Course Title: COSC 3421: Data Structures

Semester Credit Hours: 4 (3,1)

I. Course Overview

Data structures is the systematic study of some advanced data structures, including list, stack, queue, dictionary, and graph. Sorting and hashing algorithms and their associated computational costs are discussed. Algorithm analysis techniques are also investigated to provide a metric to measure the performance of an algorithm in question.

II. PMU Competencies and Learning Outcomes

Students in this course develop programming and quantitative 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 data structures 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 a structured laboratory component to ensure that students gain the necessary experience and skill in managing the concepts introduced in the class. 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

COSC 3421: Data Structures is concerned with the systematic study of some advanced data structures, including list, stack, queues, dictionary, graphs. Sorting and hashing algorithms and their associated computational costs are discussed. The course presents the students with the concepts of asymptotic notations, performance measurement, sorting and searching including algorithms and lower bounds, abstract data types and classes, data structures such as heaps, search trees, tries, and hashing, and graphs: representation, depth-first-search, and breadth-first search. 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 data structures and proven algorithms presented in this course.

IV. Requirements Fulfilled

This course satisfies four 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 first semester of the junior year.

V. Required Prerequisites

- GEIT 1412: Computer Science II
- MATH 1313: Statistical Methods
- MATH 2332: Differential Equations

VI. Learning Outcomes

In this course, students learn:

- To understand and use of data structures, including list, stack, queue, dictionary, and graphs.
- To understand and use of sorting and hashing algorithms.
- Learn to use mathematical and measurement techniques to analyze the performance characteristics of algorithms.
- To be able to discuss the advantages and disadvantages of different implementations of each of the data structures.
- 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 data structures and 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 techniques 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.
- Weekly structured laboratory exercises designed to guide students through specific course topics.
- Two in-class examinations to assess the student's accumulative mastery of content covered prior to the time of the examination.
- Five 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, 20% for weekly laboratory exercises, 20% on in-class examinations, 30% on programming assignments and 10% 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/discussion per week and two hours of laboratory 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)

Classroom Hours (5 hours per week) **Class:** 3
Lab: 2

IX. Topics to be Covered

A. Basic data structures
   1. List
   2. Stack
   3. Queue

B. Analysis of algorithms
   1. Time complexity
   2. Upper and lower bounds

C. Sorting
   1. Insertion sort
   2. Merge sort
   3. Quick sort
   4. Heap sort
   5. Performance bounds

D. Dictionary
   1. Binary search trees
   2. AVL tree
   3. B tree

E. Graph
   1. Definition, representation, and modeling tool
   2. Breath-first search
   3. Depth-first search
   4. Directed graph and topological sort

F. Hashing
   1. Hash tables and functions
   2. Collision resolution

X. Laboratory Exercises

This course requires a weekly two-hour laboratory component. Topics to be covered in the laboratory sessions should include:

- Basic data structures – review exercises in basic data structures.
- Time complexity – exercises in estimating time complexity of algorithms by measurement.
- Performance bounds – exercises in estimating performance bounds of algorithms by measurement.
- Sorting I – exercises in the use and analyzing of sorting algorithms including merge sort and quick sort.
- Sorting II – additional exercises in the use and analyzing of sorting algorithms including heap sort.
- Dictionary I – exercises in the use of dictionary as an abstract data type including binary search tree and AVL tree.
- Dictionary II – additional exercises in the use of dictionary including B tree.
- Breath-first search – exercises in the use of breath-first search algorithm.
- Depth-first search – exercises in the use of depth-first search algorithm.
- Topological sort – exercises in the use of topological sort algorithm

- Hashing I – exercises in the implementation of hash tables and functions.
- Hashing II – exercises in the implementation of collision resolution.
- Three additional lab sessions should be kept in reserve to allow the instructor to extend the more difficult laboratories for more than one session.

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. The course has a laboratory component that would be best implemented in university provided laboratory space.

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

Weiss, M. A. *Data Structures and Algorithm Analysis in C++*. _____: Addison-Wesley, 1999.
ISBN 0-201-36122-1

B. Alternative Textbooks

Horowitz, E., S. Sahni, and D. Mehta. *Fundamentals of Data Structures in C++*. _____: W. H. Freeman, 1995.
ISBN 0-7167-8296-0

C. Supplemental Print Materials

None.

D. Supplemental Online Materials

As available from publishers.