Computer Science Fundamentals: Algorithms & Data Structures

Course Overview

This intermediate-level programming course delves deep into the theoretical foundations and practical applications of computer algorithms and data structures. Students will engage with sorting algorithms, basic and advanced data structures, and algorithmic problem-solving techniques. The course will enhance students' ability to analyze computational complexity and solve complex problems efficiently using any programming language they are comfortable with, such as Python or Java.

Course Objectives

- Understand and apply various sorting algorithms and analyze their performance using Big O notation.
- Master basic data structures including arrays, linked lists, stacks, queues, hash tables, and binary trees.
- Explore advanced algorithmic concepts such as dynamic programming, graph algorithms, and basic machine learning algorithms.
- Gain practical experience through hands-on exercises in implementing data structures and algorithms.
- Complete a final project focusing on the visualization of a pathfinding algorithm, demonstrating the application of algorithms to solve real-world problems.

Unit 0: Data Representation and Variables (Comp Sci Fundamentals)

Learn how computers represent data and how variables are used to store and manipulate this data. The unit covers number systems, data types, and the fundamentals of variables in programming.

- [ ] Video - How I would learn to code (If I could start over)
- [ ] Converting Binary to (and from) Decimal
- [ ] Two's Complement
- [ ] Variables and Data Types
- [ ] Memory Allocation
- [ ] Operators
    - [ ] Arithmetic
    - [ ] Assignment
    - [ ] Increment

    - ☐ Comparison
- ☐ Syntax Errors vs Semantics Errors

Unit 1: Sorting Algorithms

Covers foundational sorting algorithms and introduces “Big O” notation.

- ☐ Reading - What Are Algorithms?
- ☐ Video - How to Analyze Algorithms
- ☐ Exercise 1.1 - Introduction to Algorithmic Problem Solving
- ☐ Bubble Sort
- ☐ Shaker Sort
- ☐ Counting Sort
- ☐ Merge Sort
- ☐ Quick Sort
- ☐ Reading - Understanding Big O Notation
- ☐ Exercise 1.2 - Calculating Time Complexity
- ☐ Video - An Overview of Computational Complexity
- ☐ **Unit 1 Test**

Unit 2: Basic Data Structures

Explores fundamental data structures including linked lists, trees, and more.

- ☐ Reading - Introduction to Data Structures
- ☐ Exercise 2.1 - Implementing an Array
- ☐ Reading & Exercise 2.2 - Linked Lists
- ☐ Video - Stacks and Queues
- ☐ Exercise 2.3 - Implementing Stacks and Queues
- ☐ Reading & Exercise 2.4 - Hash Tables
- ☐ Video - Tree Data Structures
- ☐ Exercise 2.5 - Binary Trees (BST)
- ☐ **Unit 2 Test**

Unit 3: Advanced Algorithms

Delves into complex algorithms involving searching, dynamic programming, graph theory, and introduces basic concepts of machine learning algorithms.

- ☐ Reading - Searching Algorithms
- ☐ Exercise 3.1 - Binary Search

- ☐ Reading - Dynamic Programming
- ☐ Exercise 3.2 - Fibonacci Series with Memoization
- ☐ Video - Graph Algorithms
- ☐ Exercise 3.3 - Implementing Dijkstra’s Algorithm
- ☐ Reading - Introduction to Machine Learning Algorithms
- ☐ Exercise 3.4 - k-Nearest Neighbors Algorithm
- ☐ Unit 3 Test
- ☐ **Final Project: Pathfinding Algorithm Visualization**
    - Overview - Create a visualization of a pathfinding algorithm like A* or Dijkstra’s Algorithm.
    - Design & Implementation - Use your choice of programming language and libraries to implement the project.
    - Testing and Debugging - Test various cases to ensure your algorithm is working as expected.