Review for Algorithms Midterm

Spring 2015 Algorithms DE

Overview

The midterm consists five questions. The questions cover complexity analysis, recurrences, algorithms analysis, sorting, and SCI.

Complexity analysis section tests your understanding for complexity notations, and your skills in finding algorithm's complexity…etc.

Recurrences questions test the concepts of divide and conquer strategy, master formula, analysis of divide and conquer algorithms, …etc.

Algorithms Analysis questions test your skills in designing efficient algorithms for given problems.

Sorting questions test your understanding for sorting concepts and sorting algorithms.

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 algorithm

Things you should know for the exam

1. Be able to express an algorithm using the pseudo-code syntax developed in Lesson 2. Be able to implement pseudo-code in Java.

2. 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 $\Theta(n^2)$. You should know how to work with the definition directly.

3. 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!), $O(n^n)$

4. 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.

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

6. 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.

7. 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.)

8. You should be familiar with the pseudo code for Bubble sort.

9. 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.

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