TCSS 342 - Data Structures

Course Syllabus - Spring 2022

Instructor: Dr. Chris Marriott
Preferred: “Chris” or “Dr. Marriott” or “Professor Marriott”

Office: CP 123
Office Hours: Tuesday, 1pm-3pm, (via Discord)
or by appointment.

Email: cmarriot@uw.edu

Lecture: via YouTube
Policies: UWT E-Syllabus

Distance Learning: This course will operate asynchronously online. This means you will not be expected to attend the scheduled meetings. Nonetheless, all lecture slides and videos, homework and quizzes will be distributed through Canvas.

Description: Covers abstract data types, design and complexity analysis of data, and usage of generic structure libraries in high-level programming languages. Includes sequential and linked lists, heaps, binary search trees and balanced binary search trees, B-trees, hash tables, and graphs.
Prerequisites: A minimum grade of 2.0 in TCSS 321.
Student Learning Goals:

- Define what a data abstraction is.
- Identify the operations that characterize the following abstractions and structures.
    - Data abstractions: List, Stack, Queue, Priority Queue, Map/Dictionary, Graph
    - Data structures: Array, Linked List, Tree, Binary Tree, Binary Search Tree (BST), Balanced BST, B-Tree, Heap, Hash Table, Graph
- Identify the API of each abstraction (or its closest correlates) in a standard generic library of a modern programming language, such as the Java Collections Framework.
- Apply pictorial representations to explain the implementations of the data structures and their operation.
- Adapt and/or extend the generic code implementation of at least 3 data structures from the canonical set, at least 1 of which is recursively defined.
- Instantiate and apply the API’s of at least 4 data abstractions (or their correlates) in the canonical set as provided in the standard generic library of a modern programming language.
- Analyze the worst- and average-case time and space complexity of the operations in common implementations of the data abstractions and structures in the canonical set.
- Analyze the worst-case time complexity of code that uses data abstractions in the canonical set.
- Compare and contrast time/space characteristics of different implementations of the same data abstraction.
- Select data abstractions, structures, and implementations that a developer would use in solving

larger problems and defend the appropriateness of these choices.

Program Outcomes:

- An ability to apply knowledge of computing and mathematics appropriate to the discipline.
- An ability to analyze a problem, identify and define the computing requirements appropriate to its solution.
- An ability to design, implement and evaluate a computer-based system, process, component, or program to meet desired needs.
- An ability to use current techniques, skills, and tools necessary for computing practice.
- An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.

UWT Student Learning Goals:

- Inquiry and Critical Thinking - Students will acquire skills and familiarity with modes of inquiry and examination from diverse disciplinary perspectives, enabling them to access, interpret, analyze, quantitatively reason, and synthesize information critically.

(Recommended) Textbook: *Introduction to Algorithms*, Cormen, Leiserson, Rivest and Stein. 3rd Ed.

Online materials: This syllabus, lecture slides and videos, homework assignments, quizzes, discussion board and other supplemental materials are available via the Canvas course management system.

Grading: Grades will be assigned according to the labor-based grading contract. Please take the time to read it and the associated documents carefully.