|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Course code** | | | **ADVANCED ALGORITHMS** | | | | | | | **L** | **T** | **P** | **J** | | **C** |
| **CSI1005** | | |  | | | | | | | **2** | **0** | **2** | **0** | | **3** |
| **Pre-requisite** | | | **Data Structures and Algorithm Analysis** | | | | | | **Syllabus version** | | | | | | |
|  | | |  | | | | | | v. xx.xx | | | | | | |
| **Course Objectives:** | | | | | | | | | | | | | | | |
| 1. To focus on the design of algorithms in various domains 2. To provide a foundation for designing efficient algorithms. 3. To provide familiarity with main thrusts of work in algorithms- sufficient to give some context for formulating and seeking known solutions to an algorithmic problem. | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | |
| **Expected Course Outcome:** | | | | | | | | | | | | | | | |
| 1. To familiarize students with different algorithmic techniques 2. To introduce students to the advanced methods of designing and analyzing algorithms. 3. The student should be able to choose appropriate algorithms and use it for a specific problem. 4. Students should be able to understand different classes of problems concerning their computation difficulties. 5. Implement algorithm, compare their performance characteristics, and estimate their potential effectiveness in applications. | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | |
| **Student Learning Outcomes (SLO):** | | | | | **1,5,14** | | | | | | | | | | |
| 1. Having an ability to apply mathematics and science in engineering applications   5. Having design thinking capability  14. Having an ability to design and conduct experiments, as well as to analyze and interpret data | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | |
| **Module:1** | | **ALGORITHM DESIGN TECHNIQUES** | | | | | | **5 hours** | | | | | | **CO:**1 | |
| Revisit of Greedy algorithms, divide-conquer, dynamic programming.  Backtracking: General method, N-queen problem, Subset sum, Graph coloring, Hamiltonian cycles. Branch and Bound: General method, applications - Traveling sales person problem, 0/1 knapsack problem. LC Branch and Bound solution, FIFO Branch and Bound solution. | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | |
| **Module:2** | | **Network Flow** | | | | | | **4 hours** | | | | | | **CO: 1,**2 | |
| Flow Networks, Networks with multiple sources and sinks, Floyd-Warshall algorithm, Max Flow and Min Cut, Ford-Fulkerson Method and Edmonds-Karp Algorithm,  Bipartite Matching. | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | |
| **Module:3** | | **Computational Complexity** | | | | | | **5 hours** | | | | | | **CO:**2 | |
| Class complexity classes: P, NP, Reductions, NP-completeness and NP hard , NP-Complete Problems, CNF-SAT and 3SAT, Vertex-Cover and Clique | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | |
| **Module:4** | | **Randomized Algorithms** | | | | | | **3 hours** | | | | | | **CO:2,**3 | |
| Las Vegas algorithms, Randomized Quick Sort, Monte Carlo algorithms, Primality Testing | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | |
| **Module:5** | | **Approximation Algorithms** | | | | | | **4 hours** | | | | | | **CO:2,3** | |
| Limits to Approximability, Bin Packing (First fit, Best fit), (decreasing First fit, decreasing Best fit) 2 – Approximation algorithm for Metric TSP, Euclidean TSP, Max-SAT and Vertex Cover | | | | | | | | | | | | | | | |
| **Module:6** | | **Computational Geometry** | | | | | | **4 hours** | | | | | | **CO:3** | |
| Segment-intersection algorithm, Algorithms for finding convex hull: Graham’s scan,  Gift wrapping Algorithm. Finding the closest pair of points. | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | |
| **Module:7** | | **Algorithms for AI** | | | | | | **3 hours** | | | | | | **CO: 2** | |
| Uninformed search, Heuristic search (8 queen and tiling problems), A\* and AO\* algorithms. | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | |
| **Module:8** | | **Recent Trends** | | | | | | **2 hours** | | | | | | **CO:**3 | |
| Recent trends in advanced algorithms and design techniques | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | |
|  | | **Total Lecture hours:** | | | | | | **30 hours** | | | | | |  | |
| **Text Book(s)** | | | | | | | | | | | | | | | |
| 1.  2.  3. | T.H Cormen, C.E Leiserson, R.L Rivest, and C. Stein, ‘Introduction to algorithms’,3rd Edition, MIT Press, 2009.  S. Sridhar, ‘Design and Analysis of Algorithms’, Oxford University Press, 2015.  Ellis Horowitz, S. Sahni and S. Rajasekaran, ‘Fundamentals of Computer Algorithms’, 2nd edition, University Press, 2008. | | | | | | | | | | | | | | |
| **Reference Books** | | | | | | | | | | | | | | | |
| 1  2  3.  4. | Harsh Bhasin, ‘Algorithms: Design and Analysis’, Oxford University Press, 2015.  M.T.Goodrich and R.Tomassia, ‘Algorithm Design: Foundations, Analysis and Internet examples’ , John Wiley and sons, 2011.  Sara Baase, Allen, Van, Gelder, ‘Computer Algorithms, Introduction to Design and Analysis’, 3rd Edition, Pearson Education., 2003.  A.Levitin, ‘Introduction to the Design and Analysis of Algorithms’, Third Edition, Pearson Education, 2012. | | | | | | | | | | | | | | |
| Mode of Evaluation: CAT / Assignment / Quiz / FAT / Project / Seminar | | | | | | | | | | | | | | | |
| **List of Experiments (Indicative)** | | | | | | | | **CO:**5 | | | | | | | |
| 1. | Implementation of algorithms for problems that can be solved by one or more of the following strategies: Divide and Conquer, Brute force, Greedy, Dynamic Programming. Branch-and-Bound algorithm for the 0-1 Knapsack problem to maximize the profit for a given problem instance. | | | | | | | | | | | | | 6 hours | |
| 2. | Implementation of Graham’s scan and Gift wrapping algorithms. In addition to that, using the implementation compare the running time of both the algorithms empirically by taking large input size range. Finally, compare empirical analysis and theoretical time complexity of both the algorithms. | | | | | | | | | | | | | 4 hours | |
| 3. | Implementation of Ford-Fulkerson algorithm for computing a maximum flow in a network. | | | | | | | | | | | | | 2 hours | |
| 4. | Randomized Algorithms: Las Vegas and Monte Carlo algorithms | | | | | | | | | | | | | 2 hours | |
| 5. | Implementation of solution techniques for the minimum-cost flow problem. | | | | | | | | | | | | | 2 hours | |
| 6 | Heuristic search and A\*, AO\* algorithms | | | | | | | | | | | | | 2 hours | |
| 7 | Implementation of algorithms for Bin Packing, TSP, Vertex cover | | | | | | | | | | | | | 4 hours | |
| 8 | Implementation of search algorithms for graphs and trees: fundamental algorithms, Floyd Washall algorithm, Ford-Fulkerson Method and Edmonds-Karp Algorithm | | | | | | | | | | | | | 6 hours | |
| 9 | A simple polygon is defined as a flat shape consisting of straight non- intersecting line segments or sides that are joined pair –wise to from a closed path. Let P {p1, p2 , p3 ,....pn} be a set of points in the two dimensional plane.   1. Write a program to find the simple polygon of P . 2. Write a program (linear time) to convert that the simple polygon of P to a Convex Hull. | | | | | | | | | | | | | 2 hours | |
| Total Laboratory Hours | | | | | | | | | | | | | | 30 hours | |
| Mode of evaluation: Regular Assignments, Continuous Assessment Test / FAT (Lab) | | | | | | | | | | | | | | | |
| Recommended by Board of Studies | | | | DD-MM-YYYY | | | | | | | | | | | |
| Approved by Academic Council | | | | No. xx | | Date | DD-MM-YYYY | | | | | | | | |