EE 367 Computer Data Structures and Algorithms

Credits: 3

Categorization of credits: Engineering topic

Instructor's or course coordinator: Galen Sasaki, Jan. 9, 2021.

Text Book and Other Required Materials: Cormen, Leiserson, Rivest, and Stein, "Introduction to Algorithms", third edition, The MIT Press.

Designation: Required for Computer Engineering, Elective for Electrical Engineering

Catalog Description: EE 367L Computer Data Structures and Algorithms (3) Design and analysis of data structures and algorithms, including correctness and performance. Topics include time complexity, hash tables, sorting, search trees, self-balancing trees, greedy algorithms, dynamic programming, and graph algorithms. Pre: (205 or ICS 212) and (362 or ICS 241) with a minimum grade of C-.

Pre- and Co-requisites: (205 or ICS 212) and (362 or ICS 241) with a minimum grade of C-.

Class/Lab Schedule: 3 lecture hours per week

Topics Covered:

- Linear data structures: arrays, linked list, stacks, queues. (1 hr)
- Analysis of algorithms: correctness; time and space complexity; and amortized analysis. (4 hrs)
- Divide and conquer, recursion, dynamic programming. (3 hrs)
- Linear and binary search. (1 hrs)
- Sorting: insertion sort, selection sort, divide and conquer, mergesort, quicksort, count and radix sort, heapsort. (4 hrs)
- Probability: discrete probability, counting, Bernoulli trials, discrete variables, probabilistic analysis of algorithms, randomization in algorithms, pseudo random number generation. (6 hrs)
- Augmenting data structures. (1 hr)
- Hash functions and tables, and analysis. (2 hrs)
- Rooted trees: binary trees, binary search trees, tree traversals, tries, 2-3-4 trees, B-trees, black-red trees. (5 hrs)
- Heap, heap sort, priority queue. (1 hrs)
- Graph theory. (2 hr)
- Graph algorithms: depth-first and breadth-first search, topological sort, minimum spanning trees, single-source and all-pairs shortest paths. (6 hrs)
- Maxflow. (2 hr)
- NP Completeness. (2 hr)
- Approximation algorithms. (1 hr)

Course Objectives and Relationship to Program Objectives:

The student learns data structures and algorithms and how to apply them to design correct and efficient software. The student must also be able to apply this knowledge to develop software that will solve computer engineering problems. Performance efficiency is a key objective, and the student must be able to analyze algorithms, and to apply the algorithms appropriately. The student understand and be able to apply discrete math models and formulas including probability theory. The student must be able to prove discrete math formulas and correctness of algorithms.

Program Objectives this course addresses: 1, 2, 3, 4.

Course Outcomes and Their Relationship to Program Outcomes

The following are the course outcomes and the subset of Program Outcomes (numbered 1-8 in square braces "[ ]") they address:

- Able to implement and design data structures and algorithms in software [1,2,8]
- Able to apply and develop efficient data structures and algorithms to solve computer engineering problems [1,2,8]
- Able to apply discrete math and probability to solve computer engineering problems [1,8]
- Be able to analyze algorithms both in correctness and complexity [1,8]
- Be able to evaluate performance by simulation, e.g., Monte Carlo simulation, and interpret data, e.g., confidence intervals [1,6,8]
- Be able to design an experiment to evaluate performance [1,6,8]
- Be able to design and interpret data from simulation experiments to analyze an algorithm [1,6,8]
- Be able to understand, apply, and derive discrete math models and formulas [1,8]
- Be able to understand, apply, and derive elementary probability models and formulas [1,8]

Contribution of Course to Meeting the Professional Component

Engineering topics: 100%.

Computer Usage:

About 30% of the grade is based on programming problems.

Design Credits and Features:

EE 367 has 2 design credits. The programming assignments are about 30% of the grade. All assignments involve design. In addition, the paper-and-pencil homeworks cover computer engineering problem solving used in design.