CSC 364 Data Structures and Algorithms

CATALOG DESCRIPTION:

CSC 364 Data Structures and Algorithms (3,0,3) Analysis and efficient implementation of container types such as stacks, queues, hash tables, search trees, and graphs; sorting algorithms. PREREQ: C or better in CSC 360.

LAST TAUGHT: Spring 2009 (L. Weiner)

SCHEDULED LAB USAGE: None

STUDENT BACKGROUND EXPECTATIONS:

Students are expected to know Java at the CS I-II level, or know another object-oriented language (e.g. C++ or C#) at this level and be capable of picking up equivalent Java skills during the course.

CORE TOPICS COVERED:

- Recursion (review)
- Data Abstraction (ADTs)
- Linked Lists
- Stacks
- Queues
- Advanced Java Topics (review: inheritance, dynamic binding, abstract classes, interfaces, iterators, generics)
- Algorithm Efficiency and Time Complexity (Big-O)
- Sorting
- Trees
- Tables and Priority Queues
- Balanced Search Trees, Hashing
- Graphs

MOST RECENT TEXTBOOK USED:

Chapters Covered: 2.3, 3-5, 6.1, 7-14.

SOFTWARE REQUIRED:

Any Java IDE (free).

STUDENT WORK

Programming assignments and in-class exams.

LEARNER OUTCOMES

Students will be able to...

1. Understand the concept of abstract data types (ADT);
2. Describe and define (via Java interfaces) abstract data types for standard containers such as: lists, stacks, queues, sets, maps, priority queues, trees, graphs;
3. Implement (via concrete Java classes) the above ADTs using various data structures (such as arrays, linked lists) and their attendant algorithms;
4. Understand and implement the following basic sorting algorithms: bubble sort, selection sort, insertion sort, merge sort, quick sort, radix sort, heap sort;
5. Design and build applications starting from abstract data types;
6. Choose the appropriate structure for a certain application;
7. Design and implement algorithms that use these data structures;
8. Understand the concept of computational complexity;
9. Define big-O and contrast big-O values for the data structures and the algorithms they use;
10. Create the ability to compare the cost of static and dynamic allocation for different data structures;
11. Analyze and implement recursive algorithms.

CROSS-LISTINGS
None