Two hours

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

Advanced Algorithms I

Date: Thursday 27th January 2011
Time: 14:00 - 16:00

Please answer THREE Questions: ONE Question from EACH Section
Use a SEPARATE answerbook for EACH Question

For full marks your answers should be concise as well as accurate
Marks will be awarded for reasoning and method as well as being correct.

This is a CLOSED book examination

The use of electronic calculators is NOT permitted

[PTO]
Section A

1. String searching.
   a) Describe the Knuth-Morris-Pratt (KMP) string searching algorithm.
      i) You should describe in detail the principles of the preprocessing phase and of the searching phase. You need not write programs or pseudocode but you should explain in detail how each phase works. (8 marks)
      ii) What is the worst-case time complexity of the searching phase of the KMP algorithm? Explain your answer. (2 marks)
   b) Consider now string searching based on moving the pattern left-to-right through the text, but matching the pattern to the text from right to left.
      i) A simple technique is as follows: when a pattern character fails to match a text character, to use the text character to determine another possible alignment by preprocessing the pattern. Describe clearly this simple algorithm, both its preprocessing phase and its matching phase. You may give a program, a pseudocode presentation, or a description in which you explain the details of each step clearly. (6 marks)
      ii) Notice that when a fail occurs in the matching, the pattern must move at least one character to the right (unless it is at the end of the text), so the text character next on the right to the failed character must be involved in the matching. Explain how your simple algorithm may be modified, using this observation and a suitable preprocessing of the pattern, to produce an improved algorithm. (4 marks)
2. Graph algorithms.

a) Describe a generic algorithm for performing both Depth-first Search (DFS) and Breadth-first search (BFS) on finite trees. You should present your algorithm as a program or in pseudocode and explain it. (4 marks)

b) Explain clearly how to modify this algorithm to perform DFS and BFS on finite directed graphs. (3 marks)

c) A topological sort of a finite directed graph is a list of the nodes of the graph (each node occurring once), ordered so that if there is an edge from node \( i \) to node \( j \) in the graph, then \( i \) precedes \( j \) in the list.

i) Explain why a directed graph which has a topological sort must be acyclic. (2 marks)

ii) Explain clearly why every finite directed acyclic graph has a node with out-degree zero. (2 marks)

iii) Give an algorithm to construct a topological sort of a finite acyclic graph. [Hint: one approach is to use the above property of acyclic graphs.]. You need not give a program, but you should explain clearly the steps of your algorithm. (4 marks)

d) Describe ONE NP-complete task on directed, or undirected, graphs, and an exponential-time algorithm for the task. You need not give a program, but a clear step-by-step explanation is required. (5 marks)
3. Asymptotic Analysis

a) Formally define what is meant by saying that \( f(n) \) is \( O(g(n)) \). (4 marks)

b) Given that \( f_i(n) \) is \( O(g_i(n)) \) prove that:
   i) \( f_1(n)f_2(n) \) is \( O(g_1(n)g_2(n)) \). (4 marks)
   ii) What can you say about \( f_1(n) - f_2(n) \)? Prove your hypothesis. (4 marks)

c) Where does \( O(n!) \) fit into the exponential heirarchy? Using Stirling’s Formula – which you may quote – prove your conjecture. (8 marks)

4. Recurrence Relations

a) Explain the “Method of Moments” for solving linear recurrence relations. (4 marks)

b) Find a closed form for the recurrence relation \( T(n) \)
   \[
   T(0) = 0 \\
   T(1) = 1 \\
   T(n+2) = T(n)
   \]
   If you haven’t used the method of moments, prove your conjecture (6 marks)

c) Find a closed form for the recurrence relation \( T(n) \)
   \[
   T(0) = 0 \\
   T(1) = 1 \\
   T(n+2) = T(n+1) + T(n)
   \]
   If you haven’t used the method of moments, prove your conjecture (10 marks)
5. a) Explain informally what is meant by a polynomial-time approximation algorithm, giving two examples of why and when one might look for, or use, such an algorithm. (3 marks)

b) Define, in formal terms, what is meant by a factor $f$ approximation algorithm. (2 marks)
c) What is a Steiner tree? Describe the minimal Steiner tree problem. (3 marks)
d) What is a metric Steiner tree? (2 marks)
e) Justify why an algorithm for the minimal spanning tree over the required vertices of the Steiner tree will give a factor 2 approximation for the minimal Steiner tree. Hint: You may assume that there is an approximation factor preserving reduction from the Steiner tree problem to the metric Steiner tree problem. (4 marks)
f) A full Steiner tree over $n$ required vertices has $n - 2$ Steiner vertices, where the degree of each required vertex is 1 and the degree of each Steiner vertex is 3. For minimal (full) Steiner trees on a Euclidean metric, i.e. the vertices are points in Euclidean space, any two edges incident on a Steiner point meet at 120°. Give a geometric construction for the Steiner point for the case when $n = 3$ and the points form an acute-angled triangle, and then indicate how the general result can be used to construct a good approximation for the minimal Steiner tree in polynomial time. (6 marks)

6. a) Informally explain what is meant by a randomised algorithm. (2 marks)

b) Describe an example of
   i. a Monte Carlo algorithm, and
   ii. a Las Vegas algorithm,
   explaining, in particular, the different characteristics of each type. (8 marks)
c) You are given a Monte Carlo algorithm for solving problem $\Pi$ with expected time $T(n)$ for any instance of size $n$, which yields a correct solution with probability $\gamma(n)$, together with a solution checker, which will verify correctness in time $t(n)$. How can you construct a Las Vegas algorithm for $\Pi$ that always gives a correct answer in expected time $\frac{T(n) + t(n)}{\gamma(n)}$. (4 marks)
d) Describe the basic ideas underlying an improvement that can be made to the simple randomised algorithm for computing candidate minimum cuts of a multigraph (min-cut algorithm), which will run in $O(n^2 \log n)$ time and have probability of $\Omega(\frac{1}{\log n})$ of finding a minimum cut. (6 marks)