Course Title: COSC 3351: Algorithms

Semester Credit Hours: 3 (3,0)

I. Course Overview

This course is the study of the design and performance analysis of algorithms. Time and space complexity analysis of algorithms, design paradigms, and graph algorithms are discussed.

II. PMU Competencies and Learning Outcomes

Students of COSC 3351: Algorithms develop quantitative and programming skills necessary for continued success in computer science. The skills enhance their abilities to devise, analyze, and comprehend mathematically the performance characteristics of algorithms and various design paradigms common to computer science as a discipline and to effectively communicate their solutions to fellow professionals. This course makes extensive use of the PMU technology infrastructure to provide communication between faculty and students. The course includes individual as well as group projects, establishes both mathematical reasoning skills and technical communication skills, and provides opportunities for the presentation and defense of designed solutions.

III. Detailed Course Description

This course is concerned with the study of the design and performance analysis of algorithms, including maximum contiguous subarray, divide-and-conquer algorithms, graph algorithms, dynamic programming, and greedy algorithms. One important lasting effect of this course is to enhance and develop the ability to specify, design, implement, test, and analyze solutions to programming problems utilizing the proven design paradigms presented in this course.

IV. Requirements Fulfilled

COSC 3351: Algorithms satisfies 3 hours of the requirements for the degree in computer science. It is required of all students pursuing a degree program in computer science within the College of Information Technology. It should be taken in the second semester of the junior year.

V. Required Prerequisites

- COSC 3421: Data Structures
- MATH 1313: Statistical Methods

VI. Learning Outcomes

In this course, students learn:

- To understand and apply performance analysis techniques to analyze maximum contiguous subarray.
- To understand and analyze the performance of divide-and-conquer algorithms.
- To understand and analyze the performance of graph algorithms.
- To understand and analyze the performance of dynamic programming.
- To understand and analyze the performance of greedy algorithms.
- To understand the concepts of complexity classes.
- To use mathematical and measurement techniques to analyze the performance characteristics of algorithms.
- To understand and apply techniques of algorithm analysis and express performance characteristics of algorithms.
- To develop improved communication and collaborative skills.

VII. Assessment Strategy

This course is designed with three primary goals in mind: to introduce students to the conceptual basis and practical issues associated with the use, development, and analysis of designed algorithms, to lead students to connect the mathematics to its application in computer science, and to provide students with the opportunity to communicate their designs and implementations to their peers in a professional setting. With this in mind, the course grade involves an assessment of their performance on examinations that focus on the application of programming paradigms to the solutions of problems, the performance analysis of the designed solutions, and the communication of designed solutions to those problems to an audience. Course grades are based on:

- Weekly assigned homework to motivate students to do the work and earn credit accordingly.
- Weekly, in-class presentations by students of solutions to real world problems related to the course material and classroom discussion and critique of the presentation.
- Two in-class examinations to assess the student's accumulative mastery of content covered prior to the time of the examination.
- Three programming assignments testing students understanding of the major concepts introduced during the course.
- A comprehensive final examination to assess the student's accumulative mastery of course material.

The final grade is based on 10% credit for the homework, 10% for the presentations and participation in classroom discussion, 30% on in-class examinations, 30% on programming assignments and 20% for the final examination.

Students are required to maintain a journal of thoughts and commentaries during the course. The journal contains daily entries including the identification of areas of interest and concern, notes on the preparation of presentation and comments and analysis of classmate's presentations. The journal is reviewed weekly by the instructor to provide feedback to the students.

Final grades and the student and instructor observations from reflective notebooks are included in the student's portfolio for use in the final assessment capstone course. The intent is to document the student's maturation as he proceeds through the curriculum.

VIII. Course Format

A. Instruction

This course utilizes both lecture/discussion and laboratory exercises. Students are expected to attend three hours of lecture per week. At least once per week students should be prepared to make presentation on the design and implementation of a solution to a problem selected by the instructor and to take part in a discussion based on that presentation. Once a week students should have at least 30 minutes of collaborative problem solving activity.

B. Web supplement

Course home page (the university's Web tool, WebCT or Blackboard) should contain the following:

- Course syllabus
- Course assignments
- Sample solutions to examinations (after being graded and returned)
- Sample solutions to programming assignments (after being graded and returned)
- Course calendar (an active utility)
- Course e-mail (an active utility)
- Course discussion list (an active utility)
- Student course performance (an active utility)

IX. Topics to be Covered

A. Case study in algorithm design
   maximum contiguous subarray
B. Divide-and-conquer algorithms
   1. Merge sort
   2. Polynomial multiplication
   3. Randomized selection
C. Graphs
   1. Notations
   2. Breadth-first search and depth-first search
   3. Cycle finding and topological sort
   4. Maximum spanning trees
   5. Dijkstra's shortest path algorithm

D. Dynamic programming
    1. Knapsack
    2. Chain matrix multiplication
    3. Longest common subsequence
    4. All pairs shortest path
E. Greedy algorithms
    1. Activity selection
    2. Huffman coding
F. Complexity classes
    1. Nondeterminism
    2. Classes P and NP
    3. NP-complete problems
    4. Polynomial reductions

X. Laboratory Exercises

This course does not offer a separate laboratory to students.

XI. Technology Component

This course makes use of the university's wireless access infrastructure. The course relies on the university and the students having access to professional grade application development environments for the students to use.

XII. Special Projects/Activities

Students are required to keep a "reflective notebook" in which, after each class, they enter their own assessments of what they learned, and what questions remain from the class. From each exercise set, each student selects one problem, which the student thinks best reflects the way the topic is used in a technical context. A detailed solution to the problem is included in the student's reflective notebook.

XIII. Textbooks and Teaching Aids

A. Required Textbook

Cormen, T., C., R. Rivest Leiserson, and C. Stein. *Introduction to Algorithms*. _____: The MIT Press, 2001.
ISBN 0-262-03293-7

B. Alternative Textbooks

1. Bentley, J. *Programming Pearls*. _____: Addison-Wesley, 2000.
   ISBN 0-201-65788-0
2. Garey, M.R., and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, 1979.
   ISBN 0-7167-1045-5

C. Supplemental Print Materials

None.

D. Supplemental Online Materials

As available from publishers.