# UNIVERSITY OF OTAGO EXAMINATIONS 2016

## 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, with two bonus marks available in addition.

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) What is the worst case time complexity of Sequential Search on an input array of length $n$? (1)

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

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

(d) Okay, now Aunt Maud asks a tough one: Given your answers to the above, which is better: To keep the array unsorted and simply use Sequential Search, or to sort the array using something like Merge Sort and then use Binary Search? Why? (2)

(e) Suppose you need to sort an already nearly sorted input array of length $n$. Which of the following algorithms would you use? Merge Sort or Insertion Sort? (1)

(f) Suppose you need to sort an input array of length $n$ and memory is very tight. Which of the following algorithms would you use? Quicksort or Merge Sort? (1)

(g) Suppose you need to sort an input array of length $n = 14000$ and the keys in the array are integers ranging in value from 0 to 5000. Is there a sorting algorithm that is faster than Quicksort? If you think there is, then what is it called? Can you use it if memory is tight? (3)

2. **Recurrences, Big-O, Proof Techniques**

(a) Use the iteration method to solve the recurrence equations
$f(1) = 0$
$f(n) = f(n-1) + (n-1)$.
(You need not prove that your solution is correct.) (4)

(b) Use proof by contradiction to show that $n^3 \neq O(n^2)$. (6)

3. **Sorting**

(a) How could you improve the running time of Merge Sort? (2)

(b) How could you improve the average case running time of Quicksort? (2)

(c) Show how Radix Sort would sort the keys 23, 42, 13, 35, 38, 28, 45, 12. (Show each stage of the sorting process, not just the sorted output.) (3)

(d) What does it mean to say that a sorting algorithm is stable? Name one example of a stable sorting algorithm. (Stick to the algorithms we've analysed in COSC242. Minus 99 marks if you tell me "Wikipedia says Merge Sort is stable".) (3)

TURN OVER

4. **Hash Tables**

(a) Given a table of size 7 and input keys $72, 37, 35, 51, 42, 64, 70$ (in that order), and the hash function $h_{(10,10)}(k) = ((10k + 10)\%101)\%7$, draw the hash table that results from:

(i) Chaining. (5)

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

(iii) Cuckoo hashing, using the above $h$ as primary hash function and the above $g$ as secondary hash function. (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) Show all the red-black trees that result after successively inserting the keys
$72, 37, 35, 51, 42, 64, 70$
into an initially empty red-black tree. State which cases apply. (6)

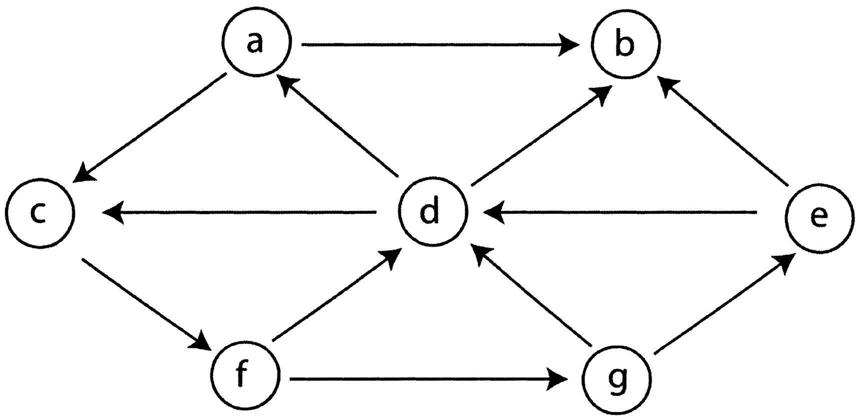
(b) Show the results of successively deleting 64 and 42, giving details of your reasoning. (State which cases apply.) (7)

(c) By a 2-3-4 tree we mean a B-tree of minimum degree t = 2. Show the results of successively inserting the keys $72, 37, 35, 51, 42, 64, 70$ 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. (3)

(d) Show the results of deleting first 42 and then 37 from your 2-3-4 tree. (4)

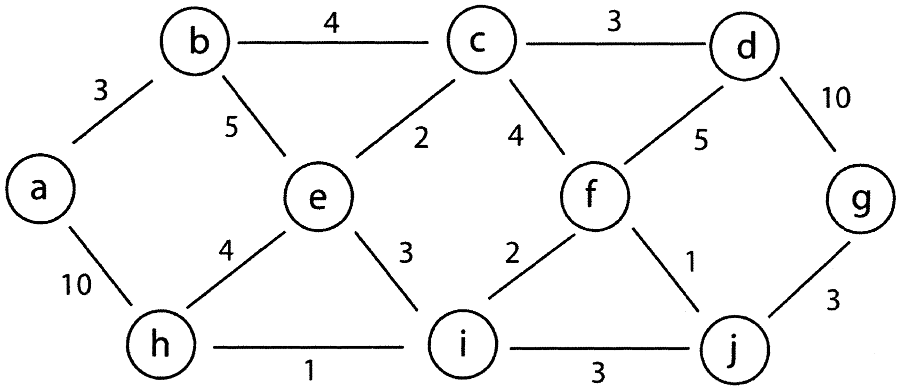
## 6. **Graphs**

(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) For a bonus mark, mention one reason why you might want to label the edges. (1)

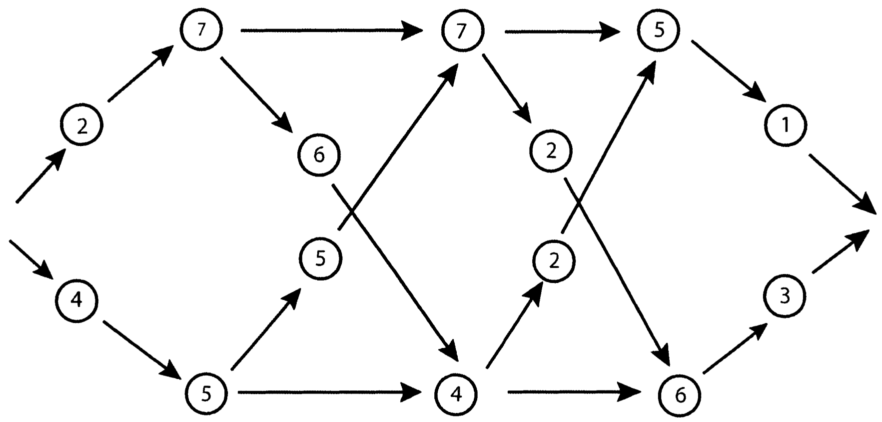
(c) Copy the following weighted undirected graph into your answer book. Show how Dijkstra's algorithm would find the shortest paths from source *a*. Show clearly how the priority values change, and show the order in which vertices are extracted from the priority queue. Give a table showing vertices and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




(d) For a bonus mark, mention one example of a company that might ask you to use Dijkstra's algorithm for their business. (1)

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)