ICS C255: ALGORITHM DESIGN

Item	Value
Curriculum Committee Approval Date	12/06/2024
Top Code	070200 - Computer Information Systems
Units	3 Total Units
Hours	54 Total Hours (Lecture Hours 54)
Total Outside of Class Hours	0
Course Credit Status	Credit: Degree Applicable (D)
Material Fee	No
Basic Skills	Not Basic Skills (N)
Repeatable	No
Open Entry/Open Exit	No
Grading Policy	Standard Letter (S), • Pass/No Pass (B)

Course Description

This course introduces algorithm design strategies and their applications in solving common computational problems. Key topics include analyzing the asymptotic behavior of algorithms and exploring core design paradigms such as brute force, exhaustive search, divide-and-conquer, dynamic programming, greedy techniques, backtracking, and branch-and-bound approaches. The course also covers intractability, including NP-complete problems, and introduces parallel and distributed computing concepts to address computational efficiency in complex problem-solving. ADVISORY: ICS C123 and ICS C141 and CIS C157 and a course taught at the level of college algebra or appropriate math placement. Transfer Credit: CSU.

Course Level Student Learning Outcome(s)

1. Analyze and describe the time and space complexity of algorithms using Big O notation.
2. Apply appropriate algorithmic design strategies to efficiently solve a range of computational problems.
3. Implement algorithms in various design paradigms such as dynamic programming and divide-and-conquer.
4. Demonstrate an understanding of the basics of parallel and distributed computing in algorithm optimization.

Course Objectives

- 1. Define algorithms, their history, and importance in computer science.
- 2. Describe the different types of algorithms.
- 3. Provide context for ethical considerations related to the development and use of algorithms.
- 4. Explain the foundational skills necessary for algorithm analysis and design.
- 5. Demonstrate use of tools to evaluate the efficiency of algorithms in terms of time and space.
- 6. Outline a variety of algorithmic strategies applicable to real-world problems.
- 7. Introduce the theory and practice of parallel and distributed computing for handling large-scale data and computational tasks.
- 8. Describe the basics of parallel and distributed computing in algorithm optimization
- 9. Explain algorithmic design strategies and demonstrate how to apply appropriate use.
- 10. Describe the time and space complexity of algorithms using Big O notation.

Lecture Content

Introduction to Algorithms Input and Output Efficiency and Complexity Correctness History of Algorithms Early Mathematical Foundations Algorithmic Processes in Algebra and Computation Formalization in the 20th Century and Turing Machines Modern Use of Algorithms Data-Driven Decision Making Automation and Optimization Ethics and Transparency in AI Design Paradigms Introduction Approaches and techniques Efficient algorithms Introduction to Algorithm Analysis Basics of algorithm analysis: time and space complexity Big O, Big Theta, and Big Omega notations Growth of functions and asymptotic analysis Brute Force and Exhaustive Search Techniques Basics of brute-force algorithms Applications of exhaustive search in combinatorial problems Divide-and-Conquer Strategy Concept and application of divide-and-conquer Case studies: Merge Sort, Quick Sort, Binary Search Dynamic Programming Principles of dynamic programming and memorization Applications: Fibonacci numbers, knapsack problem, shortest path algorithms Greedy Techniques Characteristics of greedy algorithms Examples: Prim s and Kruskal s algorithms for Minimum Spanning Tree, Dijkstra s algorithm for shortest paths Backtracking and Branch-and-Bound Concepts of backtracking for constraint satisfaction Branch-and-bound techniques for optimization Intractability and NP-Completeness Introduction to computational complexity theory Classes P, NP, NP-Complete, and NP-Hard Reductions and implications of NP-complete problems Introduction to Parallel and Distributed Computing Fundamentals of parallel computing models and distributed systems Basic parallel algorithms and their applications in big data processing

Lab Content

Introduction to Algorithm Analysis and Sorting Algorithms Introduce students to basic algorithm analysis and common sorting techniques. Divide and Conquer Algorithms Explore the Divide and Conquer paradigm through a classic algorithm. Dynamic Programming Basics Introduce students to dynamic programming by solving common optimization problems. Greedy Algorithms Teach students how greedy algorithms work and their applications. Graph Algorithms - Depth First Search (DFS) and Breadth First Search (BFS) Implement basic graph traversal algorithms. Dijkstra s Algorithm for Shortest Path Learn how to find the shortest path in a weighted graph using Dijkstra s algorithm. Backtracking Algorithms Introduce students to the backtracking technique for solving problems with constraints. Algorithm Optimization and Complexity Analysis Teach students how to optimize algorithms and analyze their complexity.

Method(s) of Instruction

- Lecture (02)
- DE Delayed Lecture (02D)
- DE Online Lecture (02X)

Instructional Techniques

This course will utilize a combination of lecture, hands-on guided assignments, classroom/discussion student interactions, problem solving, quizzes, tests, and troubleshooting assignments to achieve the goals and objectives of this course. All instructional methods are consistent across all modalities.

Reading Assignments

Students will read about algorithm design concepts and problem-solving methods. Students will read about algorithm analysis techniques. Students will read about bias and ethical and social implications of algorithms.

Writing Assignments

Students will complete written reports related to algorithm design concepts such as dynamic programming and divide-and-conquer. Students will complete written reports related to parallel and distributed computing in algorithm optimization. Students will complete written reports related to bias and ethical and social implications of the use of algorithms.

Out-of-class Assignments

Students will complete written reports and hands-on exercises related to algorithm analysis and design. Students will complete hands-on assignments related to algorithm design such as time and space complexity of algorithms using Big O notation. Students will work in a lab environment to complete the following assignments: Introduction to Algorithm Analysis and Sorting Algorithms Introduce students to basic algorithm analysis and common sorting techniques. Divide and Conquer Algorithms Explore the Divide and Conquer paradigm through a classic algorithm. Dynamic Programming Basics Introduce students to dynamic programming by solving common optimization problems. Greedy Algorithms Teach students how greedy algorithms work and their applications. Graph Algorithms - Depth First Search (DFS) and Breadth First Search (BFS) Implement basic graph traversal algorithms. Dijkstra s Algorithm for Shortest Path Learn how to find the shortest path in a weighted graph using Dijkstra s algorithm. Backtracking Algorithms Introduce students to the backtracking technique for solving problems with constraints. Algorithm Optimization and Complexity Analysis Teach students how to optimize algorithms and analyze their complexity.

Demonstration of Critical Thinking

Students will apply critical thinking skills through the implementation of a basic machine learning algorithms (e.g., linear regression or k-nearest neighbors) using the chosen library. Students will demonstrate critical thinking skills by loading a sample dataset, preprocessing the data, and training the model.

Required Writing, Problem Solving, Skills Demonstration

Students will complete written reports related to algorithm design concepts such as dynamic programming and divide-and-conquer. Students will complete written reports related to parallel and distributed computing in algorithm optimization. Students will complete written reports related to bias and ethical and social implications of the use of algorithms. Students will complete written reports and hands-on exercises related to algorithm analysis and design. Students will complete hands-on assignments related to algorithm design such as time and space complexity of algorithms using Big O notation.

Eligible Disciplines

Computer information systems (computer network installation, microcomputer ...: Any bachelor's degree and two years of professional experience, or any associate degree and six years of professional experience. Computer science: Master's degree in computer science or computer engineering OR bachelor's degree in either of the above AND master's degree in mathematics, cybernetics, business administration, accounting or engineering OR bachelor's degree in engineering AND master's degree in cybernetics, engineering mathematics, or business administration OR bachelor's degree in mathematics AND master's degree in cybernetics, engineering mathematics, or business administration OR bachelor's degree in any of the above AND a master's degree in information science, computer information systems, or information systems OR the equivalent. Note: Courses in the use of computer programs for application to a particular discipline may be classified, for the minimum qualification purposes, under the discipline of the application. Master's degree required. Computer service technology: Any bachelor's degree and two years of professional experience, or any associate degree and six years of professional experience.

Textbooks Resources

1. Required Erickson, J.. Algorithms, 1st ed. Independently published, 2019 Rationale: Open Educational Resource

Other Resources

1. Coastline Library 2. White papers, security reports, and articles are available at no charge to all students at multiple sites as recommended by the instructor.