SYLLABUS

Required:

Text:

The following *required* text can be purchased at the Rutgers (Ferren Mall) bookstore:

Data Structures Outside In with Java, 1st Edition.
Sesh Venugopal
Prentice-Hall, 2006. ISBN 978-0131986190.

The programs in the textbook are in dsoi_progs_src.zip. The HTML documentation for all the programs is in dsoi_progs_doc.zip. These are attached at the end of the syllabus.

Pre-requisite Reading

You will be expected to hit the ground running with all the topics you learned in 111 - strings, arrays, searching, sorting, recursion, Big O, objects. In order to review objects and Big O in particular, you are urged to read the following from the text:

Chapter 1: Object-Oriented Programming in Java - Sections 1.1 and 1.2

Chapter 3: Efficiency of Algorithms - Entire chapter, all sections

Eclipse

You will be using Eclipse for all assignments in this class. If you don't know how to use Eclipse, read the "Eclipse" page in this site - see the link in the navigation bar on the left.

Part I: Linear Structures

In this first part, you will study linked lists (and a couple of variations of the basic linked lists), exceptions, array lists, stacks and queues. Stacks and queues are specialized structures that can be built out of either arrays or linked lists, or "smart arrays", which are arrays that grow automatically on demand.

You will learn how to analyze the time and space efficiency of the stack and queue, and compare them across various implementations.

We will discuss the following specific topics. Each topic is listed with references to reading in the textbook:

1. **Linked Lists**
   - Linked Lists and Circular Linked Lists [Sec 4.5,4.6]
   - Doubly Linked Lists [Programming Problems P4.14, P4.16]
2. **Exceptions** [1.5]
3. **Generic Types**
4. **ArrayList**
5. **Complexity Order/Running Time and Space**
   - Basic Operations, Input Size, Asymptotic Growth of Functions [Sec 3.2,3.3,3.4]
   - Order of complexity and Big Oh, Worst case and average case analyses [Sec 3.5,3.6]
6. **Stack**
   - Properties, applications, and class [Sec 7.1,7.2.1, 7.3]
   - Implementation using array list or linked list [Sec 7.5]
7. **Queue**
   - Properties, applications, and class [Sec 6.1, 6.3]
   - Implementation of bounded queue using array list, and unbounded queue using linked list [Sec 6.5]

Part II: Search Structures

Starting with arrays and linked lists, you will learn increasingly specialized and sophisticated structures that are tailored for efficient searching. The emphasis is on understanding the connection between the design of the structure and the speed of search.

When speaking of search structures, we are concerned not only with searching, but also with adding (inserting) and removing (deleting) data from the structure. So, the speed of inserts and deletes becomes important as well.

1. **Array and Linked List: Sequential Search**
   - Sequential Search and Complexity Analysis [Sec 4.2]
2. **Sorted Array: Binary Search**
   - **Binary Search and Complexity Analysis** [Sec 5.2]
   - **Comparison Tree** [10.1]
3. **Binary Search Tree (BST)**
   - BST Properties [Sec 10.2]
   - Search, Insert and Delete [Sec 10.3]
   - Recursive Inorder Traversal of BST and Treesort [Pg. 362, Sec 10.2, 10.5.1]
4. **Height-balanced BST: AVL Tree**

- AVL Tree properties [Sec 10.7.1]
- AVL Search [10.7.2]
- AVL Insert [10.7.3, 10.7.4]
- Insertion general scenarios [10.7.7 Pg. 335 figures (a) and (b)]

5. **Hash table**
   - Hashing [Sec 12.1,12.2]
   - Collision resolution: chaining [Sec 12.3.3]
   - java.util.HashMap class [Sec 12.4]

Part III: Binary Tree and Applications

1. **Traverals**
   - Inorder, preorder and postorder traversals [Sec 9.2]
2. **Huffman Coding**
   - Coding and compression, Algorithm, Average code length and prefix property [Sec 9.5.1, 9.5.2, 9.5.3]
3. **Heap**
   - Heap as Priority Queue [Sec 11.1,11.2,11.3]
   - Implementation using array list [Sec 11.7]
   - Building a heap out of a pre-existing set of objects with priorities [Sec 13.3.1]

Part IV: Graphs

1. **Types of Graphs and Representation**
   - Directed and undirected graphs, weighted graphs [Sec 14.1]
   - Adjacency matrix and adjacency linked lists representations [Sec 14.2]
2. **Graph Traversals**
   - Depth-first traversal (DFS) [Sec 14.3]
   - Breadth-first traversal (BFS) [Sec 14.3]
3. **Graph algorithms**
   - Topological sort [Sec 14.4]
   - Dijkstra's Shortest Paths [Sec 14.6]

Part V: Sorting

1. **Insertion sort** [Sec 13.1, skip average analysis)]
2. **Quicksort** [Sec 13.2.1]
3. **Mergesort** [Sec 13.2.2]
4. **Heap sort** [Sec 13.3]

dsoi_progs_src.zip
dsoi_progs_doc.zip

POLICIES

Course Coordination

There are 3 lectures in this course, taught by professors Sesh Venugopal, and Ana Paula Centeno. The course will be coordinated by Prof. Venugopal. Students in all lectures will have the same exams and same assignments, and will be graded under the same set of rules.

Grading

The distribution of points that will be used in determining the letter grade is as follows, out of a total of 1000 points:

PROGRAMMING ASSIGNMENTS: 350
RECITATION PROBLEM SETS 50
MIDTERM EXAM 1: 150 (SUNDAY OCT 9, 8:10pm-9:40pm)
MIDTERM EXAM 2: 150 (SUNDAY NOV 13, 8:10pm-9:40pm)
FINAL EXAM: 300 (FRIDAY DEC 16, 4pm-7pm)

Programming Assignments

There will be five required assignments.

One Time Extension Pass

You get a ONE time extension pass for the semester. You can use the pass to submit any ONE of the programming assignments until the extension deadline, which will be 72 hours after the regular deadline. A separate Sakai assignment will be opened for extensions **AFTER** the deadline for the regular submission has passed.

Special Notes

Your best strategy for working on assignments is to start early, and hand in something **at least** a full day before the deadline. You can continue working on the assignment and submit as many updated versions of your assignment as you want before the deadline.

Late assignments will NOT be accepted, even if they are late only by a minute. You have plenty of time to work in each assignment, and you are responsible for planning and managing your time.

You are responsible for making sure the program compiles before you hand it in. **If it does not compile, your program will get a zero, whatever the reason for it not compiling. Note that you are REQUIRED to use Eclipse. If you don't use Eclipse, and you make your program compile by playing around with the package amd import statements, it will NOT compile when we test it, and you WILL be given a zero.**

When you are working on your assignment, you can get help from your TAs or from your instructor at any time, so there is no reason for you to get into a situation where you don't submit your assignment, or your program does not compile.

Exams

There will be two midterm exams and one final exam. They will both be closed text and closed notes written exams (paper and pencil, NOT on computer). See dates in the Grading section above.

There is NO pre-scheduled general makeup exam. A makeup exam may be given to specific individuals only on the basis of a **legitimate**, **documented** conflict, or a **university approved** reason, or any other emergency that is deemed a legitimate reason for taking a makeup.

Problem Sets/Recitation

At the end of every week of class (except the first week), a problem set will be posted on this site, to be covered in recitation the following week. For every recitation (except the first), you will be asked to solve one problem in the problem set for that week, and turn in your solution during recitation. This will count for 5% of your course grade.