UNIVERSITY OF OTAGO EXAMINATIONS 2012

COMPUTER SCIENCE

Paper COSC242

ALGORITHMS & DATA STRUCTURES

Semester 2

(TIME ALLOWED: THREE HOURS)

This examination consists of 5 pages including this cover page.

Candidates should answer **all** questions.

Questions are worth various marks each and submarks are shown thus: (5)

The total number of marks available for this examination is 100.

No supplementary material is provided for this examination.

Candidates may **not** bring calculators, reference books, notes, or other written material into this examination room.

At the end of the examination, hand in the entire exam attached to answer book.

TURN OVER

1. Complexity classes

(a) How many comparisons will Insertion Sort do if the input is an already-sorted array of length n? (1)

(b) What is the worst case time complexity of Insertion Sort on an input array of length n? (1)

(c) What is the worst case time complexity of Mergesort on an input array of length n? (1)

(d) If you were given an input array of small integers ranging in value from 0 to 113, and the length of the array was 100, what would be the most efficient algorithm with which to sort the input? (1)

(e) What is the worst case time complexity of searching an unsorted array of length n? (1)

(f) What is the worst case time complexity of Binary Search on a sorted input array of length n? (1)

(g) What is the worst case time complexity of searching a Red-Black Tree with n real nodes? (1)

(h) How many distinct subsets does a set of size n have? (1)

2. Recurrences, Big-O, Induction

(a) Use the iteration method to solve the recurrence
$f(1) = 5$
$f(n) = f(n-1) + 4.$
You do **not** need to prove that your solution is correct. (3)

(b) Using the definition of big-O and induction, prove that $n^2 = O(n!)$. (7)

3. Sorting

(a) Mention one way to improve Mergesort. (2)

(b) Write a well-planned paragraph describing why you might want to improve the partitioning in Quicksort and how you could do it. (4)

(c) Name one algorithm that can sort keys in $O(n)$ time. What requirements must be satisfied in order for this algorithm to be used and to work efficiently? (4)

4. Hash Tables

(a) Given a table of size 7, a hash function $h(k)$, and input keys 72, 37, 35, 51, 42, 64 and 71 (in that order), draw the hash table that results from:

(i) Chaining, with $h(k) = k\%7$. (4)

(ii) Chaining, with universal hash function $h_{(10,0)}(k) = (10k\%101)\%7$. (6)

(iii) Open addressing with double hashing. Use $h(k) = k\%7$ as the primary hash function, and $h_{(10,0)}(k) = (10k\%101)\%7$ as the secondary hash function. (5)

(iv) Cuckoo hashing, with $h(k) = k\%7$ as the primary hash function, and $h_{(10,0)}(k) = (10k\%101)\%7$ as the secondary hash function. (5)

(b) Suppose you were using a perfect hashing scheme to create a hash table from the keys above. Would the function $h_{(10,0)}$ used above be acceptable as the primary hash function? Show your reasoning. (2)

5. Trees

(a) Draw the final binary search tree T that results from successively inserting the keys 75, 64, 53, 42, 31 into an initially empty tree. (1)

(b) Write down the keys of T in the order in which they would be visited during a postorder traversal. (1)

(c) Show all the red-black trees that result after successively inserting the keys 75, 64, 53, 42, 31 into an initially empty red-black tree. State which cases apply. (6)

(d) Show all the red-black trees that result from the deletion of 64. State which cases apply. (4)

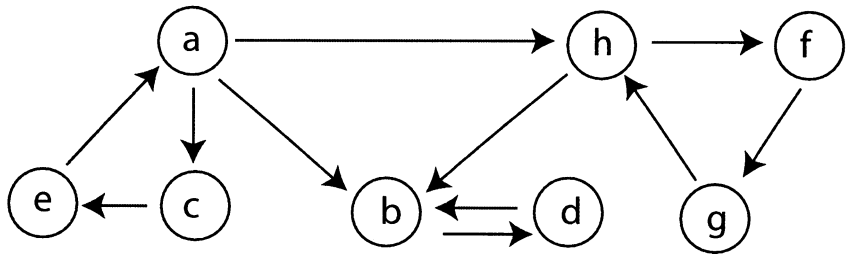
(e) By a 2-3-4 tree we mean a B-tree of minimum degree t = 2. Show the results of successively inserting the keys 6, 2, 4, 8, 7, 9, 10, 11, 12 into an initially empty 2-3-4 tree. You should at least draw the trees just before some node must split and just after the node has split. (5)

(f) Show what happens when you delete first key 8 and then key 9. (3)

6. **Graphs**

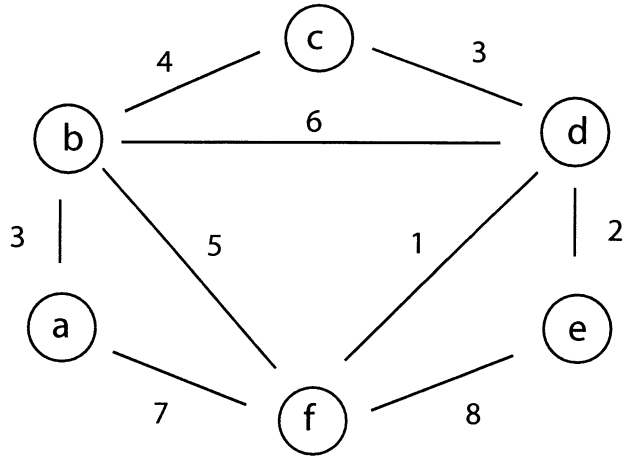
(a) Give one reason why you might prefer to use an adjacency matrix representation of a graph, and one reason why you might decide to use an adjacency list representation. (4)

(b) Copy the following directed graph into your answer book. Use depth-first search to find the strongly connected components of the graph. Show the finishing times computed by the first application of depth-first search, and show the trees produced by the second depth-first search. Start at key a and consider adjacency lists to be alphabetically ordered.



(8)

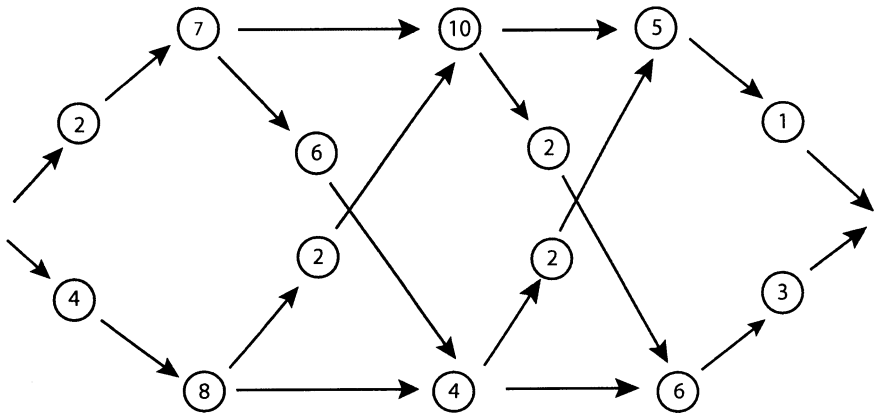
(c) Copy the following weighted undirected graph into your answer book. Show how Dijkstra's algorithm would find the shortest paths from source a. Show clearly how the priority values change, and show the order in which vertices are extracted from the priority queue. Give a table showing vertices and their predecessors/parents, from which the shortest paths can be computed.



(8)

7. **Dynamic programming**

Consider the assembly line scheduling problem below. Give a dynamic programming solution. Show any bottom-up tables used in your solution and any calculations you perform. Explain what the entries in your tables mean.



(5)

8. **P and NP**

In a few well-chosen sentences, explain to Aunt Maud what the classes P and NP are, and what it means to say that a problem is NP-complete. Give her one example of an NP-complete problem. (5)