Two hours

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

Algorithms and Imperative Programming (CSE)

Date: Tuesday 24th May 2011
Time: 14:00 - 16:00

Please answer THREE Questions from the FOUR Questions provided

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 computational tasks (a), (b) and (c):

(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. In addition to marks awarded for a correct algorithm for each task, marks will also be awarded for devising efficient algorithms.

(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) Symmetric difference of lists: Given two lists of integers, compute a list of integers which consists of those integers which appear in either of the two lists but not in both. The order of the resulting list does not matter and, if numbers appear several times in the lists, the result 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 symmetric difference list is \([5, 3, 8, 6, 9, 1]\). (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. Complexity measures.

a) A computer program uses \( 3n^2 \) basic operations to process inputs of size \( n \). If \( N \) is the size of input that can be processed with \( R \) resources (e.g. units of time), what is the size of problem that can be tackled with \( 2R \) resources? Show your working. (2 marks)

b) i) Consider a function \( f(n) = n^3 + \log n \). Which of the following are true statements about its asymptotic growth rate? Indicate all that apply.

(A) \( f(n) \) is \( O(n) \)  
(B) \( f(n) \) is \( O(n^3) \)  
(C) \( f(n) \) is \( \Theta(n^3) \)  
(D) \( f(n) \) is \( O(n^4) \)  

(2 marks)

ii) A function \( f(n) \) is \( O(n \log \log n) \). List all of the statements below that are true.
(A) $f(n)$ grows faster than $O(n)$  

(B) Asymptotically, $f(n)$ grows more slowly than $n\sqrt{n}$

(C) $f(n)$ might be $2n \times 3\log\log n$

(D) $f(n)$ is exponential in $n$. 

(2 marks)

c) Consider the following list of eight functions of a variable $n$:

\[ n(\log_2 n)^2, \; (n/4)! + n, \; 100, \; n/2, \; \sqrt{4n}, \; 2n^2, \; n^4, \; n\log_2 n \]

Place the functions in increasing order of size asymptotically. (3 marks)

d) Here is C-like pseudocode for an algorithm to sort an array of $n$ integers into ascending numerical order.

\[
\text{sort}(a[], n)\{
    \text{smallest} := a[0];
    \text{largest} := a[0];
    \text{for each } i \text{ in } 1 \text{ to } n-1\{
        \text{if } (a[i] < \text{smallest}) \{ \\
            \text{smallest} := a[i]; \\
        \} \\
        \text{else if } (a[i] > \text{largest}) \{ \\
            \text{largest} := a[i]; \\
        \}
    \}
    \text{range} := \text{largest} - \text{smallest};
    \text{allocate memory for count array of size } \text{range}
    \text{for each } k \text{ in } 0 \text{ to } \text{range} - 1 \{
        \text{count}[k] := 0;
    \}
    \text{for each } i \text{ in } 0 \text{ to } n-1 \{
        \text{count}[a[i] - \text{smallest}]++; \\
    \}
    c:=0;
    \text{for each } k \text{ in } 0 \text{ to } \text{range} - 1 \{
        \text{while(count}[k]-- > 0) \\
        a[c++] = k + \text{smallest}; \\
    \}
\}
\]

i) Explain briefly how this sorting algorithm works and why it is correct. (3 marks)

ii) What is the space complexity of the algorithm? Do not count the space taken up by the input array $a[]$. Express your answer in Big-O notation and use appropriate variables. Explain your answer. (3 marks)

iii) What is the complexity of the algorithm in the number of integer assignments
iv) In a certain application of this sorting algorithm, the integers to be sorted are in the range 0 to approximately \(\log_2 n\), where \(n\) is the length of the list being sorted. Restricting to this type of input, will the sorting algorithm have a better asymptotic time complexity than a comparison-based sort such as mergesort? Explain your answer. (2 marks)

3. Consider the following number sequence:

1, 14, 9, 56, 16, 28, 7, 3, 11, 2.

We wish to insert these numbers in this order into a hash table of size 11 (with positions 0 to 10), using the hash function \(h(x) = (3x + 7) \mod 11\). The values of the hash keys for the above set of data are 10, 5, 1, 10, 0, 3, 6, 5, 7, 2.

a) Show the resulting hash table if collisions are handled by chaining. (2 marks)

b) Show the resulting table if collisions are handled by linear probing. Explain in detail how each element is inserted into the hash table. (3 marks)

c) Show the resulting hash table if collisions are handled by double hashing, with the primary hash function \(h(x) = (3x + 7) \mod 11\), and the secondary hash function \(h'(x) = 5 - (x \mod 5)\). You will need to calculate the values of the second hash function \(h'(x)\) in cases where a collision occurs. (4 marks)

d) How many comparisons are required to insert an element into a hash table of length \(n\) which currently contains \(m\) elements \((m < n)\) using linear probing? Discuss the best and the worst case scenarios. (4 marks)

e) Give a pseudocode description of an algorithm for performing the removal of an element from a hash table created using linear probing. Consider in your answer the most general case that the element is not in the original position determined by the hash key, and that some elements between the original and the actual position of the element in the hash table may have been previously removed. (Hint: One method is to introduce a mechanism to deal with ‘deactivated’ elements) (7 marks)
4. Graphs and graph algorithms.

   a) Consider the following two representations of finite directed graphs

      - Adjacency lists
      - Adjacency matrices

   i) Explain clearly what these representations are. Choose an illustrative example of a directed graph with at least 4 nodes and show how it can be presented using these two representations. (3 marks)

   ii) Give ONE computational task on directed graphs that can be more efficiently undertaken using adjacency lists rather than adjacency matrices. Similarly, give ONE computational task on directed graphs that can be undertaken more efficiently using adjacency matrices rather than adjacency lists. In both cases you should explain clearly what the task is, what algorithm is involved and show why it is more efficient on the given representation by calculating the relevant time complexity measures (and giving the calculation). (4 marks)

   b) Explain what is meant by a Depth-First Search (DFS) and a Breadth-First Search (BFS) of

      - a finite rooted binary tree, and
      - a finite directed graph.

   In each case you should explain clearly the principles involved and give examples to illustrate your answer, but you need not give explicit algorithms. (4 marks)

   c) For the case of finite undirected graphs, give an explicit algorithm for performing DFS of such graphs which numbers the nodes in the order that they are first encountered. You may either present a program or express the algorithm in pseudocode. (5 marks)

   d) A spanning tree in an undirected graph is a subset of the edges which forms a rooted tree and which includes all the nodes of the graph.

   Explain clearly why a DFS of a finite, connected, undirected graph produces a spanning tree. (4 marks)