IT3215: DESIGN AND ANALYSIS OF ALGORITHMS

Course Prerequisites: Basic programming Skills, Data structures, Discrete Structures.

Course Objectives:

1. To understand asymptotic notations and apply suitable mathematical techniques to find asymptotic time and space complexities of algorithms.
2. To provide students with foundations to deal with a variety of computational problems using different design strategies.
3. To select appropriate algorithm design strategies to solve real world problems.
4. To understand notions of NP-hardness and NP-completeness and their relationship with the intractability of decision problems.
5. To understand randomized, approximation algorithms for given computational problems.

Cedits:5 **Teaching Scheme Theory:**3 Hours/Week
Tut: 1 Hour/Week
Lab:2 Hours/Week

Course Relevance: This is an important course for Information Technology Engineering. It develops algorithmic thinking capability of students. Designing algorithms using suitable paradigms and analyzing the algorithms for computational problems has a high relevance in all domains of IT (equally in Industry as well as research). Once the student gains expertise in Algorithm design and in general gains the ability of Algorithmic thinking, it facilitates systematic study of any other domain (in IT or otherwise) which demands logical thinking. This course is also relevant for students who want to pursue research careers in theory of computing, computational complexity theory, and advanced algorithmic research.

SECTION-1

Basic introduction to time and space complexity analysis: Asymptotic notations (Big Oh, small oh, Big Omega, Theta notations). Best case, average case, and worst-case time and space complexity of algorithms. Using Recurrence relations and Mathematical Induction to get asymptotic bounds on time complexity. Master's theorem and applications. Proving correctness of algorithms.

Divide and Conquer: General strategy, Binary search and applications, Analyzing Quick sort, Merge sort, Counting Inversions, finding a majority element, Josephus problem using recurrence, Efficient algorithms for Integer arithmetic (Euclid's algorithm, Karatsuba's

algorithm for integer multiplication, fast exponentiation).

Greedy strategy: General strategy, Analysis and correctness proof of minimum spanning tree and shortest path algorithms, fractional knapsack problem, Huffman coding,conflict free scheduling.

Dynamic Programming: General strategy, Principle of Optimality, simple dynamic programming-based algorithms to compute Fibonacci numbers, binomial coefficients, Matrix Chain multiplication, Optimal binary search tree (OBST) construction, Coin change problem, 0-1 Knapsack, Traveling Salesperson Problem, Bellman Ford shortest path algorithm, longest increasing subsequence problem, Largest independent set for trees.

SECTION-II

Backtracking strategy: General strategy, n-queen problem, backtracking strategy for some NP Complete problems (e.g., graph coloring, subset sum problem)

Branch and Bound strategy: Control abstraction for LIFO, Least Cost Search and FIFO branch and bound, 0-1 knapsack problem using LC branch and bound

Computational Complexity classes: Complexity classes P, NP, NP complete, NP Hard and their interrelation, Notion of polynomial time reduction, Cook-Levin theorem and implication to P versus NP question, Satisfiability Problem,NP-hardness of halting problem, NP-Complete problems (some selected examples).

Introduction to Randomized and Approximation algorithms: Introduction to randomness in computation, Las-Vegas and Monte-Carlo algorithms, Abundance of witnesses/solutions and application of randomization, randomized quick sort, Introduction to Approximation algorithms for NP-optimization problems (like Vertex Cover).

List of Practical: (Any Six)

1. Assignment based on some simple coding problems on numbers, graphs, matrices
2. Assignment based on analysis of quick sort (deterministic and randomized variant)
3. Assignment based on Divide and Conquer strategy (e.g. majority element search, finding

   kth rank element in an array)
4. Assignment based on Divide and Conquer strategy (e.g. efficient algorithm for Josephus

   problem using recurrence relations, fast modular exponentiation)
5. Assignment based on Dynamic Programming strategy (e.g. Matrix chain

multiplication,

Longest increasing subsequence)

6. Assignment based on Dynamic Programming strategy (e,g, All pair shortest path, Traveling Sales Person problem)

7. Assignment based on Greedy strategy (e.g. Huffman encoding, fractional knapsack problem)

8. Assignment based on Backtracking (e.g. graph coloring, n-queen problem)

9. Assignment based on Las-Vegas and Monte-Carlo algorithm for majority element search

10. Assignment based on factor-2 approximation algorithm for metric-TSP

List of Projects:

1.Applications of A* algorithm in gaming

2. Pac-Man game

3. Creation / Solution of Maze (comparing the backtracking-based solution and Dijkstra's algorithm)

4. Different exact and approximation algorithms for Travelling-Sales-Person Problem

5. Knight tour algorithms

6. Network flow optimization and maximum matching

7. AI for different games such as minesweeper, shooting games, Hex, connect-4, sokobanetc

7. SUDOKU solver

8. Algorithms for factoring large integers

10. Randomized algorithms for primality testing (Miller-Rabin, Solovay-Strassen)

11. Slider puzzle game

List of Course Seminar Topics:

1. Complexity classes
2. Space complexity
3. Divide and Conquer Vs Dynamic Programming
4. Greedy strategy Vs Backtracking strategy
5. Dynamic Programming Vs Greedy
6. Computational Complexity
7. Comparison of P Vs NP problems
8. Compression Techniques
9. Approximation algorithms
10. Pseudorandom number generators

List of Home Assignments:

Design:

1. Divide and Conquer strategy for real world problem solving
2. Dynamic Programming strategy for real world problem solving
3. Problems on Randomized Algorithms
4. Problems on Approximation Algorithms
5. Problems on NP completeness

Case Study:

1. Encoding techniques
2. Network flow optimization algorithms
3. Approximation algorithms for TSP
4. Sorting techniques
5. AKS primality test

Blog:

1. How to decide suitability of Approximation Algorithms
2. When do Randomized Algorithms perform best
3. Applications of Computational Geometry Algorithms
4. Role of number-theoretic algorithms in cryptography
5. Performance analysis of Graph Theoretic Algorithms

Surveys:

1. Primality Testing Algorithms
2. Integer Factoring Algorithms
3. Shortest Path Algorithms
4. Algorithms for finding Minimum Weight Spanning Tree
5. SAT solvers

Suggest an assessment Scheme:

1. Home Assignment
2. MSE & ESE
3. Seminar
4. LAB-Course Assignment and Project Evaluation

Text Books:

1. *Cormen, Leiserson, Rivest and Stein "Introduction to Algorithms", 3rd edition, 2009. ISBN 81-203-2141-3, PHI*
2. *Horowitz and Sahani, Fundamentals of computer Algorithms, Galgotia, ISBN 81-7371-612-9*
3. *Jon Kleinberg, Eva Tardos "Algorithm Design", 1st edition, 2005. ISBN 978-81-317-0310-6, Pearson*

4.Dasgupta, Papadimitriu, Vazirani "Algorithms", 1st edition (September 13, 2006), ISBN-

10:9780073523408, ISBN-13: 978-0073523408, McGraw-Hill Education

Reference Books:

1. *Anany Levitin, "Introduction to the Design & Analysis of Algorithm", Pearson, ISBN 81- 7758-835-4.*
2. *Gilles Brassard, Paul Bratle, Fundamentals of Algorithms, Pearson, ISBN 978-81-317-1244-3.*
3. *Motwani, Raghavan "Randomized Algorithms", 1st edition (August 25, 1995), ISBN-10:0521474655, ISBN-13: 978-0521474658, Cambridge University Press*

4.Vazirani, "Approximation Algorithms", ISBN-10: 3642084699, ISBN-13: 978-3642084690,Springer (December 8, 2010)

Moocs Links and additional reading material:

1. https://nptel.ac.in
2. https://www.udemy.com
3. https://www.coursera.org
4. https://www.geeksforgeeks.org

Course Outcome:

The student will be able –

1. To formulate computational problems mathematically.
2. To apply appropriate algorithmic paradigm to design efficient algorithms for computational problems.
3. To apply suitable mathematical techniques to analyze asymptotic complexity of the algorithm for a complex computational problem.
4. To understand the significance of NP--completeness of some decision problems and its relationship with intractability of the decision problems.
5. To understand significance of randomness, approximability in computation and design randomized and approximation algorithms for suitable problems.
6. To incorporate appropriate data structures, algorithmic paradigms to craft innovative scientific solutions for complex computing problems.