Enrolment number (student number):

The University of Melbourne
Practice Exam Paper

School of Computing and Information Systems

COMP90038 Algorithms and Complexity

Reading Time: 15 minutes
Exam Duration: 3 hours

This paper has 11 pages, including this front page.

Authorised Materials:
None. This is a closed book exam. Electronic devices, including calculators and laptop computers are **not** permitted.

Instructions to Invigilators:
Students will provide answers in the exam paper itself. The exam paper must remain in the exam venue and must be returned to the examiner.

Instructions to Students:
This is not an actual exam paper. It is a practice paper which has been put together to show you the format that you can expect in the exam. Many aspects of this paper's contents do not necessarily reflect the contents of the actual exam paper: The selection of topics, the number of questions or sub-questions, the perceived difficulty of individual questions, and the distribution of weights are all aspects that may be different. When preparing for the exam, you should cover the entire syllabus and not focus only on topics or question types used in this practice paper. If anything, the exam paper can be expected to be harder than this practice paper.

There are 12 questions. As in the exam, you should attempt them all. Of course your answers must be *readable*. Any unreadable parts will be considered wrong. You will find some questions easier than others; in the actual exam you should allocate your time accordingly. Marks are indicated for each question, adding to a total of 70.

The actual exam paper will be printed single-sided, so you will have plenty of space for rough work on the flip sides. Only what you write inside the allocated boxes will be marked. Page 11 is overflow space, in case you need more writing space for some question.

	1	2	3	4	5	6	7	8	9	10	11	12
Examiners' use:												

Question 1

(4 marks)

A. Give the names of two *stable* sorting algorithms, together with their worst-case time complexities. Write the names and complexities in the box:

B. Give the names of two *unstable* sorting algorithms, together with their worst-case time complexities. Write the names and complexities in the box:

Question 2

(4 marks)

We are given an array A holding n integers, for some large n. **The array is sorted**, and the values in A range from -2147483648 to 2147483647, evenly distributed. Give Θ expressions for the following tasks:

A. Running the insertion sort algorithm on the array A:

B. Running the selection sort algorithm on the array A:

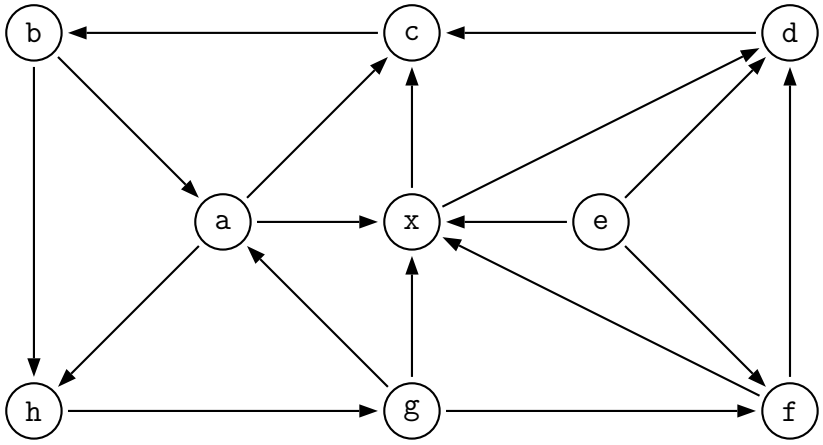
C. Performing binary search for integer k which is not in A:

D. Performing interpolation search for integer k not in A:

Question 3

(4 marks)

For the directed graph below, list the order in which the nine nodes are visited during a depth-first (DFS) traversal, as well as the order in which they are visited during a breadth-first (BFS) traversal. As always, assume that any **ties are resolved** by taking nodes in alphabetical order. Write the answers in the boxes given.



DFS sequence:

BFS sequence:

Question 4

(4 marks)

Given the pattern `A T G A` and the text

`T C A T C A T C C A T G C A C A A T G A C T T T`

how many character comparisons will Horspool's algorithm make before locating the pattern in the text? Write the number in the box:

Question 5

(4 marks)

Assume the array A holds the keys 77, 64, 15, 43, 28, 91, 80, 32, 56 in index positions 1 to 9. Show the heap that results after application of the linear-time bottom-up heap construction algorithm. You may show the heap as a tree or as an array.

Question 6

(4 marks)

The functions A–D are defined recursively as follows (all divisions round down, to the closest integer):

$$\begin{array}{lcll} A(n) & = & 2\ A(n/3) + 2, & \text{with } A(1) = 1 \\ B(n) & = & B(n/2) + n/2, & \text{with } B(1) = 1 \\ C(n) & = & 512\ C(n/8) + 4n^2, & \text{with } C(1) = 4 \\ D(n) & = & 4\ D(n/2) + n^2, & \text{with } D(1) = 2 \end{array}$$

In the following table, for each of the four functions, tick the most precise correct statement about the function's rate of growth:

	$O(n)$	$\Theta(n)$	$O(n \log n)$	$\Theta(n^2)$	$O(n^2 \log n)$	$\Theta(n^2\sqrt{n})$	$O(n^3)$
A							
B							
C							
D							

Question 7 **(4 marks)**

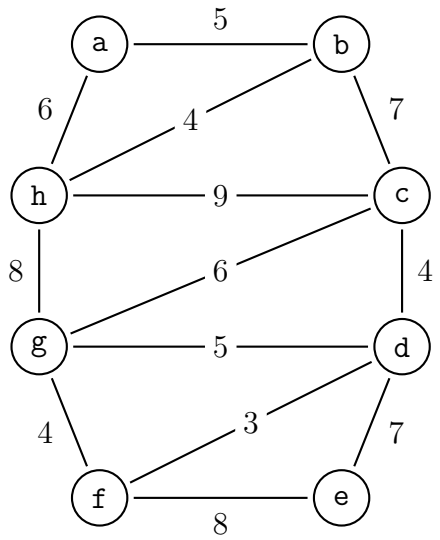
For each of **A**–**D** below, answer yes or no, and, in each case, briefly explain your reasoning (just a justification of your answer, rather than detailed calculations). A yes/no answer that is not justified will not attract marks, even if correct.

Question	**Answer/explanation**
A. Is $\sqrt{n} \in \Omega(n)$?	
B. Is $n^{2 \log n} \in O(n^{\log n})$?	
C. Is $\Theta(\log(n^{2 \log n})) = \Theta(\log(n^{\log n}))$?	
D. Is $\Theta(\log(2^n)) = \Theta(\log(3^n))$?	

Question 8

(6 marks)

A. The box below contains a weighted undirected graph with eight nodes. Give a minimum spanning tree for the graph. You may do that either by outlining a minimum spanning tree on the graph itself, or by drawing the tree in the empty space next to the graph.



B. Given a weighted graph $G = \langle V, E \rangle$, a subgraph $\langle V, E' \rangle$ (that is, $E' \subseteq E$) which is a tree with minimal weight is a *maximum spanning tree* for G.

We want a transformation of the graph G so that we can run Prim's algorithm on the transformed graph G', and the algorithm will find a maximum spanning tree for G.

Describe such a (systematic) transformation from G to G'.

Question 9 **(6 marks)**

Consider the function F below. The function takes as input an integer array A, and the size n of A. The array indices run from 1 to n. The division used is integer division, that it, it rounds down to the closest smaller (or equal) integer value.

In the box, give a Θ expression for the function's time complexity.

```
function F(A[·], n)
    s ← 0
    m ← n
    while m > 0 do
        for i ← 1 to m do
            s ← s + A[i]
        m ← m/2
```

Question 10 **(10 marks)**

Using pseudo-code, give an algorithm for deleting the smallest element of a binary search tree (a BST). Assume a non-empty binary tree T has attributes $T.left$, $T.right$, and $T.root$ which denote T's left sub-tree, right sub-tree, and the key of T's root node, respectively. You can use these tests if they seem useful: IsLeaf(T) tests whether the binary tree T is a leaf, and IsEmpty(T) tests whether it is empty.

Question 11 **(10 marks)**

Consider an array A of n distinct integers (that is, all elements are different). It is known that A was originally sorted in ascending order, but A was then right-rotated r places, where $0 < r < n$. In other words, the last r elements were moved from the end of the array to the beginning, with all other elements being pushed r positions to the right. For example, for $n = 7$ and $r = 3$, the result may look like this:

$$[43, 46, 58, 12, 20, 29, 34]$$

For $r = 5$, the result, based on the same original array, would be

$$[29, 34, 43, 46, 58, 12, 20]$$

You know that the given $A[0..n-1]$ has this particular form, that is, for some r, the sequence $A[r], \ldots, A[n-1], A[0], \ldots A[r-1]$ is in ascending order, but you do not know what r is. Design an algorithm to find the largest integer in A. Full marks are given for an algorithm that works in time $O(\log n)$; half marks are given for a solution that is correct, but less efficient.

Question 12 **(10 marks)**

Two programmers face the following problem. Given an array containing n random integers in random order, find the largest integer. The integers are placed in cells $A[1] \ldots A[n]$.

Programmer X has come up with the code shown below, on the left. (In the programming language used, arrays are indexed from 0, but X's method does not use $A[0]$.)

```
function X(A[·], n)
    max ← A[1]
    i ← 2
    while i ≤ n do
        if A[i] > max then
            max ← A[i]
        i ← i + 1
    return max
```

```
function Y(A[·], n)
    i ← n
    while i > 0 do
        A[0] ← A[i]
        i ← i − 1
        while A[0] > A[i] do
            i ← i − 1
    return A[0]
```

Programmer Y has solved the same problem differently, as shown above on the right.

Compare the two solutions using three criteria: Correctness, time complexity class, and the number of comparisons performed. Write your analysis in the box:

Overflow space

Use this page if you ran out of writing space in some question. Make sure to leave a pointer to this page from the relevant question.