Office Use Only

MONASH University

SAMPLE EXAM

Semester X, 3018

Examination Period (June 3018)

Faculty of Information Technology

EXAM CODES: **FIT2004**

TITLE OF PAPER: **Algorithms and Data Structures**

EXAM DURATION: ∞ hours writing time

READING TIME: ∞ minutes

THIS PAPER IS FOR STUDENTS STUDYING AT: (tick where applicable)

☐ Berwick	✓Clayton	✓ Malaysia	☐ Off Campus Learning	☐ Open Learning
☐ Caulfield	☐ Gippsland	☐ Peninsula	☐ Monash Extension	☐ Sth Africa
☐ Parkville	☐ Other (specify)			

During an exam, you must not have in your possession any item/material that has not been authorised for your exam. This includes books, notes, paper, electronic device/s, mobile phone, smart watch/device, calculator, pencil case, or writing on any part of your body. Any authorised items are listed below. Items/materials on your desk, chair, in your clothing or otherwise on your person will be deemed to be in your possession.

No examination materials are to be removed from the room. This includes retaining, copying, memorising or noting down content of exam material for personal use or to share with any other person by any means following your exam.
Failure to comply with the above instructions, or attempting to cheat or cheating in an exam is a discipline offence under Part 7 of the Monash University (Council) Regulations.

AUTHORISED MATERIALS

OPEN BOOK	☐ **YES**	✓ **NO**
CALCULATORS	☐ **YES**	✓ **NO**
SPECIFICALLY PERMITTED ITEMS	☐ **YES**	✓ **NO**
if yes, items permitted are:		

Section	Mark
1	
2	
3	
4	
5	
6	
Total	

Candidates must complete this section if required to write answers within this paper

STUDENT ID: __ __ __ __ __ __ __ __ DESK NUMBER: __ __ __ __ __

INSTRUCTIONS

- You must answer ALL the questions.
- Answers to each question should be in the space DIRECTLY BELOW the questions and (if required) on the blank page overleaf of each question.
- Script book may be used if ADDITIONAL SPACE is required for answering these questions

General exam technique

Do not throw marks away by not attempting all questions. Suppose you get 7/10 on a question for a 20 minutes effort. Spending another half hour on the same question gets *at most* 3 more marks. On the other hand, were you to spend that time on a new question, you might get another 10 marks.

Answer the question that is asked of you. If the question asks for Insertion sort, do not write Quick-sort – this only wastes your time.

Do not write un-necessarily long answers. This wastes your valuable exam time. The question will specifically ask for the information required. Therefore, do not include the information that is not specifically asked for. If asked to justify your answer, provide a clear, logical and concise reasoning.

You do not have to attempt the questions in order. Some questions require less work but may be worth more marks. Carefully read the paper to decide the order in which you should attempt the questions based on the marks associated with each question and whether you know the answer or not.

Several questions ask of you to write Pseudocode. Pseudocode is essentially a **high-level description** of a program that should allow a human to understand when it is read. If you feel more comfortable using Python syntax, you are welcome to use it but don't get bogged down with syntax. What is essentially being assessed for such 'pseudocode' questions is your basic understanding and the logic of the algorithm (and not its syntactical correctness).

Write neatly. Use a new pen not an old, leaking/splodgy pen.

Best of Luck!

Section 1 (Short Questions) **(Section weight = 18 marks)**

This section is composed of 8 short questions numbered (i) to (viii). Write your answers to each of these questions in **no more than a few lines**.

(i) What is an in-place algorithm? Is the recursive Quick Sort algorithm an in-place algorithm? Provide brief reasoning for your answer.

(ii) What is the average-case time complexity of Quick Select? Briefly justify your answer.

(2+4 = 6 marks)

6

(Page intentionally left blank for working space)

(iii) Insert 27 in the AVL Tree shown in Fig. 1. Show the rotations involved.

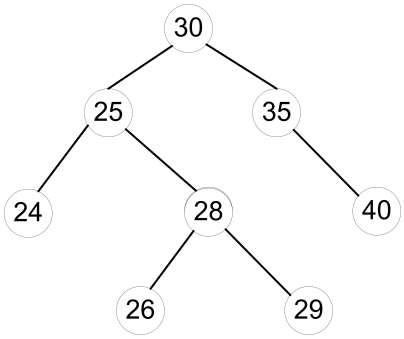

Figure 1: AVL Tree

(2 marks)

2

(Page intentionally left blank for working space)

(iv) What is the worst-case time complexity of insertion in a hash table? Briefly describe reasoning.

(v) Write the loop invariant for Selection Sort.

(vi) What is the best-case time complexity of Insertion sort? Briefly justify your answer.

(2+2+2 = 6 marks)

6

(Page intentionally left blank for working space)

(vii) Given an **unweighted directed graph**, all-pairs shortest distances can be computed using one of the following approaches: Floyd-Warshall algorithm, Bellman-Ford algorithm, calling Dijsktra's algorithm for each vertex or calling Breadth-First search for each vertex. Which of these is the most efficient for computing all-pairs shortest distances in an unweighted directed graph? Provide brief reasoning.

(viii) Is the topological sort of a directed acyclic graph unique? Justify your answer using an example.

(2+2 = 4 marks)

4

(Page intentionally left blank for working space)

Section 2 (Recurrence Relations) **(Section weight = 12 marks)**

This section is composed of 2 questions: (A) and (B)

(A) Using **induction**, prove that the following **recurrence relationship**:

$$T(N) = \begin{cases} T(N/3) + a, & \text{if } N = 3^k \text{ where } k > 0. \\ b & \text{if } N = 1 \end{cases}$$

has the solution

$$T(N) = b + a \log_3 N$$

where a and b are constants.

(5 marks)

5

(Page intentionally left blank for working space)

(B) Consider the following function that returns Nth power of a positive number x where $N \geq 0$.

```
def power(x,N):
    if N == 0:
        return 1
    if N == 1:
        return x
    else:
        return x * power(x,N-1)
```

(i) Write the recurrence relation for the `power(x,N)` function and solve it. What is its time complexity in Big-O notation?

(ii) What is the space complexity of `power(x,N)` function? Give a brief justification of your answer.

(5+2 = 7 marks)

7

(Page intentionally left blank for working space)

Section 3 (**Proving correctness**) **(Section weight = 6 marks)**

Prove the correctness of the following psuedocode of Binary Search in a sorted array containing N integers.

```
lo = 1
hi = N + 1
while ( lo < hi - 1 )
    mid = floor( (lo+hi)/2 )
    if key >= array[mid]
        lo=mid
    else
        hi=mid
if N > 0 and array[lo] == key
    print(key found at index lo)
else
    print(key not found)
```

(6 marks)

6

(Page intentionally left blank for working space)

Section 4 (Retrieval Trees) **(Section weight = 12 marks)**

This section is composed of 2 questions: (A) and (B)

(A) (i) Draw a trie containing the following strings: baby, bad, bank, box, banks, dog, dogs.

(ii) Write pseudocode for searching a string from a trie.

(3+4 = 7 marks)

7

(Page intentionally left blank for working space)

(B) (i) Draw a suffix tree for the string "referrer".

(ii) Briefly describe how you would count the number of occurrences of a substring in a string using suffix tree.

(3+2 = 5 marks)

5

(Page intentionally left blank for working space)

Section 5 (**Dynamic Programming**) **(Section weight = 6 marks)**

Given a capacity c and a set of items with their weights and values, the unbounded knapsack problem is to pick items such that their total weight is at most c and their total value is maximized. You can pick an item as many times as you want. Consider four items $A : (9kg, \$550)$, $B : (5kg, \$350)$, $C : (6kg, \$180)$, and $D : (1kg, \$40)$. If the capacity c is $12kg$, the solution to unbounded knapsack is to pick two items of B and two items of D with total weight $12kg$ and total value \$780.

Write pseudocode for a dynamic programming algorithm to solve the unbounded knapsack problem. What is the space and time complexity of your algorithm?

(5+1 = 6 marks)

6

(Page intentionally left blank for working space)

Section 6 (Graphs) **(Section weight = 20 marks)**

This section is composed of 3 questions: (A) to (C)

(A) Let $G = (V, E, W)$ be a **weighted undirected graph**, with the vertex set V, the edge set E, and the corresponding weights set W.

(i) Write pseudocode for the Bellman-Ford algorithm that finds **shortest distances** from a source vertex s to all other vertices in the graph.

(ii) What is the time-complexity of this algorithm for a dense graph? Briefly justify your answer.

(4+2 = 6 marks)

6

(Page intentionally left blank for working space)

(B) Write pseudocode for Prim's algorithm.

(C) Provide a proof of correctness for Prim's algorithm.

(4+4 = 8 marks)

8

(Page intentionally left blank for working space)

(D) Consider the flow network shown in Figure 2 where, for each edge label i/j, i indicates flow value across the edge and j indicates the capacity of the edge. Flow is not shown if 0. Answer the following questions.

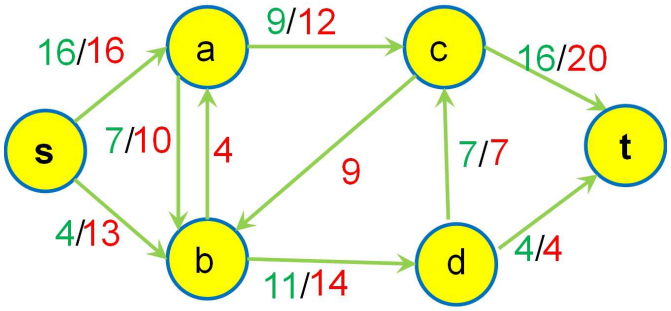

Figure 2: Flow Network

(i) What is the current value of flow in this network?

(ii) Draw a residual network corresponding to this flow network and identify an augmenting path.

(iii) Show the flow network after the flow of the augmented path is augmented in the flow network. What is the value of the flow after the flow has been

(1+3+2 = 6 marks)

6

(Page intentionally left blank for working space)

(Page intentionally left blank for working space)