Expand All Print View

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.

Clicker:

You are *required* to use a clicker in lecture. **(iClicker 2 or iClicker 1)**. Note that quizzes will be given in lecture that you will answer using your clicker. These quizzes will count toward 10% of your course grade. (See **Policies** section.)

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 including Comparison Tree** [Sec 5.2, 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