One and a half hours

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

Algorithms and Imperative Programming

Date: Monday 17th January 2011
Time: 14:00 - 15:30

Please answer any TWO Questions
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
1. Algorithm design.

For each of the following computational tasks

(i) describe an algorithm for the task. Your description may be a program in a standard language, in pseudocode, or as a clear and precise step-by-step description. You should explain your algorithm. A correct algorithm is required. However, some marks will be awarded for an algorithm which is not only correct, but also is efficient.

(ii) give the worst-case time complexity of your algorithm in terms of the size of the input and the number of operations required. Explain your answer.

a) Given a list of integers as input, determine whether or not two integers (not necessarily distinct) in the list have a product $k$. For example, for $k = 12$ and list $[2, 10, 5, 3, 7, 4, 8]$, there is a pair, 3 and 4, such that $3 \times 4 = 12$. (6 marks)

b) List intersection: Given two lists of integers, compute a list of integers which consists of those integers which are in both lists (the order of the intersection list does not matter and, if numbers appear several times in the lists, the intersection need not reflect this multiplicity - though it may). Thus, given lists $[2, 5, 3, 8, 2, 4, 7]$ and $[6, 7, 2, 4, 9, 1]$, one possible intersection list is $[2, 7, 4]$. (7 marks)

c) A majority element in a list of integers of length $N$, is an element that occurs strictly more than $N/2$ times in the list. Determine whether or not a list of integers has a majority element. (7 marks)
2. Sorting

a) What is the lower bound on the time complexity for sorting using a comparison-based sort? (Your answer should be in Big-Oh notation).
   
   (1 mark)

b) What is the lower bound on the time complexity for sorting using a distribution-based sort (such as a bucket sort)? (Your answer should be in Big-Oh notation).
   
   (1 mark)

c) Explain why it is not practically possible to use bucket sort to sort arbitrary floating point numbers.

   (2 marks)

d) Merge Sort recursively divides up the input until it is in lists of length one. Merging of the lists proceeds from the bottom up. Each merge operation ensures that the output of the merge is sorted.

   i) Give pseudocode for the merge operation. The input should be two sorted lists of length \( N \) and \( M \) respectively, and the output must be a sorted list of length \( N + M \).

   (4 marks)

   ii) Derive the time complexity of the merge operation.

   (2 marks)

   iii) Explain whether a Merge Sort based on your merge operation will be stable. Explain why or why not.

   (2 marks)

e) Quicksort is another divide-and-conquer sorting algorithm.

   i) In Quicksort, the work of sorting is done in the ‘divide’ part of the algorithm. Explain how this is achieved.

   (3 marks)

   ii) One method of choosing the pivot is to use the value of the last element in the input. Another method is to take a random element. Explain one advantage and one disadvantage of using the last element method (compared to the random element).

   (2 marks)

   iii) Peter has implemented a version of Quicksort that uses Insertion Sort to sort all sublists of length less than ten. Insertion Sort has a time complexity of \( O(n^2) \) on most inputs. What will the time complexity of Peter’s algorithm be on most inputs, and why?

   (3 marks)
3. Complexity

a) Algorithms F and G are two different solutions to the same problem. Algorithm F has a time complexity \( f(n) \) that is \( O(n^2) \). Algorithm G has a time complexity \( g(n) \) that is \( O(n^3) \). Consider the following statements and decide if they are true or false. (4 marks)

i) It is certain that F is always faster than G, given the same inputs. (True or False)

ii) For small inputs (small \( n \)), G might run faster than F. (True or False)

iii) There is a value of \( m \), such that for all \( n > m \), \( f(n) < g(n) \). (True or False)

iv) Janet thinks of a way to improve G, so that it runs 10 times faster. This will affect its time complexity written in Big-Oh notation. (True or False)

b) Simplify the following Big-Oh expressions

i) \( O(n + 2n + 2) \) (2 marks)

ii) \( O(2n) \times O(n^2) \) (2 marks)

iii) \( O(n \log n + 3n^2) \) (2 marks)

c) A student does some experiments to estimate the running time \( t(n) \) of an algorithm for processing text inputs of length \( n \) characters.

Wishing to estimate the asymptotic time complexity, she runs the algorithm several times for different \( n \), and collects the following data.

<table>
<thead>
<tr>
<th>( n )</th>
<th>4</th>
<th>16</th>
<th>64</th>
<th>256</th>
<th>1024</th>
<th>4096</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean running time (microsecs)</td>
<td>62</td>
<td>286</td>
<td>2078</td>
<td>16414</td>
<td>131100</td>
<td>1048600</td>
</tr>
</tbody>
</table>

She makes a plot of this data, shown below.
i) Comment on the type of plot, and what can be inferred from it, if a straight line can be fitted to the points at larger values of $N$. (2 marks)

ii) Estimate the asymptotic running time of the algorithm from the plot. Show your working. (3 marks)

d) Here is an algorithm for computing the value of a polynomial $p(x)$ of the form

$$p(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + \ldots + a_nx^n$$

where the coefficients $a_i$ are given in an array $a[\ ]$ of length $n + 1$:

```c
polynom(a[ ], x)
{
    p=0;
    for(i=0 to n){
        y=1;
        for(j=0 to i-1)
            y = y * x;
        p = p + a[i] * y;
    }
    return p;
}
```

i) Express the total number of multiplications and the total number of additions done using big-Oh notation. Combine these and simplify to give the overall big-Oh of the algorithm. (2 marks)

ii) $p(x)$ can be rewritten as

$$p(x) = a_0 + x(a_1 + x(a_2 + x(a_3 + \ldots x(a_{n-1} + x.a_n)\ldots))$$

Give pseudocode for evaluating $p(x)$ using the above factorization. (2 marks)

iii) How many additions and multiplications are done by the new algorithm, in terms of $n$? (1 mark)