Two hours

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

Algorithms and Imperative Programming

Date: Thursday 20th May 2010
Time: 14.00 – 16.00

Please answer THREE Questions for the FOUR Questions provided
This is a CLOSED book examination

The use of electronic calculators is permitted provided they are not programmable and do not store text.
1. a) Three algorithms use the following number of basic operations on an input of size $n$.

   A: $n^3 + 4n^2 + 21$
   B: $\frac{1}{2} n^3 + 6$
   C: $20n^2 + 10n$

   i) What is the time complexity of each algorithm in big-O notation? (3 marks)

   ii) Show that doubling the time available for a run of algorithm B only increases the size of input that can be handled by about 26%. (Assume $n$ large). (3 marks)

b) Consider the following growth functions. Each one has a different asymptotic complexity.

   $4^n$, $(4n)^2$, $4n^2$, $n^{(3/2)}$, $10^n$, $n \log(n)$, $n!/2$, $\log(n)$

   i) Write them out in complexity order, slowest growth rate first. (3 marks)

   ii) Which of the growth rates are normally considered intractable? (List all of them.) (3 marks)
c) Consider the following two algorithms, which perform the same task in two different ways. The input of each algorithm is two dictionary-sorted lists of unique words, $A[]$ and $B[]$. The output is a listing of the words that appear in both input lists. (HINT: You do not need to understand the algorithms in order to calculate their complexities.)

```
All_pairs_compare( A, B )
for ( i = 0 to |A|-1 )
    for ( j = 0 to |B|-1 )
        if ( string_match?(A[i], B[j]) == TRUE )     // comment: this operation is O(1)
            print(A[i], "NEWLINE");
    END
END

Merge_then_compare( A, B )
i=0; j=0; k=0;
while ( k < |A| + |B|)
    if (B[j] precedes A[i])     // comment: this operation is O(1)
        C[k]=B[j];
        j++;
    else
        C[k]=A[i];
        i++;
        k++
    for(i=0 to k-1)
        if (C[i] equals C[i+1])     // comment: this operation is O(1)
            print(C[i],"NEWLINE")
    END
END
```

i) What is the worst-case time complexity of each algorithm? Assume that: The input array of strings $A[]$ is of length $|A|$. The input array of strings $B[]$ is of length $|B|$. Assume that the total number of strings on the input is $|A| + |B| = N$, and that $|A|$ is approximately equal to $|B|$ (i.e. both are $1/2 N$). SHOW YOUR WORKING. The final complexity should be written in big-O notation. (4 marks)

ii) What is the best-case time complexity of each algorithm? Hint: assume that no words match in the two input files. SHOW YOUR WORKING. The final complexity should be written in big-O notation. (4 marks)
2. **Hash Tables**

Consider the following sequence of numbers:

\[ 23, 88, 13, 11, 1, 6, 22, 39, 5, 16 \]

We wish to insert these numbers in this order in a hash table of size 11 indexed from 0 through to 10 using the hash function \( h(x) = (2x + 5) \mod 11 \) (where “mod N" means the remainder after division by N). The resulting hash values of the sequence above is:

\[ 7, 5, 9, 5, 7, 6, 5, 6, 4, 4 \]

a) Show the resulting hash table if collisions are handled by chaining, i.e. sequences at the same value. (2 marks)

b) Show the resulting hash table if collisions are handled by linear probing. Explain how the numbers are entered into the table. (3 marks)

c) Show the resulting hash table if collisions are handled by quadratic probing. Again, show how the numbers are entered into the table. Can all the numbers be inserted successfully? Explain your answer. (4 marks)

d) Time complexity: in the best case, inserting an element in a hash table, using linear probing, requires one operation (if the hash values are known): simply insert at the hash value. If collisions occur, more operations are required. How many operations are required to insert a new element in a hash table of length N already containing M elements in the worst case? Explain your answer. (4 marks)

e) Consider now the action of removing an element from a sequence stored as a hash table using linear probing.

Explain why locating the element in the hash table and overwriting it with a space is not, in general, correct. Give an example to show that the location of other elements needs to be changed.

Describe one method for removing elements from a sequence stored as a hash table using linear probing. You should give a clear explanation of your method but need not give a program (or a pseudo-code description). [Hint: one method is to remove some elements and re-insert them using their hash values and linear probing.] (7 marks)
3. a) Assuming that a total order relation on the keys of a dataset is given (e.g. by a comparator), define the heap-order property of a binary tree. (2 marks)

b) Show the steps for replacing the key 5 with 18 in the following heap, where the highlighted node containing 8 is the last element of the heap.

```
       4
      / \
     5   6
    /   / \
   15  9   20
  /   /   / \
16  25  14  12 11 [8]
```

(7 marks)

c) Discuss the time complexity of the operation from Part b). (3 marks)

d) Give an efficient algorithm for finding all the keys in a heap that are smaller than or equal to a given key \( k \). The key \( k \) is not necessarily equal to any element in the heap. The complexity of the proposed algorithm should be \( O(n_k) \), where \( n_k \) is the number of keys smaller than \( k \). (8 marks)
4. a) Explain the following two representations of directed graphs:
   i) Adjacency lists,
   ii) Adjacency matrices.

   In your explanation you should define each representation and you should show how the following graph is represented as an adjacency list and as an adjacency matrix.

   ![](graph.png)

   (3 marks)

b) Which of the two representations from Question 4(a) is most suitable for the following applications? Explain your answers, using time and space complexities where appropriate.

   i) Finding a node with maximum in-degree (the in-degree of node \( n \) is the number of edges whose target is node \( n \)),

   ii) Finding all nodes which are reachable by a path of length 2 from a given node,

   iii) Given a directed graph \( G \), the operation of constructing the opposite graph, which is the graph with the same nodes as \( G \) but reverses the direction of each edge of \( G \),

   iv) Representing a graph with many nodes but few edges. (6 marks)

c) How may the two above representations of graphs be used to represent undirected graphs? (2 marks)

d) Explain clearly what is meant by a depth-first search (DFS) of an undirected graph. You may give a program or a pseudocode description or a clear step-by-step explanation of the traversal method. (5 marks)

e) Explain how a DFS of an undirected graph may be used to compute the number of connected components of an undirected graph. You need not give a program or pseudocode, but should explain your method clearly. (4 marks)

END OF EXAMINATION