CS 403 Algorithm Design and Analysis

Credits	**:** 3-0-2-4	**Approval:**	Approved in 6th Senate
Prerequisites	**:** CS-3xx (ADSA) or equivalent, or the instructor's consent		
Distribution	**:** Discipline elective for CS and EE		
Intended for	**:** UG	**Semester:**	5th or 6th

Preamble: The proposed elective course, building on top of discipline core course on *Advanced Data Structures and Algorithms* (ADSA), offers first formal introduction to various common algorithm design techniques, methods for analyzing the performance of corresponding algorithms and improving their efficiency, and to provide performance guarantees. The theoretical aspects of this course are going to be supplemented by comprehensive practice exercises and weekly programming labs worth one lab credit.

Objective: After the students have gone through a course on discrete structures, where they learn formal and abstract representations of data and its manipulation, and another course on data structures, where they learn concrete implementations and usage of such discrete structures, a first course on algorithm design and analysis should teach the students how to design an efficient algorithm for a given computational task using one or more of such data structures, analyze performance of a given algorithm, and provide performance guarantees. On completion of such a course, students should be able to

- analyze the asymptotic performance of algorithms and write formal correctness proof for algorithms
- demonstrate their familiarity with major algorithm design paradigms and methods of analysis
- demonstrate their knowledge of major algorithms and data-structures corresponding to each algorithm design paradigm
- construct efficient algorithms for common computer engineering design problems
- classify a problem as computationally tractable or intractable, and discuss strategies to address intractability

Further, as programming is an integral part of the CS education, in this course students should implement the algorithms they learn and compare the corresponding achievable performance (computation time, memory requirement, etc.) with the corresponding asymptotic performance bounds they learn to compute in this course.

Syllabus:

1. Review of Data Structures. (3 L)

2. Program Performance: Time and space complexity, average and worst case analysis, asymptotic notation, recurrence equations and their solution. (3 L)

3. Algorithmic Techniques: Search techniques (backtracking and bounding), Sorting algorithms - lower bound, sorting in linear time, Greedy algorithms (Huffman coding, knapsack), Divide and conquer - Master theorem, Dynamic programming (0/1 knapsack, Traveling salesman problem, matrix multiplication, all-pairs shortest paths), Randomization, Randomized data structures: Skip Lists, Universal and perfect Hash functions, Backtracking, Branch and bound. (15 L)
   For each algorithm technique the following is expected: Description of the technique, explanation when an algorithm design situation requires it, examples of algorithms based on this technique, analysis of performance these algorithms.

4. Graph Algorithms: DFS and BFS, bio connectivity, spanning trees; Minimum cost spanning trees: Kruskal's, Prim's, and Sollin's algorithms; Path finding and shortest path algorithms; Topological sorting; Matching, Network Flows; Bipartite graphs. (6 L)

5. Computational complexity: Problem classes: P, NP, NP-complete, NP-hard. Reduction. Cook's theorem. Examples of NP-complete problems. (6 L)

6. Competitive analysis (3 L)

7. Amortized analysis: aggregate analysis, accounting, potential method. (3 L)

8. Other topics: Number theoretic algorithms (GCD, modulo arithmetic, Chinese remainder theorem), string matching algorithms (Rabin Karp algorithm, string matching with Finite State Automata, KMP (Knuth-Morris-Pratt) algorithm, Boyer-Moore algorithm), Strassen's matrix multiplication, FFT, integer and polynomial arithmetic. (3 L)

9. Advanced topics: Lower-bound techniques: adversary arguments, information-theoretic bounds.

Reference Books:

1. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, 3/e, 2009.
2. J. Kleinberg and E. Tardos, *Algorithm Design*, Pearson, 2006.
3. S. Dasgupta, C. H. Papadimitriou, U. V. Vazirani, *Algorithms*, McGraw-Hill, 2006.
4. S. S. Skiena, *The Algorithm Design Manual*, Springer, 2/e, 2008.