Course: CS 340 Data Structure and Algorithms

Catalog Description:
Appropriate choice of data structures, comparisons of algorithms, and complexity. Topics include lists, stacks, queues, binary search trees, AVL trees, Splay trees, B-trees, Hashing, Heaps, Sorting, Disjoint set class, Graph algorithms, computational complexity, and parallel algorithm.

Principal Topics:
- Understand what sequential, parallel, and distributed algorithms are
- Explain what greedy algorithms, dynamic programming and divide and conquer algorithms are
- Determine the best, average and worst case behaviors of an algorithm
- Compare and contrast multiple algorithms to solve the same problem
- Select the most efficient data structure and algorithm for performing a task such as sorting
- Implement various algorithms and data structures
- Apply the knowledge of graphs in network modeling and simulation
- Understand the NP-complete problem

Assignments:
- Programming assignments covering AVL trees, Min-Max Heaps, and Shortest path algorithms.
- In-class exercises and quizzes
- Two regular exams and one comprehensive final exam
- Homework including must-answer questions and bonus questions.

Outcome C:
An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs

<table>
<thead>
<tr>
<th>Performance Criteria</th>
<th>Evaluation Level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 - Unacceptable</td>
</tr>
<tr>
<td></td>
<td>2 - Developing</td>
</tr>
<tr>
<td></td>
<td>3 - Competent</td>
</tr>
<tr>
<td></td>
<td>4 - Exemplary</td>
</tr>
<tr>
<td>The student demonstrates comprehension and mastery of algorithm design techniques</td>
<td>The student cannot understand the algorithm design techniques and is not able to apply the knowledge in solving problems similar to cases discussed in class</td>
</tr>
<tr>
<td>including recursive, divide-and-conquer, greedy, and dynamic programming strategies.</td>
<td></td>
</tr>
<tr>
<td>Possible Measurement Mechanisms:</td>
<td>Homework</td>
</tr>
<tr>
<td></td>
<td>Quizzes</td>
</tr>
<tr>
<td></td>
<td>Exams</td>
</tr>
</tbody>
</table>
The student can design and implement programs using appropriate and/or specified data structures and/or algorithms.

The student cannot design and implement programs for solving problems similar to examples discussed in class.

The student can design and implement programs but fails in using specified data structures and/or algorithms.

The student can design and implement programs, using specified data structures and algorithms.

**Possible Measurement Mechanisms:**
- Homework
- Programming assignments

The student can evaluate the best case, average case, and the worst case time complexity for given programs.

The student cannot evaluate the best case, average case, or the worst case time complexity for programs similar to in-class examples.

The student can evaluate the best case and the worst case time complexity for programs similar to in-class examples but fails to identify the average case time complexity.

The student demonstrates solid mastery of time complexity analysis for programs similar to in-class examples and is able to apply the knowledge to new but related problems.

**Possible Measurement Mechanisms:**
- Homework
- Quizzes and exams

**Outcome J:**
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.

<table>
<thead>
<tr>
<th>Performance Criteria</th>
<th>Evaluation Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Demonstrates comprehension of big Oh, big theta, and big Omega notations in algorithm analysis, and can establish relative order among functions</td>
<td>1 - Unacceptable</td>
</tr>
<tr>
<td>Displays little to no understanding of big Oh, big theta, and big Omega notations in algorithm analysis and cannot establish relative order among basic functions.</td>
<td>Displays adequate understanding of big Oh, big theta, and big Omega notations in algorithm analysis and can only establish relative order among basic functions.</td>
</tr>
</tbody>
</table>

**Possible Measurement Mechanisms:**
- Homework
- Quizzes and exams
| Demonstrates capability in the modeling and design of algorithms using graph theory for given applications | Demonstrates little to no understanding in graph theory such as graph representation or its basic applications such as topological sort and depth-first search. | Demonstrates adequate understanding in graph theory and its basic applications, but fails to use it in complex applications including shortest path routing, network flow problems, and minimum spanning tree. | Demonstrates good understanding in graph theory and its basic and complex applications, as well as advanced applications including finding articulation points, and strong components. | Demonstrates solid understanding in graph theory and its basic, complex, and advanced applications, and can apply the knowledge to research problems and/or new applications |
| Possible Measurement Mechanisms: | • Homework  
• Quizzes and exams  
• Programming assignments |  |  |  |
| Demonstrates comprehension of the tradeoffs for algorithms employing different data structures and/or design strategies for given problems | Demonstrates little to no understanding in the tradeoffs for algorithms employing different data structures and/or design strategies for sorting problems. | Demonstrates basic understanding of the tradeoffs for algorithms employing different data structures and/or design strategies for sorting problems, but fails for graph algorithms. | Discusses tradeoffs present in sorting algorithms and graph algorithms employing different data structure and design strategies. | Discusses tradeoffs present in sorting and graph algorithms employing different data structure and/or design strategies, and can choose approximate data structure/algorithms for given problems |
| Possible Measurement Mechanisms: | • Homework  
• Quizzes and exams  
• Programming algorithms |  |  |  |