Two hours

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

Advanced Algorithms

Date: Monday 8th June 2009

Time: 09:45 – 11:45

Please answer any THREE Questions from the FIVE questions provided

Each question carries equal marks

This is a CLOSED book examination

The use of electronic calculators is NOT permitted.
1. a) Describe the Knuth-Morris-Pratt algorithm for string searching based on moving a pattern left-to-right through the text and matching the pattern to the text from left to right. Your description should explain:

i) The pre-processing phase using the pattern alone,

ii) The matching routine,

iii) The time complexity of the algorithm and the circumstances in which it is an efficient choice. (10 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.

A simple technique is as follows: when a pattern character fails to match a text character, to use this text character to determine another possible alignment by pre-processing the pattern.

Describe clearly this simple algorithm, both its pre-processing phase and its matching phase. Under what circumstances is it an efficient choice of algorithm? (6 marks)

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 above may be improved using this observation and a suitable pre-processing of the pattern. (4 marks)
2. For each of the following problems on graphs:
   a) State the smallest time complexity class containing the problem,
   b) Describe an algorithm for the problem showing that it is in this complexity class,
   c) Explain clearly why the algorithm is correct.

Your algorithms may be presented as programs, or in pseudocode, or as clear step-by-step descriptions. You should say what representation of graphs is being used.

   i) Determining the number of connected components in undirected graphs. (5 marks)
   ii) Cycle detection in directed graphs, i.e. finding whether or not a directed graph has a non-empty path from a node to itself. (5 marks)
   iii) Constructing Eulerian circuits, where possible, in undirected graphs, i.e. finding whether or not a graph has a cycle visiting all nodes and passing through each edge just once and, if so, returning such a cycle. (5 marks)
   iv) Determining whether an undirected graph may be coloured with \( k \)-colours and, if so, returning such a colouring. (5 marks)
3. a) In the context of formal program verification, explain briefly and clearly what is meant by the term *model checking*. What are the features of model checking that have contributed to its success? (3 marks)

b) Complete and explain the components of the formal definition of a Kripke structure

\[ K = (S, S_0, R, I) \]

where

- \( S \) is …
- \( S_0 \) is …
- \( R \) is …
- \( I \) is … (4 marks)

c) Explain how a Kripke structure can be used, in general, to model a shared variable concurrent program. As an example, construct a Kripke structure for the following program

```plaintext
1:   while (true) do
2:   {
3:     L1: y = (x+1) mod 2;
4:     L2:
5:   ||
6:     R1: x = (y+2) mod 3;
7:     R2:
8:   }
```

in which the assignment statements from lines 3 and 6 are executed concurrently, but atomically, within the body of the while loop. You may find it helpful to assume a program counter pc that takes values according to the following table

<table>
<thead>
<tr>
<th>Label Pairs</th>
<th>pc Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1, R1</td>
<td>1</td>
</tr>
<tr>
<td>L2, R1</td>
<td>2</td>
</tr>
<tr>
<td>L1, R2</td>
<td>3</td>
</tr>
<tr>
<td>L2, R2</td>
<td>1</td>
</tr>
</tbody>
</table>

The variables \( x \) and \( y \) are restricted to finite integers, in the range 0 to 2, and have initial values 0. (7 marks)

d) Explain what is meant by a graph *reachability* problem and then outline a *forwards reachability* algorithm. Describe how such an algorithm can be used to establish whether particular “bad” states may occur in a program’s execution. Hence, or otherwise, establish that the above program never assigns the value 1 to \( y \) and 1 to \( x \). (6 marks)
4. a) Briefly explain the concept of a Büchi automaton, represented by the structure

\[ BA = (S, \Sigma, T, S_0, A), \]

in particular stating what the components \( S, \Sigma, T, S_0 \) and \( A \) represent, and what the acceptance condition is for an infinite word. (4 marks)

b) Create a Büchi automaton that accepts exactly the language \( L \), defined by the following \( \omega \)-regular expression

\[ L = (a^*bc)\omega \]

Present the automaton as a diagram making sure you clearly mark the initial and accepting states as well as clearly labelling the transitions. (2 marks)

c) What are strongly connected components of a graph? Outline an algorithm for determining terminal maximal strongly connected components and then show how such an algorithm can be used to determine the emptiness, or otherwise, of a given Büchi automaton. (6 marks)

d) Construct a Büchi automaton that corresponds to the product of the Büchi automaton for the language \( L \), from part (b) above, with the Büchi automaton corresponding to the language \( M \) defined by

\[ M = (b^*c)\omega \]

(4 marks)

e) Use your algorithm from part (c) to show that the product automaton obtained in answer to part (d) is non-empty. (4 marks)
5. a) What is meant by a polynomial-time reduction of one computational task to another? (3 marks)

b) Define the following complexity classes:

i) P – polynomial time-bounded,
ii) NP – non-deterministic polynomial time-bounded,
iii) NP-complete. (4 marks)

c) Explain clearly how a polynomial-time reduction may be used to establish the NP-completeness of a computational task from a task known to be NP-complete. (3 marks)

d) Using the NP-completeness of the Boolean satisfiability problem, describe in detail a proof of the NP-completeness of a problem of your choice on graphs using a polynomial-time reduction. (10 marks)