Review for Algorithms Midterm

Overview

The midterm consists of four parts: complexity analysis, skills/analysis, and sorting, in addition to an SCI question.

Each part might include multiple-choice questions. Multiple-choice questions require some thoughts and no partial credit will be awarded for them if no justification/rational is required. If the question requires justification, you may get partial credit. Analysis and skills problems may require you to perform some kind of analysis of the running time of an algorithm, or to create your own algorithm to solve a problem.

There might be questions that ask you to obtain an asymptotic running time, using the Master Formula or other methods, given that the running time satisfies a typical recurrence relation; the Master Formula will be provided to you in the exam (no need to memorize it).

Not covered on the exam

1. I will not ask you to do proofs of **correctness of algorithms**

2. I will not ask you to reproduce proofs given in the lectures

3. There will be no questions about the GCD algorithms

Things you should know for the exam

1. Be able to show that a function belongs to a particular complexity class using the definition of O, Θ, or Ω. Example: Show that $2n^2-1$ belongs to $\Theta(n^2)$. Some questions might require you to work with the definition directly.

2. Know the relationships between the most common complexity classes O(1), O(log n), $O(n^{1/k})$ ($k > 1$), O(n), O(n log n), $O(n^k)$ ($k > 1$),$O(2^n)$, O(n!), $O(n^n)$

3. Be familiar with the iterative and recursive versions of the factorial and Fibonacci algorithms and how to compute their running times. There are questions require similar skills.

4. You should understand what inversion bound sorting algorithms are, and that InsertionSort, SelectionSort, and BubbleSort are examples of these.

5. You should know the relative strengths and weaknesses of the main sorting algorithms that were discussed: InsertionSort, SelectionSort, BubbleSort, LibrarySort, MergeSort, QuickSort, BucketSort, RadixSort. You should be able to decide which of these would be the best choice to solve different kinds of sorting problems.

   You should know the average case and worst case running times of all of these.

6. You should be familiar with the pseudo code for MergeSort and QuickSort including the partition algorithms for each and the merge step. You should be familiar with different pivot selection strategies that could be used in QuickSort. (The lectures emphasized the strategy of selecting pivots at random; this made average case analysis go smoothly. But two more approaches are mentioned in lectures.)

7. You should be familiar with the pseudo code for SelectionSort and BubbleSort.

8. You should have a good idea about how the running times for MergeSort and QuickSort are established. You should also know the worst-case running time for QuickSort occurs extremely rarely, and, roughly, why this is the case. You will not need to prove any of these things on the test, but you should understand these analyses well enough to answer questions about them. For instance, you should understand what a "good pivot" is in the analysis of QuickSort.

9. SCI Question. There will be one SCI question worth 5 points. You will be able to pick your own topic and elaborate on a parallel between SCI and principles underlying analysis and development of algorithms.

Points Distribution

Question #	**1** **14 pt**	**2** **14 pt**	**3** **12 pt**	**4** **5 pt**	**Total Grade**
Grade (45 points)	Complexity Analysis	Analysis	Sorting	SCI	