Course Outcomes	CSC 3102

CSC 3102: Advanced Data Structures and Algorithm Analysis

Credit Hours: 3 hours

Prerequisites:

CSC 1254 or 1351 and credit or concurrent enrollment in CSC 2259 or EE 2720.

Prerequisites By Topic:

Logic, sets, permutations and combinations, mathematical induction, discrete probability, garph and trees, binary trees, Boolean algebra, advanced programming skills, concept of abstract data type, linear data structures (linked list, array, stack, queue), binary search trees.

Catalog Course Description:

Description and utilization of formal ADT representations, especially those on lists, sets, and graphs; time and space analysis of recursive and nonrecursive algorithms, including graph and sorting algorithms; algorithm design techniques.

Course Outcomes

1. Basic ability to analyze algorithms and to determine algorithm correctness and time efficiency class.
2. Master a variety of advanced abstract data type (ADT) and data structures and their implementations.
3. Master different algorithm design techniques (brute-force, divide and conquer, greedy, etc.)
4. Ability to apply and implement learned algorithm design techniques and data structures to solve problems.

Texts and Other Course Materials

The Design & Analysis of Algorithms - Anany Levitin, 0-201-74395-7, HB Latest PWS
Reference: Data Structures – a Pseudocode Approach with C .R.F. Gilberg and B.A. Forouzan

Major Topics

- Introduction of basic concepts of algorithms, ADT, and a quick review of linear data structures.
- Fundamentals of algorithm time complexity analysis including orders of growth, big-O, big-Omega, big-Theta notations. Analysis of recursive and non-recursive algorithms.
- Introduction and comparison of the efficiency of various sorting algorithms including the insertion sort, selection sort, quick sort, merge sort, and heap sort,
- Advanced searching methods, tree traversals, and implementation.

- Advanced trees including binary search trees, B-trees, heaps, AVL trees, minimum spanning trees.
- String matching, combinatorial problems, greedy techniques and dynamic programming.
- Graph-related algorithms, topological sorting, shortest path, graph traversal,
- Principles of heaps and priority queues, and implementation,
- Sets, dictionaries and maps including hash tables.

Assignments/Projects/Laboratory Projects/Homeworks

- Individual written homework assignments (4-5).
- Individual programming assignments (totally 5-6). Each takes about one to two weeks to complete.

Sample projects:

1. Use the AVL-tree insertion/search algorithms to write an AVL-tree ADT, and use it in your program to construct a dictionary representing the book titles held in various libraries. The program then should answer queries to the dictionary about book titles.
2. Implement a graph ADT containing procedures to perform graph traversals (DFS, BFS), graph connectivity test (for undirected graph), topological sorting (for directed graphs).
3. Use either B-tree algorithms or open-hashing algorithms to implement a dictionary for the same problem as (1).

Curriculum Category Content (estimated in semester hours)

Area	Core	Advanced	Area	Core	Advanced
Algorithms	9	10	Data Structures	9	12
Software Design	3		Prog. Languages		
Computer Architecture			Mathematical fundamentals	2	3

Relationship to Criterion 3 Outcomes

A	B	C	D	E	F	G	H	I	J	K
*	*	*		*		*	*	*		*

Math and Fundamentals -- 2hr core/3hr advanced

Fundamentals of algorithm analysis, basic time complexity classes, mathematical analysis of non-recursive and recursive algorithms. Recurrence relations and their use in algorithm complexity analysis. Totally about 5 hours are spent on these topics.

Data Structures -- 9 hr core/12 hr advanced
Binary search trees, B-trees, AVL-trees, heaps, priority queues, hashing, graphs, and utilization of these data structures for problem solving.

Algorithms and Software -- 12 hr core/10 hr advanced

Analysis of time complexity for various algorithms discussed through the semester.
Algorithm design using various techniques: brute-force, divide and conquer, decrease and conquer, transformation-based methods, dynamic programming, and greedy method. Algorithms for sorting, search, for combinatorial problems, numerical and geometric problems, and string processing algorithms. Software design issues are discussed along with algorithm design and occasionally during discussions on programming assignment.

Computer Organization and Architecture:

Concepts of Programming Languages:

Social and Ethical Issues:

Oral Communication (presentations):

Written Communication:
Students are required to submit 5 written home works involving discussions of algorithm design issues. The programming assignments (4-5) also require the students to write comments/specifications of the ADT functions in their programs.

Course Coordinator: Dr. Jianhua Chen
Last Modified: March 15, 2007