|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Course Code | | | 21CSC204J | | **Course Name** | | | | | DESIGN AND ANALYSIS OF ALGORITHMS | | **Course Category** | | | *C* | | | *Professional Core* | | | | | | | | | | | | L | | T | | P | | C | | |
| 3 | | 0 | | 2 | | 4 | | |
| **Pre-requisite Courses** | | | | *Nil* | | | | | **Co-requisite Courses** | | *Nil* | | | | | | **Progressive Courses** | | | | |  | | | | | | | | | | | | | | | | |
| **Course Offering Department** | | | | | | | | *Data Science and Business Systems* | | | **Data Book / Codes/Standards** | | | | | | *Nil* | | | | | | | | | | | | | | | | | | | | | |
| **Course Learning Rationale (CLR):** | | | | | | *The purpose of learning this course is to:* | | | | | | |  |  | | **Program Outcomes (PO)** | | | | | | | | | | | | | | | | | | | | |
|  |  | |
| **CLR-1 :** | Design efficient algorithms in solving complex real time problems | | | | | | | | | | | |  |  | | 1 | | | 2 | 3 | 4 | | 5 | 6 | 7 | 8 | 9 | 10 | 11 | | 12 | 13 | 14 | | 15 | | |
| **CLR-2 :** | Analyze various algorithm design techniques to solve real time problems in polynomial time | | | | | | | | | | | |  |  | | Engineering Knowledge | | | Problem Analysis | Design & Development | Analysis, Design, Research | | Modern Tool Usage | Society & Culture | Environment & Sustainability | Ethics | Individual & Team Work | Communication | Project Mgt. & Finance | | Life Long Learning | PSO - 1 | PSO - 2 | | PSO – 3 | | |
| **CLR-3 :** | Utilize various approaches to solve greedy and dynamic algorithms | | | | | | | | | | | |  |  | |
| **CLR-4 :** | Utilize back tracking and branch and bound paradigms to solve exponential time problems | | | | | | | | | | | |  |  | |
| **CLR-5 :** | Analyze the need of approximation and randomization algorithms, utilize the importance Non polynomial algorithms | | | | | | | | | | | |  |  | |
|  | |  | | | | | | | | | | |  |  | |
| **Course Outcomes (CO):** | | | | | | | *At the end of this course, learners will be able to:* | | | | | | |  | |
| **CO-1 :** | Apply efficient algorithms to reduce space and time complexity of both recurrent and non-recurrent relations | | | | | | | | | | | | |  | | *2* | | | *1* | *2* | *1* | | *-* | *-* | *-* | *-* | *-* | *3* | *-* | | *3* | *1* | *1* | | *2* | | |
| **CO-2 :** | Solve problems using divide and conquer approaches | | | | | | | | | | | | |  | | *2* | | | *1* | *2* | *1* | | *-* | *-* | *-* | *-* | *-* | *3* | *-* | | *3* | *1* | *1* | | *2* | | |
| **CO-3 :** | Apply greedy and dynamic programming types techniques to solve polynomial time problems. | | | | | | | | | | | | |  | | *2* | | | *1* | *2* | *1* | | *-* | *-* | *-* | *-* | *-* | *3* | *-* | | *3* | *1* | *1* | | *2* | | |
| **CO-4 :** | Create exponential problems using backtracking and branch and bound approaches. | | | | | | | | | | | | |  | | *2* | | | *1* | *2* | *1* | | *-* | *-* | *-* | *-* | *-* | *3* | *-* | | *3* | *1* | *1* | | *2* | | |
| **CO-5 :** | Interpret various approximation algorithms and interpret solutions to evaluate P type, NP Type, NPC, NP Hard problems | | | | | | | | | | | | |  | | *2* | | | *1* | *2* | *1* | | *-* | *-* | *-* | *-* | *-* | *3* | *-* | | *3* | *1* | *1* | | *2* | | |

|  |  |  |
| --- | --- | --- |
| Unit-1 Introduction-Algorithm Design - Fundamentals of Algorithms- Correctness of algorithm - Time complexity analysis - Insertion sort-Line count, Operation count Algorithm Design paradigms - Designing an algorithm And its analysis-Best, Worst and Average case - Asymptotic notations Based on growth functions. O,O,Ө, ω, Ω - Mathematical analysis - Induction, Recurrence relations -Solution of recurrence relations - Substitution method - Solution of recurrence relations - Recursion tree - Solution of recurrence relations - examples. | | |
| Unit-2 Introduction-Divide and Conquer - Maximum Subarray Problem Binary Search - Complexity of binary search Merge sort - Time complexity analysis -Quick sort and its Time complexity analysis Best case, Worst case, Average case analysis - Strassen's Matrix multiplication and its recurrence relation - Time complexity analysis of Merge sort - Largest sub-array sum - Time complexity analysis of Largest sub- array sum - Master Theorem Proof - Master theorem examples - Finding Maximum and Minimum in an array - Time complexity analysis-Examples - Algorithm for finding closest pair problem - page104image807642688page104image807642880Convex Hull problem | | |
| Unit-3 Introduction-Greedy and Dynamic Programming - Examples of problems that can be solved by using greedy and dynamic approach Huffman coding using greedy approach Comparison of brute force and Huffman method of encoding - Knapsack problem using greedy approach Complexity derivation of knapsack using greedy - Tree traversals - Minimum spanning tree – greedy Kruskal's algorithm - greedy - Minimum spanning tree - Prims algorithm Introduction to dynamic programming - 0/1 knapsack problem - Complexity calculation of knapsack problem - Matrix chain multiplication using dynamic programming - Complexity of matrix chain multiplication - Longest common subsequence using dynamic programming - Explanation of LCS with an example - Optimal binary search tree (OBST)using dynamic programming - Explanation of OBST with an example. | | |
| Unit-4 Introduction to backtracking - branch and bound - N queen’s problem – backtracking - Sum of subsets using backtracking Complexity calculation of sum of subsets Graph introduction Hamiltonian circuit - backtracking - Branch and bound - Knapsack problem Example and complexity calculation. Differentiate with dynamic and greedy Travelling salesman problem using branch and bound - Travelling salesman problem using branch and bound example - Travelling salesman problem using branch and bound example - Time complexity calculation with an example - Graph algorithms - Depth first search and Breadth first search - Shortest path introduction - Floyd-Warshall Introduction - Floyd-Warshall with sample graph - Floyd-Warshall complexity | | |
| Unit-5. Introduction to randomized and approximation algorithm - Randomized hiring problem Randomized quick sort Complexity analysis String matching algorithm Examples - Rabin Karp algorithm for string matching Example discussion - Approximation algorithm - Vertex covering - Introduction Complexity classes - P type problems - Introduction to NP type problems - Hamiltonian cycle problem - NP complete problem introduction - Satisfiability problem - NP hard problems – Examples | | |
| Lab 1: Simple Algorithm-Insertion sort  Lab 2: Bubble Sort  Lab 3: Recurrence Type-Merge sort, Linear search  Lab 4: Quicksort, Binary search  Lab 5: Strassen Matrix multiplication  Lab 6: Finding Maximum and Minimum in an array, Convex Hull problem  Lab 7: Huffman coding, knapsack and using greedy  Lab 8: Various tree traversals,  Lab 9: Longest common subsequence  Lab 10: N queen’s problem  Lab 11: Travelling salesman problem  Lab 12: BFS and DFS implementation with array  Lab 13: Randomized quick sort  Lab 14: String matching algorithms  Lab 15: Discussion over analyzing a real time problem | | |
| Learning  Resources | 1. Thomas H Cormen, Charles E Leiserson, Ronald L Revest, Clifford Stein, Introduction to Algorithms, 3rd ed., The MIT Press Cambridge, 2014 2. Mark Allen Weiss, Data Structures and Algorithm Analysis in C, 2nd ed., Pearson Education, 2006 | 1. Ellis Horowitz, Sartajsahni, Sanguthevar, Rajesekaran, Fundamentals of Computer Algorithms, Galgotia Publication, 2010 2. S. Sridhar, Design and Analysis of Algorithms, Oxford University Press, 2015 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | |
|  | Bloom’s  Level of Thinking | **Continuous Learning Assessment (CLA)**  **- By the Course Faculty** | | | | **By The CoE** | |
| **Formative**  CLA-I Average of  unit test  (50%) | | **Life Long\***  **Learning**  CLA-II- Practice  (10%) | | **Summative**  Final  Examination  (40% weightage) | |
| Theory | Practice | Theory | Practice | Theory | Practice |
| Level 1 | Remember | 30 | - | - | 30 | 30 | - |
| Level 2 | Understand | 70 | - | - | 30 | 30 | - |
| Level 3 | Apply |  | - | - | 40 | 40 | - |
| Level 4 | Analyze |  | - | - |  |  | - |
| Level 5 | Evaluate | *-* | - | - |  |  | - |
| Level 6 | Create | - | - | - | - | - | - |
|  | Total | 100 % | | 100 % | | 100 % | |

|  |  |  |
| --- | --- | --- |
| Course Designers |  |  |
| Experts from Industry | Experts from Higher Technical Institutions | Internal Experts |
| G. Venkiteswaran, Wipro Technologies, gvenki@pilani.bits-pilani.ac.in | Mitesh Khapra, IITM Chennai, miteshk@cse.iitm.ac.in | 1. Dr. .K.Senthil Kumar, SRMIST |
| Dr.Sainarayanan Gopalakrishnan, HCL Technologies, sai.jgk@gmail.com | 2. V. Masilamani. IIITDM, masila@iiitdm.ac.in | 2. Dr. V. Sivakumar, SRMIST |
|  |  | 3. Dr. R.Vidhya,SRMIST |