Summer 2013

CS 3100: Data Structures and Algorithms

Erik Marlow Buck
Wright State University - Main Campus, erik.buck@wright.edu

Course Syllabus

I. College/School **College of Engineering and Computer Science**
Department **Department of Computer Science and Engineering**

II. Course Information

Course Title: **Data Structures and Algorithms**
Course Abbreviation and Number: **CS 3100**
Course Cross Listing(s) Abbreviation and Number:
Check ("x") all applicable:
General Education Course_____ Writing Intensive Course_____ Service Learning Course_____
Laboratory Course_____ Ohio TAG (Transfer Assurance Guide) Course _____
Ohio Transfer Module Course_____ Others (specify)_____

III. Course Registration

Prerequisites: **(C or higher in CS 1181) and (C or higher in CEG 3310) and CEG 2350 and (MTH 2570 or CS 2200)**
Corequisites: **None**
Restrictions: **None**

IV. Student Learning Outcomes

- **Analyze basic algorithms for space and time complexity**
- **Design abstract data types appropriate for a given problem**
- **Implement data structures in an efficient manner**
- **Design and implement non-graphical user interfaces**
- **Select and implement appropriate data structures for a given problem**
- **Design algorithms to solve specific problems**

V. Suggested Course Materials (required and recommended)

Example Required Text:

Data Structures and Algorithm Analysis, 2nd Edition, by C. A. Shaffer, Prentice Hall, 2001.

VI. Suggested Method of Instruction

Lecture

VII. Suggested Evaluation and Policy

Two midterm exams (15% each), one final exam (35%), 5 programming projects (35%).

VIII. Suggested Grading Policy

A: 100-90, B: 89-80, C: 79-70, D: 69-60, F: less than 60.

IX. Suggested Assignments and Course Outline

Week	Topics/Activity	Text Reading
1	Abstract Data Types, Implementation in C++	Chs. 1, 2
2	Algorithm Analysis for iterative algorithms	Ch. 3
3	Review of lists, stacks, queues	Ch. 4
4	Binary Trees	Ch. 5
5	General Trees	Ch. 6
6	Searching: Self-organizing structures, Hashing	Ch. 9
7	Tries and Height Balanced Trees	Ch. 13.1 – 13.2
8	Graphs: Definitions, Implementation, Traversal	Ch. 11.1 – 11.3

9	Graphs: Shortest Paths, Spanning Trees	Ch. 11.4 – 11.8
10	Sorting: Internal and External	Ch. 7, 8
11	Indexing: Linear and Tree Indexing, B-Trees	Ch. 10
12	Algorithms: greedy, exhaustive, dynamic programming, branch & bound	Ch. 14 & 15
13	Algorithms: decision problems, decidability, P and NP	Ch. 15
14	Hardware and data structures: garbage collection, cache, virtual memory, and efficiency	Web resources

X. Other Information

Programming assignments will include additional, more complex objectives for students enrolled in CEG 5100. Additionally, the midterm and final exams will each have one or more additional questions for students enrolled in CEG 5100. These questions should cover material of a more theoretical and technical nature than the undergraduate questions.

Knowledge Topics introduced in this course

AL/Basic Analysis • Differences among best, average, and worst case behaviors of an algorithm
AL/Basic Analysis • Asymptotic analysis of upper and average complexity bounds
AL/Basic Analysis • Big O notation: formal definition
AL/Basic Analysis • Time and space trade-offs in algorithms
AL/Basic Analysis • Big O notation: use
AL/Basic Analysis • Little o, big omega and big theta notation
AL/Algorithmic Strategies • Brute-force algorithms
AL/Algorithmic Strategies • Greedy algorithms
AL/Algorithmic Strategies • Divide-and-conquer (cross-reference SDF/Algorithms and Design/Problem-solving strategies)
AL/Fundamental Data Structures and Algorithms • Worst case quadratic sorting algorithms (selection, insertion)
AL/Fundamental Data Structures and Algorithms • Worst or average case O(N log N) sorting algorithms (quicksort, heapsort, mergesort)
AL/Fundamental Data Structures and Algorithms • Hash tables, including strategies for avoiding and resolving collisions
AL/Fundamental Data Structures and Algorithms • Binary search trees
AL/Fundamental Data Structures and Algorithms • Common operations on binary search trees such as select min, max, insert, delete, iterate over tree
AL/Fundamental Data Structures and Algorithms • Graphs and graph algorithms
AL/Fundamental Data Structures and Algorithms • Representations of graphs (e.g., adjacency list, adjacency matrix)
AL/Fundamental Data Structures and Algorithms • Depth- and breadth-first traversals
AL/Fundamental Data Structures and Algorithms • Graphs and graph algorithms
AL/Fundamental Data Structures and Algorithms • Shortest-path algorithms (Dijkstra's and Floyd's algorithms)
AL/Fundamental Data Structures and Algorithms • Minimum spanning tree (Prim's and Kruskal's algorithms)
AL/Fundamental Data Structures and Algorithms • Pattern matching and string/text algorithms (e.g., substring matching, regular expression matching, longest common subsequence algorithms)
DS/Graphs and Trees • Undirected graphs
DS/Graphs and Trees • Directed graphs
DS/Graphs and Trees • Weighted graphs
DS/Graphs and Trees • Traversal strategies
DS/Graphs and Trees • Spanning trees/forests
DS/Discrete Probability • Finite probability space, events
IS/Basic Search Strategies • Uninformed search (breadth-first, depth-first, depth-first with iterative deepening)
IS/Basic Search Strategies • Space and time efficiency of search
PL/Language Translation and Execution • Automated vs. manual memory management; garbage collection as an automatic technique using the notion of reachability
SDF/Algorithms and Design • The concept and properties of algorithms
SDF/Algorithms and Design o Informal comparison of algorithm efficiency (e.g., operation counts)
SDF/Algorithms and Design • The role of algorithms in the problem-solving process
SDF/Algorithms and Design • Problem-solving strategies
SDF/Algorithms and Design o Iterative and recursive mathematical functions
SDF/Algorithms and Design • Implementation of algorithms
SDF/Algorithms and Design • Fundamental design concepts and principles
SDF/Algorithms and Design o Program decomposition
SDF/Fundamental Data Structures • Strategies for choosing the appropriate data structure
SP/ Professional Communication • Reading, understanding and summarizing technical material, including source code and documentation
SP/ Professional Communication • Writing effective technical documentation and materials

Prerequisite knowledge topics further developed in this course

AL/Basic Analysis • Complexity classes, such as constant, logarithmic, linear, quadratic, and exponential
AL/Basic Analysis • Empirical measurements of performance
AL/Fundamental Data Structures and Algorithms • 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
AL/Fundamental Data Structures and Algorithms • Sequential and binary search algorithms
DS/Graphs and Trees • Trees
PL/Basic Type Systems • Generic types (parametric polymorphism)

PL/Basic Type Systems o Definition
PL/Basic Type Systems o Use for generic libraries such as collections
PL/Basic Type Systems o Comparison with ad hoc polymorphism (overloading) and subtype polymorphism
PL/Language Translation and Execution • Run-time layout of memory: call-stack, heap, static data
PL/Language Translation and Execution o Implementing loops, recursion, and tail calls
SDF/Algorithms and Design o Iterative and recursive traversal of data structure
SDF/Algorithms and Design o Divide-and-conquer strategies
SDF/Algorithms and Design o Abstraction
SDF/Algorithms and Design o Encapsulation and information hiding
SDF/Algorithms and Design o Separation of behavior and implementation
SDF/Fundamental Programming Concepts • Functions and parameter passing
SDF/Fundamental Data Structures • Arrays
SDF/Fundamental Data Structures • Stacks, queues, priority queues, sets & maps
SDF/Fundamental Data Structures • Simple linked structures
SDF/Development Methods • Program correctness
SDF/Development Methods • The concept of a specification
SDF/Development Methods • Defensive programming (e.g. secure coding, exception handling)
SDF/Development Methods • Documentation and program style
SF/Cross-Layer Communications • Programming abstractions, interfaces, use of libraries

Prerequisite knowledge topics used in this course
AL/Algorithmic Strategies • Recursive backtracking
AR/Machine-level representation of data • Bits, bytes, and words
AR/Machine-level representation of data • Numeric data representation and number bases
AR/Machine-level representation of data • Signed and twos-complement representations
AR/Machine-level representation of data • Representation of non-numeric data (character codes, graphical data)
AR/Machine-level representation of data • Representation of records and arrays
AR/Assembly level machine organization • Basic organization of the von Neumann machine
AR/Assembly level machine organization • Subroutine call and return mechanisms
AR/Assembly level machine organization • Heap vs. Static vs. Stack vs. Code segments
DS/Sets, Relations, and Functions • Sets
DS/Sets, Relations, and Functions • Venn diagrams
DS/Sets, Relations, and Functions • Union, intersection, complement
DS/Sets, Relations, and Functions • Cartesian product
DS/Sets, Relations, and Functions • Power sets
DS/Sets, Relations, and Functions • Cardinality of finite sets
DS/Sets, Relations, and Functions • Relations
DS/Sets, Relations, and Functions • Reflexivity, symmetry, transitivity
DS/Sets, Relations, and Functions • Equivalence relations, partial orders
DS/Sets, Relations, and Functions • Functions
DS/Sets, Relations, and Functions • Surjections, injections, bijections
DS/Sets, Relations, and Functions • Inverses
DS/Sets, Relations, and Functions • Composition
DS/Basic Logic • Propositional logic (cross-reference: Propositional logic is also reviewed in IS/Knowledge Based Reasoning)
DS/Basic Logic • Logical connectives
DS/Basic Logic • Truth tables
DS/Basic Logic • Normal forms (conjunctive and disjunctive)
DS/Basic Logic • Validity
DS/Basic Logic • Propositional inference rules (concepts of modus ponens and modus tollens)
DS/Basic Logic • Predicate logic
DS/Basic Logic • Universal and existential quantification
DS/Basic Logic • Limitations of propositional and predicate logic (e.g., expressiveness issues)
DS/Proof Techniques • The structure of mathematical proofs
DS/Proof Techniques • Direct proofs
DS/Proof Techniques • Disproving by counterexample
DS/Proof Techniques • Proof by contradiction
DS/Proof Techniques • Induction over natural numbers
DS/Proof Techniques • Structural induction
DS/Proof Techniques • Weak and strong induction (i.e., First and Second Principle of Induction)
DS/Proof Techniques • Recursive mathematical definitions
DS/Basics of Counting • Counting arguments
DS/Basics of Counting • Set cardinality and counting
DS/Basics of Counting • Sum and product rule
DS/Basics of Counting • Inclusion-exclusion principle
DS/Basics of Counting • Arithmetic and geometric progressions
DS/Basics of Counting • The pigeonhole principle
DS/Basics of Counting o Permutations and combinations
DS/Basics of Counting • Basic definitions
DS/Basics of Counting • The binomial theorem
DS/Basics of Counting • An example of a simple recurrence relation, such as Fibonacci numbers
DS/Basics of Counting • Basic modular arithmetic
PL/Object-Oriented Programming • Object-oriented design
PL/Object-Oriented Programming o Decomposition into objects carrying state and having behavior
PL/Object-Oriented Programming o Class-hierarchy design for modeling
PL/Object-Oriented Programming • Definition of classes: fields, methods, and constructors
PL/Object-Oriented Programming • Subclasses, inheritance, and overriding

PL/Object-Oriented Programming • Dynamic dispatch: definition of method-call
PL/Object-Oriented Programming • Subtyping (cross-reference PL/Type Systems)
PL/Object-Oriented Programming o Subtype polymorphism; implicit upcasts in typed languages
PL/Object-Oriented Programming o Notion of behavioral replacement
PL/Object-Oriented Programming o Relationship between subtyping and inheritance
PL/Object-Oriented Programming • Object-oriented idioms for encapsulation
PL/Object-Oriented Programming o Private fields
PL/Object-Oriented Programming o Interfaces revealing only method signatures
PL/Object-Oriented Programming o Abstract base classes
PL/Object-Oriented Programming • Using collection classes, iterators, and other common library components
PL/Basic Type Systems • A type as a set of values together with a set of operations
PL/Basic Type Systems o Primitive types (e.g., numbers, Booleans)
PL/Basic Type Systems o Reference types
PL/Basic Type Systems o Compound types built from other types (e.g., records, unions, arrays, lists, functions)
PL/Basic Type Systems • Association of types to variables, arguments, results, and fields
PL/Basic Type Systems • Type safety and errors caused by using values inconsistently with their intended types
PL/Basic Type Systems • Goals and limitations of static typing
PL/Basic Type Systems o Eliminating some classes of errors without running the program
PL/Basic Type Systems o Inherent conservative approximation of static analysis due to undecidability
PL/Language Translation and Execution o Execution as native code or within a virtual machine
SDF/Fundamental Programming Concepts • Basic syntax and semantics of a higher-level language
SDF/Fundamental Programming Concepts • Variables and primitive data types (e.g., numbers, characters, Booleans)
SDF/Fundamental Programming Concepts • Expressions and assignments
SDF/Fundamental Programming Concepts • Simple I/O
SDF/Fundamental Programming Concepts • Conditional and iterative control structures
SDF/Fundamental Programming Concepts • The concept of recursion
SDF/Fundamental Data Structures • Records/structs (heterogeneous aggregates)
SDF/Fundamental Data Structures • Strings and string processing
SDF/Fundamental Data Structures • References and aliasing
SDF/Development Methods • The role and the use of contracts, including pre- and post-conditions
SDF/Development Methods • Debugging strategies