THOMPSON RIVERS UNIVERSITY

Course Outline

Department of Computing Science
Faculty of Science

COMP 2230 - 3
Data Structures and Algorithm Analysis (3,1,0)
Fall 2015

Instructor:	Phone/Voice Mail:
Office:	E-Mail:

Course Description

Students are introduction to the basic methods of representing data in Computing Science. Students review, implement and analyze several fundamental data structures including lists, stacks, queues, and graphs. Students learn the implementation of algorithms using these data structures and the efficiency and cost tradeoffs of each of them.

Educational Objectives/Outcomes

Upon successful completion of the course, the student will demonstrate the ability to:

1. Familiar with the programming problems that can be solved using these data structures.
2. Understand the common operations on these data structures and how to implement them using Java.
3. Understand the sorting algorithms such as bubble, selection, insertion, merge, and quick.
4. Explain the use of search methods such as linear, binary, hash.
5. Familiar with the use of mathematical techniques to analyse the efficiency of the various searching, and sorting algorithms.
6. Understand the programming techniques appropriate to developing middle-sized programs.

Prerequisites

A letter grade of C or better in Computing 1230 and Computing 1390

Required Texts/Materials

1. Lewis, DePasquale and Chase; *Java Foundations, An Introduction to Program Design and Data Structures,* 3rd ed, Pearson Education Inc., 2014, ISBN 0-13-337046-1
2. Storage device for saving programs, etc. (e.g. flash memory device)
3. TRU Lab/Network Computer Account.
4. Standard 8.5 x 11 (letter-size) laser/inkjet printer paper *(for printing in TRU computer labs)*.

Syllabus - Lecture Topics:

Unit	**Chapter**	**Duration**
1. Linked Structures -- Stacks	chapter 13	1 week
2. Queues	chapter 14	1/2 week
3. Lists	chapter 15	1/2 week
4. Iterators	chapter 16	1 week
5. Sorting	chapter 18	1 week
6. Trees	chapter 19	1 week
7. Binary Search Trees	chapter 20	1 week
8. Heaps and priority queues	chapter 21	1 week
9. Hashing	App I	1 week
10. Sets and maps	chapter 22	1 week
11. Graphs	chapter 24	1 week
12. Modelling and Simulation	Instructor notes	1 week
13. Multi-way Search Trees	Chapter 23	1 week
14. Midterms & review	....	1 week

Syllabus - Lab Topics:
Unit

1. Linked Structures – Stacks
   Complete implementation of linked stack from text book
   Design and implement a drop out stack
2. Queues
   Complete implementation of linked stack from text book
   Complete implementation of Circular Array Queue from text book
3. Lists
   Complete implementation of Linked List Class from text book
   Complete implementation of Linked Ordered List Class from text book
4. Iterators
   Add an iterator to the Linked List Class
   Add an iterator to the Linked Ordered List Class
5. Sorting
   Comparison of the different sorting algorithms via testing
6. Trees

7. Binary Search Trees
   Build an array implementation of a binary search tree
8. Heaps and priority queues
   Complete implementation of heap from text book
   Implement a priority queue using a heap
9. Hashing
   Build a hashing application
10. Sets and maps
   Programming an application using sets and maps
11. Graphs
   Using graphs to solve maze problems
12. Modelling and Simulation
   Build a queuing simulation

ACM / IEEE Knowledge Area Coverage

Knowledge Areas that contain topics and learning outcomes covered in the course

Knowledge Area	Total Hours of Coverage
AL/Basic Analysis	4
AL/Fundamental Data Structures and Algorithms	12
CN/Introduction to Modeling and Simulation	1
DS/Graphs and Trees	4
PL/Functional Programming	7
SDF/Algorithm Design	11
SDF/Fundamental Data Structures	4

Body of Knowledge coverage

KA	Knowledge Unit	Topics Covered	T1 hours	T2 hours	Elective hours
AL	Basic Analysis	[Core-Tier1] • Differences among best, expected, and worst case behaviors of an algorithm • Asymptotic analysis of upper and expected complexity bounds • Big O notation: formal definition • Complexity classes, such as constant, logarithmic, linear,	2	2	0

		quadratic, and exponential • Empirical measurements of performance • Time and space trade-offs in algorithms [Core-Tier2] • Big O notation: use • Little o, big omega and big theta notation • Recurrence relations • Analysis of iterative and recursive algorithms • Some version of a Master Theorem			
AL	Fundamental Data Structures and Algorithms	[Core-Tier1] • Simple numerical algorithms, such as computing the average of a list of numbers, finding the min, max, and mode in a list, approximating the square root of a number, or finding the greatest common divisor • Sequential and binary search algorithms • Worst case quadratic sorting algorithms (selection, insertion) • Worst or average case O(N log N) sorting algorithms (quicksort, heapsort, mergesort) • Hash tables, including strategies for avoiding and resolving collisions • Binary search trees o Common operations on binary search trees such as select min, max, insert, delete, iterate over tree • Graphs and graph algorithms o Representations of graphs (e.g., adjacency list, adjacency matrix) o Depth- and breadth-first traversals [Core-Tier2] • Heaps • Graphs and graph algorithms	9	3	0

		o Shortest-path algorithms (Dijkstra's and Floyd's algorithms) o Minimum spanning tree (Prim's and Kruskal's algorithms) • Pattern matching and string/text algorithms (e.g., substring matching, regular expression matching, longest common subsequence algorithms)			
CN	Introduction to Modeling and Simulation	***Topics:*** • Models as abstractions of situations • Simulations as dynamic modeling • Simulation techniques and tools, such as physical simulations, human-in-the-loop guided simulations, and virtual reality • Foundational approaches to validating models (e.g., comparing a simulation's output to real data or the output of another model) • Presentation of results in a form relevant to the system being modeled	1	0	0
DS	Graphs and Trees	[Core-Tier1] • Trees o Properties o Traversal strategies • Undirected graphs • Directed graphs • Weighted graphs [Core-Tier2] • Spanning trees/forests • Graph isomorphism	3	1	0
PL	Functional Programming	[Core-Tier1] • Effect-free programming o Function calls have no side effects, facilitating compositional reasoning o Variables are immutable, preventing unexpected changes to program data by other code	3	4	0

		o Data can be freely aliased or copied without introducing unintended effects from mutation • Processing structured data (e.g., trees) via functions with cases for each data variant o Associated language constructs such as discriminated unions and pattern-matching over them o Functions defined over compound data in terms of functions applied to the constituent pieces • First-class functions (taking, returning, and storing functions) [Core-Tier2] • Function closures (functions using variables in the enclosing lexical environment) o Basic meaning and definition -- creating closures at run-time by capturing the environment o Canonical idioms: call-backs, arguments to iterators, reusable code via function arguments o Using a closure to encapsulate data in its environment o Currying and partial application • Defining higher-order operations on aggregates, especially map, reduce/fold, and filter			
SD F	Algorithm Design	• The concept and properties of algorithms o Informal comparison of algorithm efficiency (e.g., operation counts) • The role of algorithms in the problem-solving process • Problem-solving strategies o Iterative and recursive mathematical functions o Iterative and recursive traversal of data structures o Divide-and-conquer strategies • Fundamental design concepts and principles	11	0	0

		o Abstraction o Program decomposition o Encapsulation and information hiding o Separation of behavior and implementation			
SD F	Fundamental Data Structures		0	4	0