|  |  |
| --- | --- |
| Design & Analysis of Algorithms | |
| **Course Code:** CS-302 | **Credit Hours:** 3.0 |
| **Introduction** | **Learning Outcomes** |
| The major objective of this course is providing comprehensive knowledge of modern computer algorithms and solving scientific and engineering problems efficiently and accurately. The students will be guided, how to analyze complex algorithms comparing efficiencies of these algorithms. Students will not only be taught the design of the existing algorithms but on the other hand it will be focused to teach them designing techniques using rigorous mathematical approaches. The students will be motivated to think about procedures solving real world problems optimally and correctly. Real world problem will be taken as examples to create feelings about the usefulness of this course. | Upon successful completion of this course, students should be able to:   * Argue and prove the correctness of algorithms using rigorous mathematical techniques taught in this course. * Analyze average and worst-case running times of given algorithm. * Describe the divide-and-conquer technique and arguing when an algorithmic design calls this approach. * Derive and solve recurrence relations describing performance of divide-and-conquer algorithms. * Describe advanced topics such as dynamic programming and greedy approach and reason to use these approaches for a particular situation. * Integrate dynamic programming and recursive approach improving efficiency of an algorithm. * Know the importance of graph theory in problem solving. * Employing graphs to model science and engineering problems, and to reason about when it is appropriate to use it optimally. * Analyze and design algorithms on further advanced topics such as computational geometry, operations research, cryptography, number theoretic, algorithms etc.   Analyze several other algorithms of importance such as string matching, NP completeness, approximation algorithms etc. |
| **Text and Reference Books** | **Assessments** |
| * Primary   + Introduction to the Design and Analysis of Algorithms (3rd ed.), Levitin   + Computers and Intractability, Guide to the Theory of NP-Completeness, Garey & Johnson * Reference   + Fundamentals of Algorithmics, Brassard and Bretly | |  |  | | --- | --- | | **Instrument** | **% of grade** | | Class Tasks (10) | 10% | | Quizzes (3) | 10% | | Assignments (5) | 10% | | Mini-Project + Presentation (1) | 05% | | Mid-Term (1) | 25% | | Final Exam (1) | 40% | |
| **Lesson Plan** | |
| |  |  |  | | --- | --- | --- | | Week# | No. of Lectures | Contents | | 1 | 3 | Introduction to Course, Mathematics, Computers & Algorithms, Basic Mathematical Tools | | 2 | 2 | Logic & Proving Techniques, Proofs, Validation, Verification | | 3 | 3 | Strong Mathematical Induction, Fibonacci Sequences, Recurrence Relations | | 4 | 3 | Mathematical Models & Analysis Techniques of Recurrence Relations, Algorithm Design & Analysis Techniques | | 5 | 3 | Further Techniques Solving Recurrence Relations, Time Complexity Measuring Notations, Relations over Asymptotic Notations | | 6 | 3 | Brute Force Approach: Introduction, Starting with Primality, Sorting Sequence of numbers, Designing Algorithms using Brute Force and Divide & Conquer Approaches, Designing Algorithms using Divide & Conquer Approaches | | 7 | 3 | Dynamic Programming for Solving Optimization Problems: Chain Matrix Multiplication Problem, Chain Matrix Multiplication Problem using Dynamic Programming | | 8 | 3 | Assembly-Line Scheduling Problem, 2-Line Assembly Scheduling Problem, n-Line Assembly Scheduling Problem, 0-1 Knapsack Problem using Dynamic Programming | | 9 | 3 | Optimal Weight Triangulation, Optimal Weight Triangulation using Dynamic Programming | | 10 | 3 | Longest Common Subsequence, Optimal Binary Search Trees, Optimal Binary Search Trees using Dynamic Programming, Greedy Algorithms | | 11 | 3 | Greedy Algorithms: Activity Selection Algorithm, Fractional Knapsack, Coin Change Making, Huffman Coding, Huffman Coding Problem, Road Trip Problem, Graph Theoretic Concepts | | 12 | 3 | Breadth First Search: Shortest Paths, Proof of Breadth First Search Algorithm, Depth First Search, Proof of White Path Theorem, Applications of Depth First Search | | 13 | 3 | Backtracking, Branch & Bound Algorithms, Minimal Spanning Tree Problem, Kruskal's Algorithm, Prim's Algorithm, Road Map Problem, Paths and Shortest Paths, Bellman-Ford Algorithm, Proof: Bellman Ford Algorithm, Shortest Paths in Directed Acyclic Graphs | | 14 | 3 | Dijkstra's Algorithm: Problem Statement, Analysis, Correctness, All-Pairs Shortest Paths, Shortest Paths and Matrix Multiplication, The Floyd-Warshall Algorithm, Johnson's Algorithm | | 15 | 3 | Number Theoretic Algorithms: Definitions and Some Important Results, GCD, Euclid's Algorithm, Groups and Rings, Chinese Remainder Theorem, RSA Cryptosystem, Fermat Theorem, Euler's Theorem, RSA Cryptosystem, String Matching Problem | | 16 | 3 | String Matching: Naive Algorithm, Rabin-Karp Algorithm, String Match with Finite Automata, Polynomials and Fast Fourier Transform: Representation of Polynomials, The DFT and FFT, NP Completeness: Circuit Satisfiability, Proof: Formula Satisfiability, 3-CNF, Clique | | |
| **Key Dates/Time/Venue** | |
|  | |