CS310 Data Structures Spring 2015

Credits: 3 units
Instructor: Alan Riggins Office: GMCS 534
Office Hours: 9:00-12:00 M/W/F.

Course Materials


Course Information

Prerequisites: Computer Science 108 and Mathematics 245.

Description from the Official Course Catalog

Representations and operations on basic data structures. Arrays, linked lists, stacks, queues, and recursion; binary search trees and balanced trees; hash tables, dynamic storage management; introduction to graphs. An object oriented programming language will be used.

Course Type: Required upper division core curriculum course.

Specific Goals for CS310

Course-Level Student Learning Outcomes

1. Students should gain a thorough understanding of the classical data structures.
2. Students should gain a thorough understanding of the classical ADT (abstract data types).
3. Students should advance in their programming abilities through extensive individual programming assignments that include writing data structure/ADT implementations from scratch.
4. Ability to write generic, reusable code, and apply software design patterns.
5. Ability to understand, write and modify recursive algorithms to solve programming problems.
6. Mature in their ability to identify and repair software bugs.
7. Ability to analyze program requirements and select the optimal implementation.
8. Ability to perform rigorous algorithm complexity analysis on a wide range of algorithms and data structure methods.
9. Ability to analyze a variety of software problems and provide appropriate solutions (what to use when) including written reports with analysis of
implementation options and recommendations for an optimal solution, which includes a rigorous complexity analysis.

**Relationship to CS Program Course Outcomes**

**CS310 addresses the following outcomes:**

1. An ability to apply knowledge of computing and mathematics, including basic calculus.
2. An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.
3. An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.
4. An ability to write on technical topics clearly and effectively.
5. Recognition of the need for and an ability to engage in continuing professional development
6. An ability to use current techniques, skills, and tools necessary for computing practice
7. 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.
8. An ability to apply design and development principles in the construction of software systems of varying complexity.

**Topics Covered**

1. Classic data structures: array, linked list, max/min heap, binary search tree, balanced trees, n-ary trees, graphs.
2. Classical ADTs (abstract data types): stack, queue, list, set, priority queue, heap, dictionary, map.
3. Advanced object oriented programming: Including abstract classes and interfaces, inheritance and composition, design patterns, and generic programming.
4. Recursion: Binary search, and other algorithms to support data structure functionality.
5. Complexity analysis: Runtime complexity analysis tools and techniques.
Schedule and Grading Policies

<table>
<thead>
<tr>
<th>Week</th>
<th>Topic</th>
<th>Projects/Activities</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Course introduction, motivation for studying data structures and ADTs</td>
<td></td>
</tr>
<tr>
<td>2-3</td>
<td>Complexity Analysis</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Arrays, stacks, queues.</td>
<td>Midterm #1, Project #1</td>
</tr>
<tr>
<td>5</td>
<td>Inheritance, generic programming, parameterized types, reusable code, building ADTs from existing data structures.</td>
<td></td>
</tr>
<tr>
<td>6-8</td>
<td>Linked lists and other linked structures. Ordered, unordered, self-organizing, and skip lists. Binary heaps.</td>
<td>Project #2</td>
</tr>
<tr>
<td>9-10</td>
<td>Sorting algorithms.</td>
<td>Project #3</td>
</tr>
<tr>
<td>11</td>
<td>Maps, dictionaries. Hash tables and hash algorithms.</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Binary search trees.</td>
<td></td>
</tr>
<tr>
<td>13-14</td>
<td>Balanced trees, AVL, red/black, n-ary trees, B Trees</td>
<td>Project #4</td>
</tr>
<tr>
<td>15</td>
<td>Graphs</td>
<td>Midterm #2</td>
</tr>
</tbody>
</table>

Projects/Assignments

There are three to four programming assignments in the Java programming language:

- Linked list based project.
- Array based project (ADT built using an array)
- Map project implementing a binary search tree, a hash table, and a balanced tree.

Students will write all code for these assignments rather than using existing libraries from the java API with the exception of the final project, when techniques for using the API are covered for the balanced tree implementation.

Grading Policy

Grading is based on exams (60%), programming assignments (40%). There are four programming assignments, two midterm exams, and a final exam. Grades are not curved, but based on percentages:
93% and above A
90%-92% A-
87%-89% B+
83%-86% B
80%-82% B-
77%-79% C+
73%-76% C
70%-72% C-
67%-69% D+
63%-66% D
60%-62% D-
below 60% F

**Students with Disabilities**

If you are a student with a disability and believe you will need accommodations for this class, it is your responsibility to contact Student Disability Services at 619 594-6473. To avoid any delay in the receipt of your accommodations, you should contact Student Disability Services as soon as possible. Please note that accommodations are not retroactive, and that accommodations based upon disability cannot be provided until you have presented your instructor with an accommodation letter from Student Disability Services. Your cooperation is appreciated.