Course Number: CSC 510
Course Title: Analysis of Algorithms I
Number of Credits: 3
Schedule: Three hours of lecture/discussion per week.
Prerequisite: A grade of C or better in CSC313 or CSC340, and MATH 324.

Catalog Description
Notions of main algorithm design methodologies. Measures of algorithm complexity in space and time. Algorithms of classical problems such as sorting and scheduling and complexity analysis of such algorithms. Solving problems by determining and applying an appropriate designing methodology.

Expanded Description
The following topics will be covered:

- Algorithms: concepts and pseudo-code
- Recurrence relations: definition, application and approaches to solve them
- Notions of complexity order: definitions, applications and implications
- Complexity analysis from pseudo-code
- The sorting problem: common sorting algorithms, complexity analysis, and comparison
- The divide-and-conquer methodology: application in the sorting problem
- Dynamic programming: concepts, main implementation procedure, and applications
- The Binomial Coefficient problem: a dynamic programming solution
- The All-pairs Shortest Paths problem: Floyd-Warshall's dynamic programming solution
- The Traveling Salesperson Problem: optimal structure analysis and a dynamic programming solution
- The LCS (Longest Common Subsequence) Problem: optimal structure analysis and a dynamic programming solution
- The greedy approach: concepts, main implementation steps, and applications
- The Minimum Spanning Tree problem: Prim's and Kruskal's greedy algorithms
- The Single Source Shortest Paths problem: Dijkstra's greedy algorithm
- The Scheduling problem: different variants and their corresponding greedy algorithms
- The 0-1 Knapsack problem: a greedy algorithm
- Comparison of the dynamic programming and greedy approaches
- The backtracking optimization technique: concepts and applications
- The n-Queen problem: a solution with backtracking optimization
- The Sum-of-Subsets problem: a solution with backtracking optimization

Course Objectives and Role in Program
The students will learn:

- Designing algorithms to address classical problems, including Searching, Sorting, Scheduling, Minimum Spanning Tree, Graph Coloring, 0-1 Knapsack, Traveling Salesperson Problem, and more

- Methodologies of designing algorithms, including Divide-and-Conquer, Greedy approach, and Dynamic programming
- Techniques to analyze and compare the complexity of algorithms

The students will also be exposed to if time permits:

- Concepts of P, NP, NP-hard, NP-complete, and Theory of NP
- Parallel and distributed algorithms
- Algorithms in several main application domains such as bioinformatics or life sciences in general

Learning Outcomes

At the end of this course students will:

- Understand and be able to apply the three main algorithm designing techniques: divide-and-conquer, dynamic programming, and backtracking optimization
- Grasp the essence of the algorithms implemented to solve a list of classic problems, such as sorting, shortest paths, and scheduling
- Learn how to analyze the complexity of algorithms
- Be able to implement an algorithm to solve problem using an appropriate designing method and analyze the complexity of such algorithm

Method of Evaluation

Student learning will be evaluated on the basis of

- Completeness and quality of homework assignments.
- Grades of midterm examinations
- Grade of final examination
- Grades of in-class quizzes
- Class participation

The weight assigned to each element of evaluation will be determined by the instructor on the first day of the class.

Required Textbooks

Foundations of Algorithms Using C++ Pseudocode (Third Edition), by Richard Neapolitan and Kummars Naimipour, Jones and Bartlett Computer Science, ISBN: 0-7637-2387-8

Recommended Reference

Introduction to Algorithms, Second Edition (Second Edition) by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, MIT Press, ISBN: 0-2620-3293-7

Modified by: H. Yang
Last Revision Approved: 8/2011