The University of Melbourne
Practice Exam Paper

School of Computing and Information Systems

COMP90038 Algorithms and Complexity

Leo's solution

Question 1

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

1. Insertion sort, Θ(n^2)
2. Merge sort, Θ(n*log n)

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

1. Selection sort, Θ(n^2)
2. Quick sort, Θ(n^2)

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: Θ(n)

B. Running the selection sort algorithm on the array A: Θ(n^2)

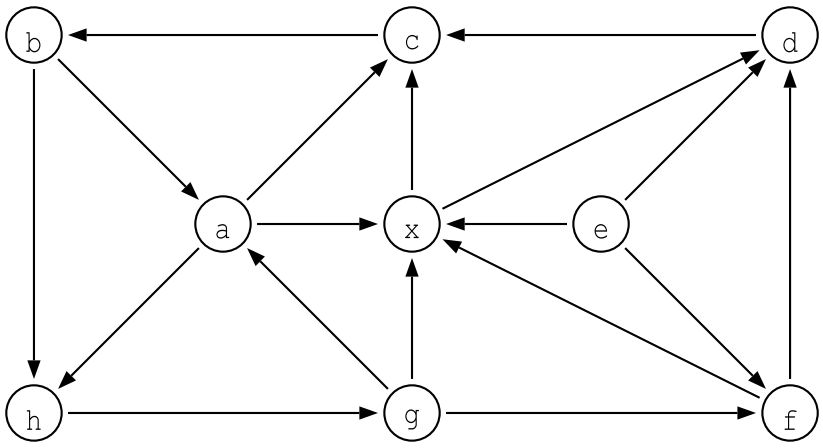
C. Performing binary search for integer k which is not in A: Θ(log n)

D. Performing interpolation search for integer k not in A: Θ(1)

Question 3

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 breadthfirst (BFS) traversal. As always, assume that any **ties are resolved** by taking nodes in alphabetical order. Write the answers in the boxes given.

(4 marks)



DFS sequence: a c b h g f d x e

BFS sequence: a c h x b g d f e

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:

1+2+1+1+2+4(successful searching) = 11

Question 5

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.

[91 64 80 56 28 15 77 32 43]

(4 marks)

Question 6

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

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

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

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 $\sqrt{n} \in \Omega(n)$?	No, because n/(root n) = root n

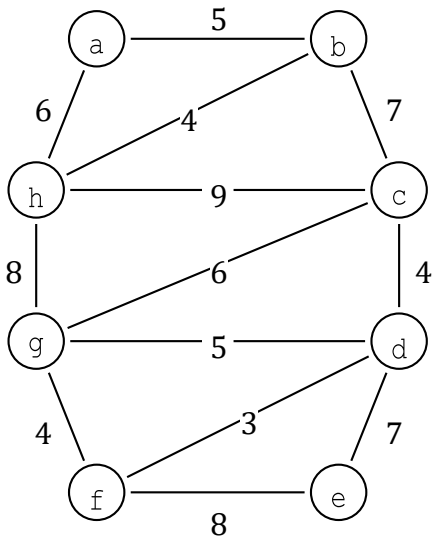
(4 marks)

B. Is $n^{(2\log n)} \in O(n^{(\log n)})$?	No, because n^(2logn)/n^(logn) = n^(logn)
C. Is $\Theta(\log(n^{2\log n})) = \Theta(\log(n^{\log n}))$?	Yes, because $\log(n^{2\log n})/ \log(n^{\log n})$ = 2logn/logn = 2
D. Is $\Theta(\log(2^n)) = \Theta(\log(3^n))$?	Yes, because $\log(2^n)/ \log(3^n)$ = $(\log(3^n) * \log(2)(3)/ \log(3^n)$ = log(2)(3)

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.



These edges were chosen: a-b, b-h, b-c, c-d, d-f, f-q, d-e

B. Given a weighted graph $G = <V,E>$, a subgraph $<V,E'>$ (that is, $E' \subseteq E$) which is a tree with *minimum* 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'.

(I think the red word should be "maximum", otherwise this question is unreasonable.)

For each edge in G, we change the weight from (x) to (-x). For example, the weight of "a to b" is 5 originally, now we let it become -5.

Then we can run Prim's algorithm to find the maximum tree.

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($A[\cdot]$,n)

$s \leftarrow 0$
$m \leftarrow n$
while $m > 0$ **do**
    for $i \leftarrow 1$ **to** m **do**
        $s \leftarrow s + A[i]$
    $m \leftarrow m/2$

n+(n/2)+(n/4)+…+1 = 2n-1 = Θ(n)

Question 10

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.

```
Function deleteSmallest(T)
    if isEmpty(T.left)
        T = T.right          //Delete T
    else
        deleteSmallest(T.left)
```

(10 marks)

Question 11

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 $A[r],...,A[n-1],A[0],...A[r-1]$ 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.

```
Function findMax(A[])
    return find(A, 0, A.length-1)

Function find(A[], start, end)
    mid = (start+end)/2
    if A[mid]>A[mid+1]
        return A[mid]
    if A[mid]>A[0]
        return find(A, mid+1, A.length-1)
    return find(A, 0, mid-1)
```

(10 marks)

Question 12

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]...*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:

	Left one	Right one
Correctness	Correct	Correct
Time complexity	O(n)	O(n)
Number of comparisons	(n-1) + n	n + x (where x may be 2 to n+1)

*The green part is for "if A[i]>max" (left) or "while A[0]>A[1]" (right).
*The red part is the number of comparisons for "while i<=n" (left) or "while i>0" (right).