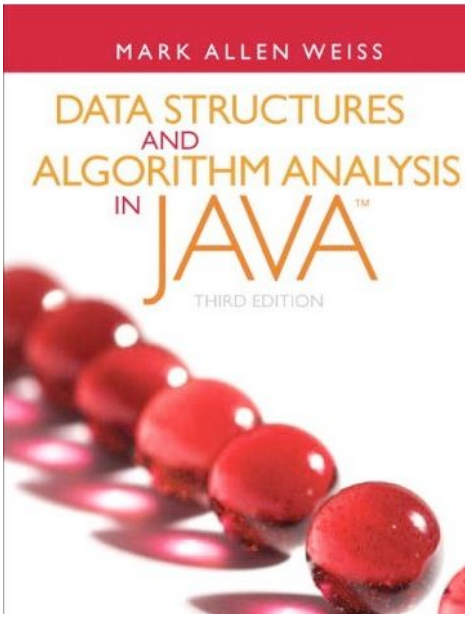
Fall 2015

CSCI 2125

Farjana Z. Eishita
University of New Orleans

Fall 2015

CSCI 2125: Data Structures



Instructor:
Farjana Z. Eishita
Office: Math 341
Email: farjana.eishita@uno.edu
Office hour:

- Monday 3 pm - 5 pm
- Tuesday 1 pm - 2 pm
- Friday 3 pm – 5 pm

Classes: Mon, Wed, Fri 2:00PM – 2:50 PM
Classroom: Math 112
Prerequisite: CSCI 2120 and 2121 with a grade of C or better or consent of department; credit for or concurrent registration in MATH 3712 is required.

Textbook: Mark Allen Weiss, *Data Structures and Algorithm Analysis in Java. 3rd Edition*

Course Description:

Consider all of the tweets on Twitter or all of the status messages on Facebook. For a moment, think of yourself as a programmer for one of these companies. If someone asked you to design a method to search or sort every tweet or message ever sent, how would you begin to think about such a problem? How do you know the difference between an approach that takes 200 milliseconds and 200 years? How would you organize the data in a way that can help you meet your goals? How do you even know where to start?

This course will give you the tools to answer these questions. We will study the organization of data and learn how to analyze the impact of algorithms not only in our own programs, but in programs that must handle thousands, millions, or billions of data points. We will learn about the following concepts:

Algorithms and algorithm analysis
Programs are often judged in terms of how fast they run and how effectively they utilize the available memory space. Interestingly, how fast one program runs compared to another program can be established by looking at their underlying algorithms in theory only (without bothering with the actual implementation, or characteristics of a particular computer on which the programs are run). We will study the way to describe the efficiency of algorithms and evaluate various approaches to implementing data structures and algorithms. This will allows us to make an intelligent choice and alter of a data structure suitable for a particular program.

Standard Data Structures
A *data structure* is a particular way of organizing and manipulating data. Smart choice of data structures can dramatically speed up program's execution (and poor choice of data structures may render a program so slow that it becomes unusable). We are going to learn organization

and proper use of classical data structures (e.g., stack, queue, array, linked list, tree).

Data Abstraction
Throughout the course, we will be practicing with a very powerful notion of separating abstract properties of data types from their concrete implementations in a particular programming language. We will learn to view complex programs as combinations of individual pieces and reason about the behavior of those pieces independently of their actual implementation. This will ultimately help you develop more versatile software that is easy to update and reuse.

Some of the key topics that will be covered in this course include the following:

- Algorithm Analysis
- Linked-Lists
- Stacks & Queues
- Trees (Binary, AVL, Red-Black, Splay, B-Trees)
- Maps
- Hash Tables and Functions
- Heaps
- Searching & Sorting (Optimal Sorting and Linear Sorting)
- Sets
- Graphs and Graph Algorithms
- Algorithm Design Techniques (Greedy, Divide & Conquer, Dynamic, etc.)

Student Learning Outcomes

After successful completion of the course the students will have fulfilled the following objectives in the least:

- Students will be able to explain and use fundamental data structures, data types, and programming techniques.
- Students will be able to explain and use introductory algorithm analysis techniques.
- Students will be able to design, implement, and test programs for problems using algorithms and data structures.

Attendance Policy:

Students are expected to attend classes on time. The UNO Senate (Feb. 20, 2002) has made the taking of attendance a requirement for "developmental, 1000, and 2000 level courses." Attendance will therefore be taken at each class meeting. Although not a formal component of the computation of grades, good attendance will impact final grades in borderline cases. Important course content is often introduced outside of the published/provided sources and/or scheduled presentations.

Academic Integrity:

Academic integrity is fundamental to the process of learning and evaluating academic performance. Academic dishonesty will not be tolerated. Academic dishonesty includes, but is not limited to, the following: cheating, plagiarism, tampering with academic records and examinations, falsifying identity, and being an accessory to acts of academic dishonesty. Refer to the Student Code of Conduct for further information. The Code is available online at

http://www.studentaffairs.uno.edu.

Academic dishonesty, in particular, includes "the unauthorized collaboration with another person in preparing an academic exercise" and "submitting as one's own any academic exercise prepared totally or in part for/by another." In the event of academic dishonesty, the student may be assigned a grade of 0 on the exam or exercise, the student may be informed in writing of the action taken, and a copy of this letter may be sent to the Assistant Dean for Special Student Services.

Students with Disabilities:

It is University policy to provide, on a flexible and individualized basis, reasonable accommodations to students who have disabilities that may affect their ability to participate in course activities or to meet course requirements. Students with disabilities should contact the Office of Disability Services as well as their instructors to discuss their individual needs for accommodations. For more information, please go to http://www.ods.uno.edu.

Course Engagement:

The students are expected to attend classes and participate class discussions and activities. There will be assignments, homework, quizzes, and exams, as described below:

- **Assignments/homework**: take home programming and/or writing assignment
- **Quizzes**: there may be sudden (i.e., pop) quizzes starting at the beginning of class. Do not miss class or be late in class so that you do not miss a quiz or have inadequate time to complete your quiz.
- **Exams:** There will be two exams, one midterm and one final exam in the scheduled week.

Grade Calculation:

(1) The tentative total/final grade points for the course is distributed as follows:

Attendance	**6%**
Homework/assignments	**30%**
Quizzes	**24%**
Final exam	**40%**

There may be weekly homework/assignment/quizzes.

(2) All work is graded on a numerical (percentage) basis. The correspondence between numerical and letter grades is given as follows:

A: >= 90,
B: 80 - 89,
C: 70 - 79,
D: 50 - 69,
F: < 50.

(3) It is expected that all homework will be turned in on time. Lateness penalties are n points off where n is the number of days late, and $n \leq 10$, which means anything past due over ten

days will not be accepted.

- 1 day late – 1 pts off;
- 2 days late – 2 pts off;
- 3 days late – 3 pts off;
- 4 days late – 4 pts off;
- 5 days late – 5 pts off;

Note: We count school days (Weekends and holidays are not included).

(4) No make-ups for graded work (either tests or homework) will be given except for a legitimate (*e.g.*, medical) reasons.

(5) Questions about the grading of student work should be raised within 72 hours of its return. After that time frame, issues raised will risk not being entertained.

(6) Students should retain all returned graded work, in case there are issues raised about the grade.

(7) The "I" grade (for Incomplete) is given only in exceptional circumstances, (*e.g.* missing the final exam because of a surgery).

Programming Assignments Grading Rubric

You are expected to write complete, working, clear, efficient programs using good programming practices and clear documentation. If you do this, your assignments will get an A. General rubric breaks down as follows, unless otherwise specified for particular assignments.

40 % – Compiles and Runs correctly.

20 % – Is your program accompanied by test code? If JUnit tests were provided or specified, does the program pass all the JUnit tests provided/specified with the assignment? If no JUnit tests are provided/specified, your program should still be accompanied by JUnit tests that you design by yourself to make program verification possible (i.e., to verify that your program accomplishes all the goals of the assignment)

20 % – Is the program efficient? Did you choose/design good algorithms and data structures for the task? Did you justify your choices in either the comments or a short write up about your program (for example, in a ReadMe.txt file)?

20 % - Were good programming practices used? Does your program have clear indentation, expressive names for variables, methods, and classes? Does the program include clear documentation (JavaDoc and inline comments)? Is your program well organized and correctly modularized into different packages and source files?

Note: Only after your program achieves that first 40% for successful compilation and correct execution, your program will then be considered for additional points. This means, if your program fails to compile and/or execute successfully, you will not get any points.

Tentative Schedule of Study:

WEEK 1 (Aug 19 – Aug 21) Chapters 1&2: Introduction and Algorithm Analysis

Math Review, Function Objects (1.2, 1.5, 1.6)
Asymptotic Analysis (2.1)

WEEK 2 (Aug 24 – Aug 29) Chapter 2: Algorithm Analysis

Algorithm Analysis (2.2, 2.3)
Runtime Analysis Examples (2.4)
Runtime Analysis Examples (2.4)

WEEK 3 (Aug 31 – Sep 5) Chapter 3: Lists, Stacks, and Queues

Abstract Data Types & Java Type Hierarchy (3.1, 3.2, 3.3)
Linked-Lists (3.5)

WEEK 4 (Sep 9 – Sep 11) Chapter 3: Lists, Stacks, and Queues & Chapter 4: Trees

Sep 7: Labor Day Holiday - University Closed
Stacks: Data Structure and Applications (3.6)
Queues: Data Structure and Applications (3.7)
Trees & Binary Trees (4.1, 4.2)

WEEK 5 (Sep 14 – Sep 19) Chapter 4: Trees

Tree Operations (4.3, 4.6)
Balanced Binary Trees: AVL Trees (4.4)
Balanced Binary Trees: Red-Black Trees (section 12.2)

WEEK 6 (Sep 28 – Oct 3) Chapter 4: Trees

Feb 16-17: Mardi Gras Holidays - no classes
Self-adjusting Binary Trees: Splay Trees (4.5)
External Searching: B-Trees (4.7) (PP)
Ordered Maps: Map as Tree (4.8) (PP)

WEEK 7 (Oct 5 – Oct 9) Midterm & Intro to Chapter 6: Priority Queues
Review
Midterm
Heaps (6.1, 6.2, 6.3)

WEEK 8 (Oct 12 – Oct 14) Chapter 5: Hashing
Oct 15,16: Mid Semester break
Unordered Maps: Map as Hash Table (5.1, 5.2, 5.3) (PP)
Open Addressing Hash Tables (5.4)
Enhanced Hashing Functions (5.5, 5.7, 5.8) (PP)

WEEK 9 (Oct 19 – Oct 23) **Chapter 6: Priority Queues**

Priority Queues & Order Statistics (6.4)
Enhanced Heaps: Leftist Heap & Skew Heap (Self-adjusting Leftish Heap) (6.6, 6.7) (PP)

WEEK 10 (Oct 26 – Oct 30) **Chapter 6: Priority Queues & Chapter 7: Sorting**
Forests: Binomial Queue (6.8) (PP)
Insertion Sort & Shell Sort (Generalized Insertion Sort) (7.1, 7.2, 7.3, 7.4) (Partly PP)
Selection Sort & Heap Sort (Enhanced Selection Sort) (7.5) (Partly PP)

WEEK 11 (Nov 2 – Nov 6) **Chapter 7: Sorting & Chapter 8: The Disjoint Set Class**
Optimal Sorting: Merge Sort & Quick Sort (7.6, 7.7)
Linear Sorting: Bucket Sort & Radix Sort (7.11) (PP)
Equivalence Relations & Disjoint Sets (8.1, 8.2, 8.3) (PP)

WEEK 12 (Nov 9 – Nov 13) **Chapter 8: The Disjoint Set & Chapter 9: Graph Algorithms**

Disjoint Set Algorithms: Union-Find & Path Compression (8.4, 8.5) (PP)
Graphs & Topological Sort (9.1, 9.2) (PP)
Shortest Paths: Dijkstra & Floyd-Warshall (9.3)

WEEK 13 (Nov 16 – Nov 20) **Chapter 9: Graph Algorithms**

Max-Flow Algorithm (9.4) (PP)
Minimum Spanning Trees: Prim & Kruskal (9.5)
Graph Applications (9.6) (PP)

WEEK 14 (Nov 23 – Nov 25) **Chapter 9: Graph Algorithms & Chapter 10: Algorithm Design Techniques**

Nov 26-27: Thanksgiving holidays
Graph Applications and NP-Completeness (9.6, 9.7)

WEEK 15 (Nov 30 – Dec 4) **Chapter 10: Algorithm Design Techniques**

Dec 4: Last day of class
Greedy Algorithms: Huffman Codes, Tries, & Data Compression (10.1)
Divide & Conquer: Matrix Multiplication (10.2) (PP)
Dynamic Programming: All Pairs Shortest Path (10.3) (PP)
Review

WEEK 16 (Dec 7 – Dec 11) **FINAL Exam**

Date, Time and Location: To be announced

*PP indicates - Parachute Points (means will be covered if time permits)