Topics for CSCI 151 Final Exam
Wednesday, May 10

Java and Programming Techniques

- Types
- Inheritance
- Generics
- Abstract classes and interfaces
- Exceptions
- Recursion
- Writing recursive methods
- Dynamic Programming

B. Data Structures

- ArrayLists
- Linked Structures, especially singly and doubly linked lists
- Stacks
- Queues
- Binary Search Trees
- AVL trees
- Heaps and Priority Queues
- Hashing, Hash Tables, and Hash Maps
- Tries
- Graphs

Algorithms

- Big-Oh etc. notation Upper bounds and lower bounds
- Algorithm analysis
- BubbleSort, SelectionSort, InsertionSort
- Lower bound for sorting
- MergeSort, QuickSort, HeapSort
- Insert, Search, and Remove algorithms for each of our data structures
- Shortest Path algorithms for a directed graph
- Topological Sort for a directed graph
- Minimum Spanning Tree for an undirected graph
- Disjoint sets ~~and Clustering~~

- For each data structure you should know how it is implemented, how it works, what it is good for, and a Big-O estimate of its running time.
- There will not be any LONG programs to write, but you could be asked to write short pieces of code.
- There is likely to be at least one recursive function to write.
- It is VERY likely that there will be a question asking you to insert values into an AVL tree – too many people got that wrong on Exam 2.
- In general I am more interested in whether you know how our data structures work and how they can be used than whether you can code their methods during the exam.

Here are some typical questions:

1. Java has an AbstractList class and a Comparable interface. Why is one an abstract class and the other an interface?

2. Give an algorithm for deleting a node in a binary search tree.

3. I have a large collection of n data items already stored in an AVL tree. I want to use this tree as a Priority Queue. How long will each of these queue operations take?
   - Peek (i.e., find the maximum element)
   - Poll (find, remove and return the maxiumum element)
   - Offer (x) (insert x into the queue)

4. A grocery store hires you to maintain their inventory in a database. You need to use as keys the Universal Product Code stored in the barcode. The UPC consists of 12 numeric decimal digits, such as the following one for Kellogg's Raisin Bran: 038000596636. The value corresponding to each key is the number of items of this product that you have on hand. Here are some options:
   - You could store the counts in an array indexed by the UPC; for example, if the store has 20 boxes of Kellogg's Raisin Bran the entry at index 38000596636 would be 20.
   - You could make an array of base type Entry, which has a UCP-count pair, with an entry for each of the 10,000 products stocked in the store.
   - You could make a HashMap where the keys are UPC's and the values are Integer counts.
   - You could make a TreeMap where the keys are UPCs and the values are Integer counts.
   - You could make a trie out of the UPC codes.

Which would you use, and why?

5. Here is a Node type for a binary search tree that holds integer data:

```
class Node {
        int data;
        Node leftChild, rightChild;
}
```

Give a definition of a recursive function Node insert(int x, Node p) that inserts value x into the tree rooted at p.

6. I have a large collection of n data items already stored in an AVL tree. I want to use this tree as a Priority Queue. How long will each of these queue operations take?
    - Peek (i.e., find the maximum element)
    - Poll (find, remove and return the maximum element)
    - Offer (x) (insert x into the queue)

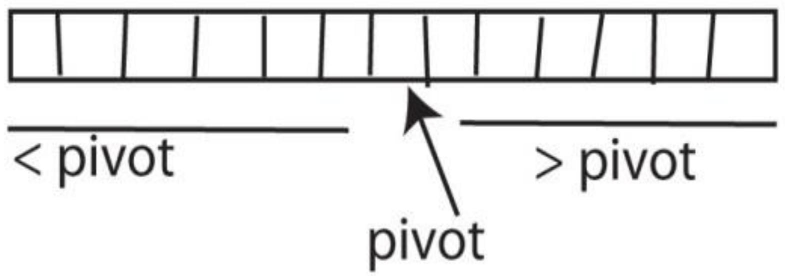
7. Here is a recursive function:

```
int f(int n)  {
        If (n == 0)
                return 1;
        else if (n == 1)
                return 3;
        else
                return 2*f(n-1)  - 3*f(n-2) + 5;
}
```

Give a dynamic programming version of f.

8. Your boss gives you a dataset of 1,000,000 items. He wants you to find the 100 largest values in this dataset as quickly as you can. If your program runs quickly enough you get promoted and get to take that dream vacation to Fiji. If your program runs too slowly you get fired and go to work at MacDonalds. What technique will you use to find the 100 largest values? What is your estimate of its running time in terms of the number of items in the dataset?

9. Here is an algorithm for finding the kth smallest element in an unsorted array of values. It works much like Quicksort – we choose a pivot value and rearrange the data so that all of the values less than the pivot are to its left and all of the values greater than the pivot are to its right:



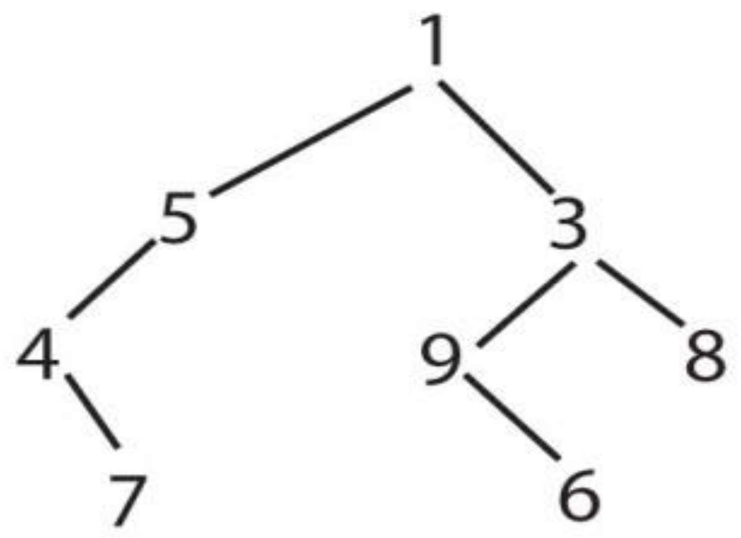
If the index of the pivot > k recurse on the portion of the array between index 0 and the index of the pivor. If the index of the pivot < k look for the k-index(pivot) smallest value in the portion of the array to the right of k. Of course, if index(pivot) == k you are done.

- What is the big-Oh worst-case running time of this on an array of size n/
- What is the average case running time on an array of size n?
- Give an argument that your answer for (ii) is correct.

10.I have a tree based on the Node class

```
class Node {
        int data;
        Node leftChild, rightChild;
}
```

Give a procedure void Print(Node p) that prints a "breadth-first" traversal of this tree. For the tree



It prints 1 5 3 4 9 8 7 6