	1	-	-	-	-	1	1	1

Course In charge **HOD-AI & DS** **IQAC Coordinator** **Principal**