ICS C123: FUNDAMENTAL DATA STRUCTURES

Item	Value
Curriculum Committee Approval Date	11/17/2023
Top Code	070700 - Computer Software Development
Units	3 Total Units
Hours	72 Total Hours (Lecture Hours 54; Lab Hours 18)
Total Outside of Class Hours	0
Course Credit Status	Credit: Degree Applicable (D)
Material Fee	No
Basic Skills	Not Basic Skills (N)
Repeatable	No
Grading Policy	Standard Letter (S), • Pass/No Pass (B)

Course Description

This course explores the core principles and techniques for designing, implementing, and using fundamental data structures in computer science. Data structures are the foundation of efficient algorithm design and are essential for organizing and managing data in various computational tasks. Students will explore a variety of data structures, their associated algorithms, and their applications through a combination of theory, practical programming exercises, and problem-solving. ADVISORY: ICS C120. Transfer Credit: CSU; UC.

Course Level Student Learning Outcome(s)

1. Explain the significance of data structures in computer science and their role in algorithm efficiency.
2. Design and describe a software project that include algorithms, binary trees, graphs, sorting and hashing.
3. Construct a software system that includes elements of memory management such as reference counts, marking algorithms, and dynamic memory allocation.

Course Objectives

- 1. Discuss the role of data structures in computer science.
- 2. Describe and provide examples of one-dimensional and multidimensional arrays.
- 3. Outline the stack and queue principles.
- 4. Explain the basics of graph theory and graph representations.
- 5. Describe the purpose and use of hashing principles and collision resolution techniques.
- 6. Discuss and provide examples of sorting algorithms (e.g., quicksort, mergesort).
- 7. Define binary search and give examples of other searching techniques.
- 8. Explain advanced data structures, including priority queues and heaps, graph data structures (e.g., adjacency matrices and lists), and balanced search trees (e.g., AVL trees).
- 9. Provide examples of analysis of data structures in programming projects.

Lecture Content

Introduction The Need for Efficiency Interfaces The Model of Computation Correctness, Time Complexity, and Space Complexity Array-Based Lists ArrayStack FastArrayStack ArrayQueue Linked Lists SLList DLList SEList Skiplists The Basic Structure Analysis of skiplists Hash Tables Hashing with Chaining Linear Probing Hash Codes Binary Trees Basic Binary Trees Unbalanced Binary Search Tree Random Binary Search Trees Scapegoat Trees Red-Black Trees Heaps An Implicit Binary Tree A Randomized Meldable Heap Sorting Algorithms Comparison-Based Sorting Count Sorting and Radix Sort Graphs Representing a Graph by a Matrix A Graph as a Collection of Lists Graph Traversal Data Structures for Integers A Digital Search Tree Searching in Doubly-Logarithmic Time A Doubly-Logarithmic Time External Memory Searching The Block Store B-Trees

Lab Content

Arrays and Linked Lists: Implement a dynamic array and a singly linked list data structure. Stacks and Queues: Create a stack and a queue data structure and implement essential operations. Binary Trees: Implement a binary tree and perform basic tree traversal operations (in-order, pre-order, post-order). Hash Tables: Create a hash table and implement hash functions and collision resolution. Graphs and Graph Algorithms: Implement an adjacency list representation of a graph and perform graph traversal algorithms (e.g., depth-first search, breadth-first search). Sorting Algorithms: Implement various sorting algorithms (e.g., bubble sort, insertion sort, quicksort) and compare their performance. Heaps and Priority Queues: Implement a min-heap or max-heap data structure and use it to build a priority queue.

Method(s) of Instruction

- Lecture (02)
- DE Live Online Lecture (02S)
- DE Online Lecture (02X)
- Lab (04)
- DE Live Online Lab (04S)
- DE Online Lab (04X)

Instructional Techniques

This course will utilize a combination of lecture, hands-on guided laboratory 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

Read about data structures, the foundation of efficient algorithm design. Read about data structures, their associated algorithms, and their applications

Writing Assignments

Analyze and compare data structures. Complete hands-on coding exercises.

Out-of-class Assignments

Evaluate data structures to select an appropriate programming language for a given scenario. Discuss data stuctures. Compare programming languages based on their capabilities.

Demonstration of Critical Thinking

Students will analyze programming languages and discuss selection of the appropriate language based on a given scenario. Students will examine a given scenario to identify potential functions to display results.

Required Writing, Problem Solving, Skills Demonstration

The problem-solving exercises will include algorithms, data, functions, recursion, and proof methods. Programming skills demonstrations are included in the projects.

Textbooks Resources

1. Required Morin, P.. Open Data Structures (in C++), 1st ed. Open-Educational Resource, 2023 Rationale: - Legacy Textbook Transfer Data: Zero-textbook cost Open-Educational Resource

Other Resources

1. Coastline Library 2. Technology related white papers and articles are available at no charge to all students at multiple sites as recommended by the instructor.