School of Computing
UNIVERSITY OF GEORGIA

Course Information Sheet

CSCI 4470

Algorithms

Brief Course Description (50-words or less)	Algorithms, covering basic analysis techniques, basic design techniques (divide-and-conquer, dynamic programming, greedy), basic and advanced graph algorithms, and NP-completeness theory.
Extended Course Description / Comments	N/A
Pre-Requisites and/or Co-Requisites	1) CSCI 2720 (Data Structures) OR CSCI 2725 (Data Structures for Data Science) AND 2) CSCI 2670: Introduction to Theory of Computation
Required, Elective or Selected Elective	Required
Approved Textbooks (if more than one listed, the textbook used is up to the instructor's discretion)	Author(s): Cormen, Leiserson, Rivest, and Stein Title: *Introduction to Algorithms* Edition: 4th ISBN-13: 9780262046305
Specific Learning Outcomes (Performance Indicators) (Approximate Course Hours)	This course provides an introduction to the modern study of computer algorithms most relevant to students studying computer engineering. At the end of the semester, all students will be able to do the following: 1. Analyze time complexity for algorithms using asymptotic notation. (4 hours) 2. Apply various sorting and selection algorithms and determine which algorithm to apply for different types of data. (9 hours) 3. Apply advanced design and analysis techniques, including greedy and dynamic programming techniques. (10 hours) 4. Analyze and prove properties for various graph algorithms. (20 hours) 5. Choose (and/or apply) the appropriate algorithm to solve real-world problems and defend the selection (integrated into the above outcomes) 6. Describe decision problems and prove problems are in P. (1 hour) 7. Prove a problem is in NP, co-NP, or NP-complete. (4 hours)

ABET Learning Outcomes

Graduates of the program will have an ability to:

A. Analyze a complex computing problem and to apply principles of computing and other relevant disciplines to identify solutions.
B. Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline.
C. Communicate effectively in a variety of professional contexts.
D. Recognize professional responsibilities and make informed judgments in computing practice based on legal and ethical principles.
E. Function effectively as a member or leader of a team engaged in activities appropriate to the program's discipline.
F. Apply computer science theory and software development fundamentals to produce computing-based solutions.

NOTE: In the construction of the student learning outcomes for this course, the instructors interpreted "computing requirements" in (B) as the functional requirements for a software solution and not as specific hardware requirements for the target platform; likewise, the phrase "[a]pply computer science theory" in (F) was interpreted as using computer science principles.

Relationship Between Student Outcomes and Learning Outcomes

		ABET Learning Outcomes					
Specific Learning Outcomes		A	B	C	D	E	F
	1	●	●				●
	2	●	●				●
	3	●	●				●
	4	●	●				●
	5	●	●				●
	6	●					●
	7	●					●

Extended Learning Outcomes

1. Analyze complexity of algorithms (Knowledge level: Assessment)
    a. Analysis of recursive algorithms using recursion trees
    b. Analysis of recursive algorithms using substitution method
    c. Analysis of recursive algorithms using Master theorem
2. Apply various sorting and selection algorithms and determine which algorithm to apply for different types of data.
    a. Provide runtime analysis of various sorting algorithm including heap-sort, quicksort, counting sort, and radix sort.
    b. For a given type of input data, determine the best choice of sorting algorithm
    c. Prove the O(n lg n) lower bound for comparison based sorting algorithms.
3. Advanced design and analysis techniques, including greedy algorithms and dynamic programming (Knowledge level: Assessment)
    a. Design and prove the optimality of greedy algorithms.
    b. Design dynamic programming algorithms, including solution recurrence formulation, iterative, bottom-up design, and development of trace-back subroutines.
4. Analyze and prove properties for various graph algorithms. (Knowledge level: Assessment)
    a. Depth-first search and breadth-first search
    b. Minimum spanning tree
    c. Single-source shortest path and all-pairs shortest path
    d. Network flow – max flow / min cut
5. Choose (and/or apply) the appropriate algorithm to solve real-world problems and defend the selection (Knowledge level: Assessment)
    a. Select between various sorting algorithm for different data scenarios and justify selection
    b. Select between using a greedy strategy or dynamic programming for optimization problems
    c. Select appropriate graph algorithm for a given graph-related question
    d. Derive a graphical representation for a real-world problem and solve the problem using graph properties.
6. Describe decision problems and prove problems are in P. (Knowledge level: Usage)
    a. Describe decision problems
    b. Prove a problem is in P by providing a deterministic decider
7. Prove a problem is in NP, co-NP, or NP-complete. (Knowledge level: varies)
    a. Prove a problem is in NP by providing a verifier or a non-deterministic decider. (Knowledge level: Usage)
    b. Prove a problem is NP-Complete by providing a polynomial time reduction from a known NP-Complete problem. (Knowledge level: Usage)
    c. Describe the Cook-Levin Theorem proving SAT is NP-Complete (Knowledge level: Familiarity)

Knowledge Levels

The following is the ACM's categorization of different levels of mastery: Assessment, Usage, and Familiarity. Note that Assessment encompasses both Usage and Familiarity, and Usage encompasses Familiarity.

Familiarity: The student understands what a concept is or what it means. This level of mastery concerns a basic awareness of a concept as opposed to expecting real facility with its application. It provides an answer to the question "What do you know about this?"

Usage: The student is able to use or apply a concept in a concrete way. Using a concept may include, for example, appropriately using a specific concept in a program, using a particular proof technique, or performing a particular analysis. It provides an answer to the question "What do you know how to do?"

Assessment: The student is able to consider a concept from multiple viewpoints and/or justify the selection of a particular approach to solve a problem. This level of mastery implies more than using a concept; it involves the ability to select an appropriate approach from understood alternatives. It provides an answer to the question "Why would you do that?"

Co-Convened Sections	If the course is co-convened (4470/6470), then graduate students will have additional problems in their homework assignments and exams.
Course Master	Dr. Liming Cai Modified 4/21/23 by Dr. Shelby Funk