Midterm Review

Structure of the Exam. You will have 2 hours 15 minutes for the exam. The exam will be given in-person in a classroom or hall that you will be told about. All your work will be written directly on the exam paper. You will be given reference sheets for the Master Formula and for calculus formulas.

1. Multiple Choice and True/False 25%
2. Short Answer 25%
3. Long Answer 50%
4. SCI (3 points)

Review Points

1. Be familiar with the file Problems.pdf that we began the course with. You should know an optimal solution for each problem and its running time, and you should have a general idea of how each one is solved using the optimal algorithm (optimal so far in the class). Also, you should know the running times of the optimal solutions to other problems discussed in class.
2. Know the MergeSort, QuickSort, and QuickSelect algorithms. Be able to work through examples of these by hand (as in slides) -- you should use recursion trees to demonstrate the behvaior of these algorithms.
3. Be familiar with the fact that performance of QuickSort and MergeSort can be enhanced by incorporating InsertionSort to handle small sections of the input list.
4. Know what *good pivots* and *good self-calls* are, in the context of analysis of QuickSort and QuickSelect.
5. Know the lower bound theorem for comparison-based sorting algorithms. In particular, be able to use the result that every comparison based sorting algorithm, running on an input array of size n, requires at least $\lceil \log n! \rceil$ comparisons in the worst case.
6. Be familiar with the ways to determine the running time of a recursive algorithm. Approaches:
    a. Set up a recurrence relation and solve using Master Formula
    b. Count self-calls
    c. Determine a lower bound (when you wish to show the algorithm is at least exponential)

    Often, to be able to count self-calls, you need to know the formulas for counting the number of terms in a sequence n, n/2, n/4, . . . , 1, 0. (Recall: The number of terms required to arrive at 1 is $1 + \lfloor \log n \rfloor$, and to arrive at 0 is $2 + \lfloor \log n \rfloor$.)
7. Know the definition of *inversion-bound sorting algorithm* and the fact that the average case running time for inversion-bound algorithms running on length-n input arrays of distinct elements is $\Omega(n^2)$

8. Be familiar with the inversion bound algorithms BubbleSort, SelectionSort, InsertionSort. You should also know that LibrarySort is a generalization of InsertionSort that has average case running time O(nlog n).
9. Know the limit definitions of O, o, Θ, and Ω. Be able to determine when a function belongs to one of these complexity classes. Be sure to be familiar with L'Hopital's Rule for evaluating limits
10. Be familiar with the BinarySearch algorithm and be sure to know its running time (as computed in Lesson 3).
11. Be familiar with BucketSort; be able to carry out the steps to solve a sorting problem using this method and to give a running time analysis, and also be able to tell when BucketSort is not efficient.
12. GCD Algorithm. You should know the two-line algorithm. You do not need to reproduce the runtime analysis or proof of correctness. However, you should appreciate these elements of that analysis:
    a. Thinking recursively
    b. Proving correctness of a recursive algorithm
    c. Computing running times by counting self-calls.
    d. Formulas for number of terms in a descending sequence
    e. The use of mathematics facts to obtain more efficient algorithms (in this case, the fact that lcm(m,n) * gcd(m,n) = mn).
13. Be familiar with the IsPrime algorithm discussed in class. You should be able to write the code/pseudo-code for the basic version of this algorithm and you should know its running time.
14. Know how to prove correctness of an iterative algorithm (including use of a loop invariant) and of a recursive algorithm.
15. Be familiar with the proof that the fib(n) algorithm for computing the nth Fibonacci number has at least exponential running time. (This includes an understanding of the definition of "at least exponential running time".)
16. Know that Hashtable performs its operations in O(1) time (average case).
17. Know what the *loadFactor* is for a hashtable; know the steps for performing a rehash on a hashtable.
18. *Binary Trees.* You should know:
    a. The structure of the Node class (including possibly a reference to parent)
    b. DFS traversal. You should know the pattern that DFS follows to visit every node. You should also know the pseudocode for the two implementations of DFS for binary trees discussed in class: Preorder Traversal and Inorder Traversal; you should also know the order in which nodes are processed in each of these traversal implementations.
    c. The definition of a binary search tree and the worst-case and average-case running times of its main operations. How is Inorder traversal used in a BST? You should be able to build a BST starting from an empty tree and a given insertion sequence.
19. Be familiar with basic results in probability, as discussed in class. In particular, you should remember these theorems:

Theorem [*Expected number of trials for success*]. Suppose an experiment is performed repeatedly, and the probability of "success" is p (where p is a real number between 0 and 1), and is not influenced by previous successes or failures. Then the expected number of trials to obtain a success is $\frac{1}{p}$.

Theorem [*Expected number of trials for k successes*]. Suppose k is a positive integer. Suppose an experiment is performed repeatedly, and the probability of "success" is p (where p is a real number between 0 and 1), and is not influenced by previous successes or failures. Then the expected number of trials to obtain k successes is $\frac{k}{p}$.

20. **SCI.** You will be asked to write a short essay elaborating upon parallels and connections between points from the field of Algorithms and SCI. Fuller elaborations will be awarded more credit. (A good resource to prepare for this is the last half of the Syllabus, available on Sakai.)