UNIVERSITY OF OTAGO EXAMINATIONS 2014

COMPUTER SCIENCE

Paper COSC242

ALGORITHMS & DATA STRUCTURES

Semester 2

(TIME ALLOWED: THREE HOURS)

This examination comprises 5 pages.

Candidates should answer questions as follows:

Candidates must answer **all** questions.
Each question is worth various marks and submarks are shown thus: (5)
The total number of marks available for this examination is 100.

The following material is provided:

Nil.

Use of calculators:

No calculators are permitted.

Candidates are permitted copies of:

Nil.

TURN OVER

1. **Complexity classes**

(a) How many comparisons will Insertion Sort perform on an already sorted input 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 Merge Sort on an input array of length n? (1)

(d) What is the worst case time complexity of Quicksort on an input array of length n, using our unmodified Partition algorithm? (1)

(e) If you were given an input array of small integers ranging in value from 0 to 108, and the length of the array was 100, what would be the most efficient algorithm for sorting the input? (1)

(f) What is the worst case time complexity of searching a perfect hash table? (1)

(g) What is the worst case time complexity of searching a linked list of length n? (1)

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

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

(j) How many permutations may be generated from a set of n items? (1)

2. **Recurrences, Big-O, Induction**

(a) Use the iteration method to solve the recurrence equations
$f(1) = 0$
$f(n) = f(n-1) + (n-1)$.
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) Describe one way to improve Merge Sort. (Stick to improvements discussed in class — don't invent your own.) (2)

(b) Describe why you might want to improve the partitioning in Quicksort and how you could do it. (3)

(c) Show how Radix Sort would sort the keys 53, 22, 23, 75, 13, 42, 45, 66. (Show each stage of the sorting process, not just the sorted output.) (3)

(d) Which of the following sorts is stable? Insertion Sort? Quicksort? Merge Sort (with our version of Merge)? Counting Sort? (2)

4. Hash Tables

(a) Given a table of size 7, a hash function h(k), and input keys $27, 47, 81, 74, 11, 50, 64$ (in that order), draw the hash table that results from:

(i) Open addressing with linear probing. Use $h(k) = k\%7$ as the hash function. (5)

(ii) Open addressing with double hashing. Use $h(k) = k\%7$ as the primary hash function, and $g(k) = 1 + (k\%6)$ as the secondary hash function. (5)

(iii) Chaining, with universal hash function $h_{(10,10)}(k) = ((10k + 10)\%101)\%7$. (5)

(b) Suppose you were using a perfect hashing scheme to create a hash table from the keys above. Would $h_{(10,10)}$ be acceptable as the primary hash function? Give your reasoning. (5)

5. Trees

(a) Draw the final binary search tree T that results from successively inserting the keys $1, 5, 2, 4, 3$ 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) Draw the results of deleting 1, then 5, then 2 from T. (3)

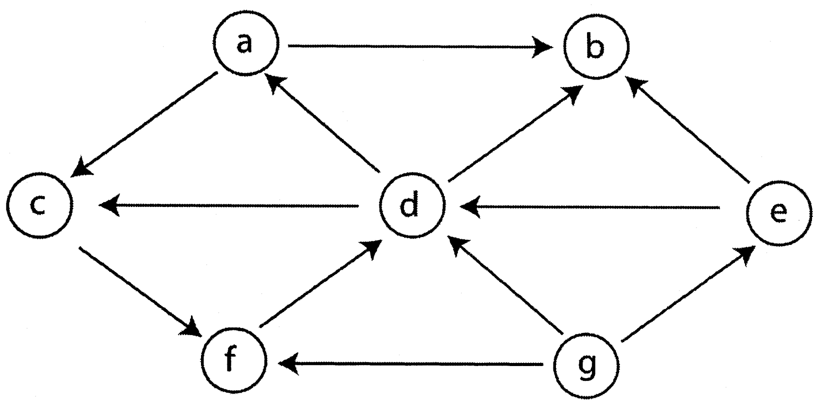
(d) Show all the red-black trees that result after successively inserting the keys $1, 5, 2, 4, 3$ into an initially empty red-black tree. State which cases apply. (5)

(e) Show all the red-black trees that result from the successive deletion of 1, then 5, then 2. State which cases apply. (5)

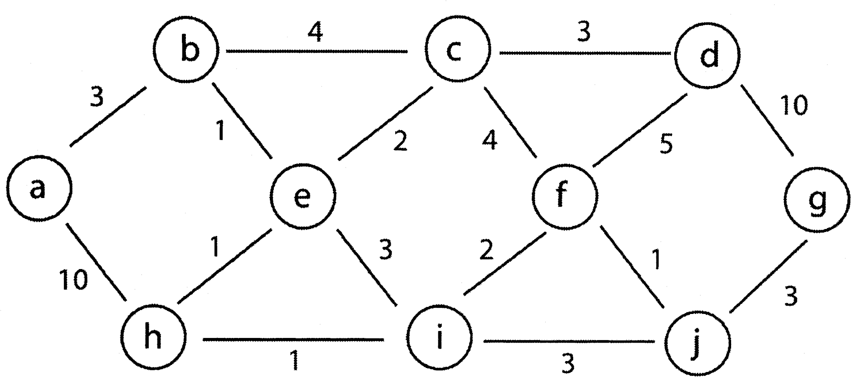
(f) By a 2-3-4 tree we mean a B-tree of minimum degree t = 2. Show the results of successively inserting the keys $10, 9, 8, 7, 6, 5, 4, 3, 2, 1$ into an initially empty 2-3-4 tree. You should at least draw the trees just before some node must split and just after the node has split. (5)

6. **Graphs**

(a) Copy the following directed graph into your answer book. Starting at a, and considering adjacency lists to be alphabetically ordered, show how depth-first search would allocate time stamps to vertices, and label the edges with T, F, B, or C according to whether each is a tree, edge, forward edge, back edge, or cross edge. (10)

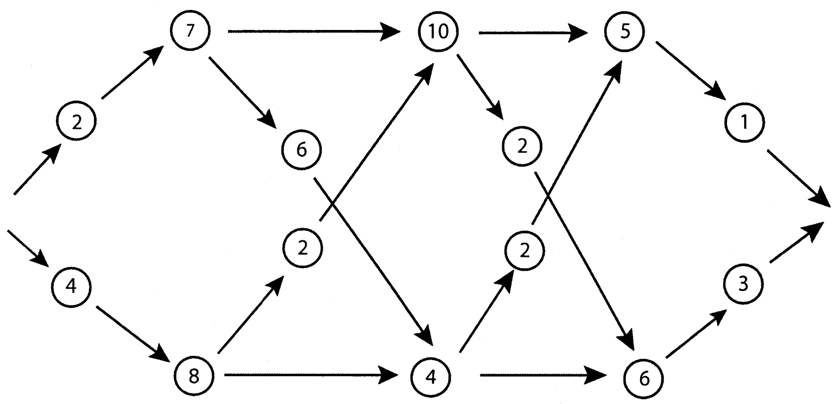


(b) Copy the following weighted undirected graph into your answer book. Show how Dijkstra's algorithm would find the shortest paths from source a. Show clearly how the priority values change, and show the order in which vertices are extracted from the priority queue. Give a table showing vertices and their parents from which the shortest path to any vertex can be computed. (10)



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 what the classes P and NP are, and what it means to say that a problem is NP-complete. Give one example of an NP-complete problem. (5)

END