Office Use Only

MONASH University

Semester One 2018
Examination Period

Faculty of Information Technology

EXAM CODES: FIT2004

TITLE OF PAPER: Algorithms and Data Structures

EXAM DURATION: 2 hours writing time

READING TIME: 10 minutes

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

☐ Caulfield ✓ Clayton ☐ Parkville ☐ Peninsula
☐ Monash Extension ☐ Off Campus Learning ✓ Malaysia ☐ Sth Africa
☐ 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 **if yes, items permitted are:**	**☐ YES**	**☒ NO**

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

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

This page is intentionally left blank. If needed, you can use this space to write answers.

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.

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.

Do not write un-necessarily long answers. This wastes your valuable exam time. The question will specifically ask for the information required. 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.

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).

Best of Luck!

Do not write anything in this table. It is for office use only.

Question	Points	Score
1	10	
2	5	
3	11	
4	4	
5	8	
6	6	
7	10	
8	6	
Total:	60	

This page is intentionally left blank. If needed, you can use this space to write answers.

1. This question is composed of four short questions. Write your answers to each of these questions in no more than a few lines.

   (a) (2 marks) What is the best-case time complexity of Quick Sort algorithm? Briefly explain the scenario in which it performs the best.

   Best-case: $O(N \log N)$; It happens when the pivot is always in the middle.

   (b) (3 marks) Assume that we are calling Breadth First Search (BFS) algorithm **for every vertex** v of the graph to compute all pairs-shortest distances on an unweighted graph. What will be the worst-case time complexity of this approach? What will be the worst-case time complexity if the graph is dense? Briefly justify your answer.

   Complexity using BFS: $O(V(V+E))$: Note that $O(VE)$ is also correct assuming connected graphs. For dense graphs, $O(E) = O(V^2)$. Therefore, complexity is $O(V^3)$.

This page is intentionally left blank. If needed, you can use this space to write answers.

(c) (3 marks) Consider the flow network shown in Figure 1 where, for each edge label i/j, i indicates the flow value across the edge and j indicates the capacity of the edge. Flow is not shown along the edge if it is 0. Consider a cut (S, T) where $S = \{s, a, b, c\}$ and $T = \{d, t\}$. What is the capacity of the cut and what is the flow of the cut?

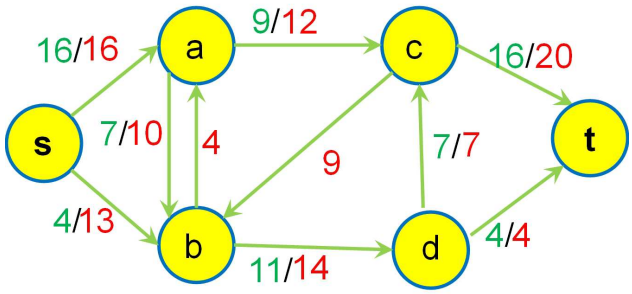


Figure 1: Flow Network

Capacity: 20 + 14 = 34
Flow: 16+11-7=20

(d) (2 marks) What is the time complexity of Radix Sort algorithm for sorting N strings where each string consists of M letters from English alphabet? Give a brief reasoning.

Complexity: $O(MN)$. Complexity for each counting sort is $O(N)$. Radix sort would need M calls to counting sort.

This page is intentionally left blank. If needed, you can use this space to write answers.

2. (5 marks) A country uses n coins with denominations $\{a_1, a_2, \cdots, a_n\}$ arranged in the increasing order of their denomination values. Given a value V, the coin change problem is to find **the minimum number of coins** that add up to V. For example, if the coins are $\{2, 5, 7, 10\}$ and the value V is 16 then the minimum number of coins required to make 16 is 3 (two coins of value 7 and one coin of value 2). There may exist a value V which cannot be constructed using any combination of coins, for example, $V = 3$. In such case, ∞ is to be returned.

   Write pseudocode for a dynamic programming algorithm to solve the above coin change problem.

Bottom-up solution

```
Initialize Memo[ ] to contain infinity for all indices
Memo[0] = 0
for v = 1 to V
    minCoins = Infinity
    for i=1 to N
        if Coins[ i ] <= v
            c = 1 + Memo[v - Coins[ i ] ]
            if c < minCoins
                minCoins = c
    Memo[v] = minCoins
return Memo[V]
```

Top-down solution

```
Initialize Memo[ ] to contain -1 for all indices
Memo[0] = 0
Function CoinChange(value)
    if Memo[value] != -1:
        return Memo[value]
    else:
        minCoins = Infinity
        for i=1 to N
            if Coins[ i ] <= value
                c = 1 + CoinChange(value - Coins[ i ])
                if c < minCoins
                    minCoins = c
        Memo[value] = minCoins
```


5

This page is intentionally left blank. If needed, you can use this space to write answers.

3. For this question, you **may** need to use $\sum_{i=0}^{N} ar^i = \frac{a\left(r^{N+1}-1\right)}{r-1}$.

(a) (5 marks) Write the recurrence relation of time complexity for the `mystery(N)` function shown below and solve it. What is its time complexity in Big-O notation?

```
def mystery(N):
    if N == 1:
        return 1
    else:
        value = 0
        for i in range(N):
            value += i
        return value + mystery(N//2)
```

$T(1) = a$
$T(N) = b * N + T(N/2)$
$T(N) = b * N + b * N/2 + T(N/4)$
$T(N) = b(N + N/2 + N/4) + T(N/8)$
$T(N) = b(N + N/2 + N/2^2 + \cdots + N/2^{k-1}) + T(N/2^k)$
Setting $k = log(N)$
$T(N) = bN(1 + 1/2 + 1/4 + \cdots + 1/2^{log(N)-1}) + a$
$1 + 1/2 + 1/4 + \cdots + 1/2^{log(N)-1} = \frac{(1/2)^{log(N)}-1}{0.5-1} = 2(1 - 1/N)$
$T(N) = 2bN(1 - 1/N) + a$
$T(N) = 2bN - 2b + a$
Time complexity in Big-O notation is $O(N)$.

5

This page is intentionally left blank. If needed, you can use this space to write answers.

(b) (2 marks) What is the space complexity of `mystery(N)` function given in part(a)? Give a brief justification of your answer.

Space complexity: $O(\log N)$.
This is because $O(\log N)$ is required to store the recursive calls in the stack.

(c) (4 marks) Using **induction**, prove that the following **recurrence relationship**:

$$T(N) = \begin{cases} T(N-1) + aN, & \text{if } N > 0. \\ b & \text{if } N = 0 \end{cases}$$

has the solution

$$T(N) = b + \frac{aN(N+1)}{2}$$

where a and b are constants. You **must** use **induction** in your proof.

Base case: N = 0
$T(0) = b + \frac{a*0(N+1)}{2} = b$
Inductive step
Assume it holds for a value k, i.e., $T(k) = b + ak(k+1)/2$. We show that it holds for $k+1$, i.e.,$T(k+1) = b + a(k+1)(k+2)/2$
$T(k+1) = T(k) + a(k+1) = b + ak(k+1)/2 + a(k+1)$
$T(k+1) = b + \frac{ak(k+1)+2a(k+1)}{2} = b + \frac{a(k+1)(k+2)}{2}$

This page is intentionally left blank. If needed, you can use this space to write answers.

4. (4 marks) Show how the following AVL tree is balanced after 17 is added. You need to identify each case (e.g., left-left case) and show how **each** rotation is done. You must also include the balance factors for the nodes in your figures.

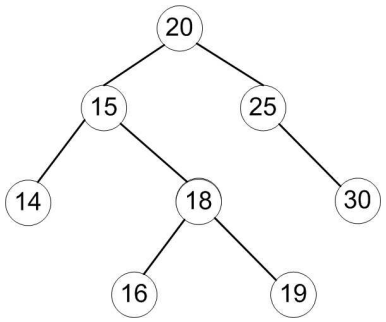
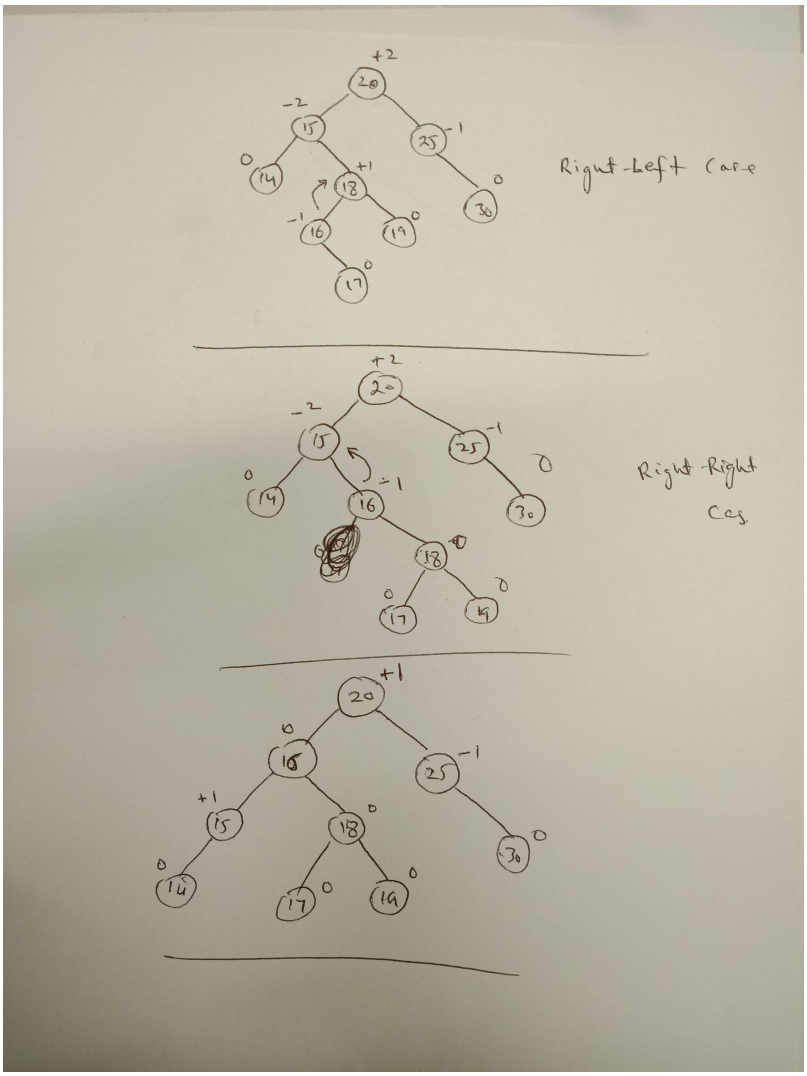


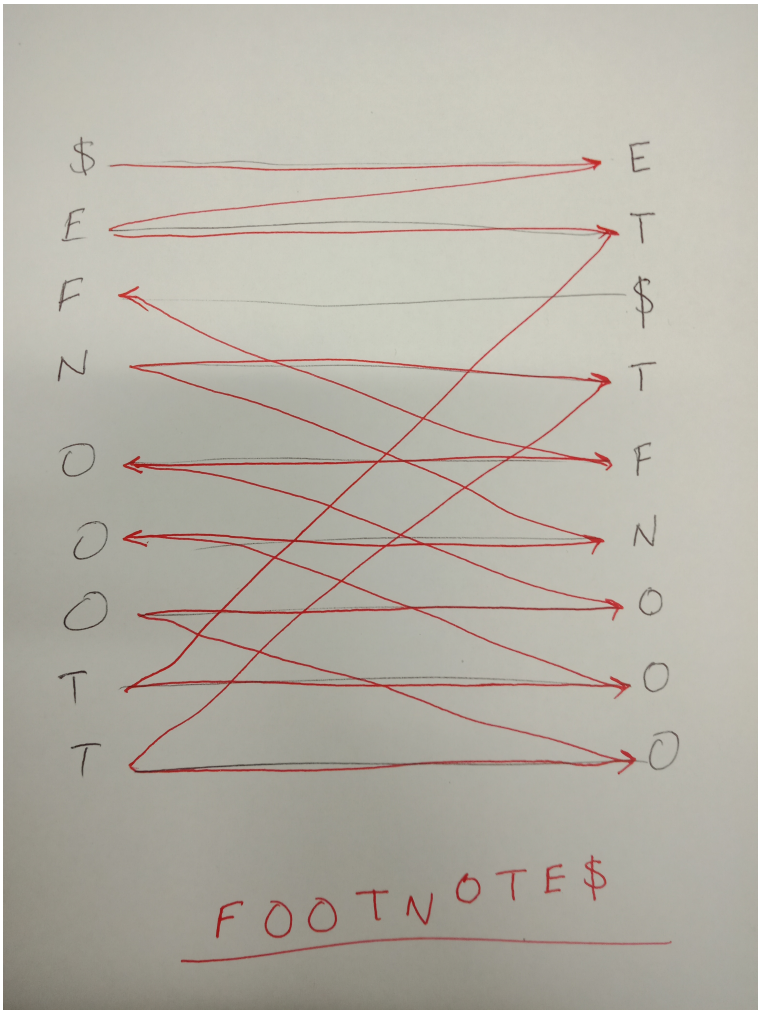
Figure 2: AVL Tree



This page is intentionally left blank. If needed, you can use this space to write answers.

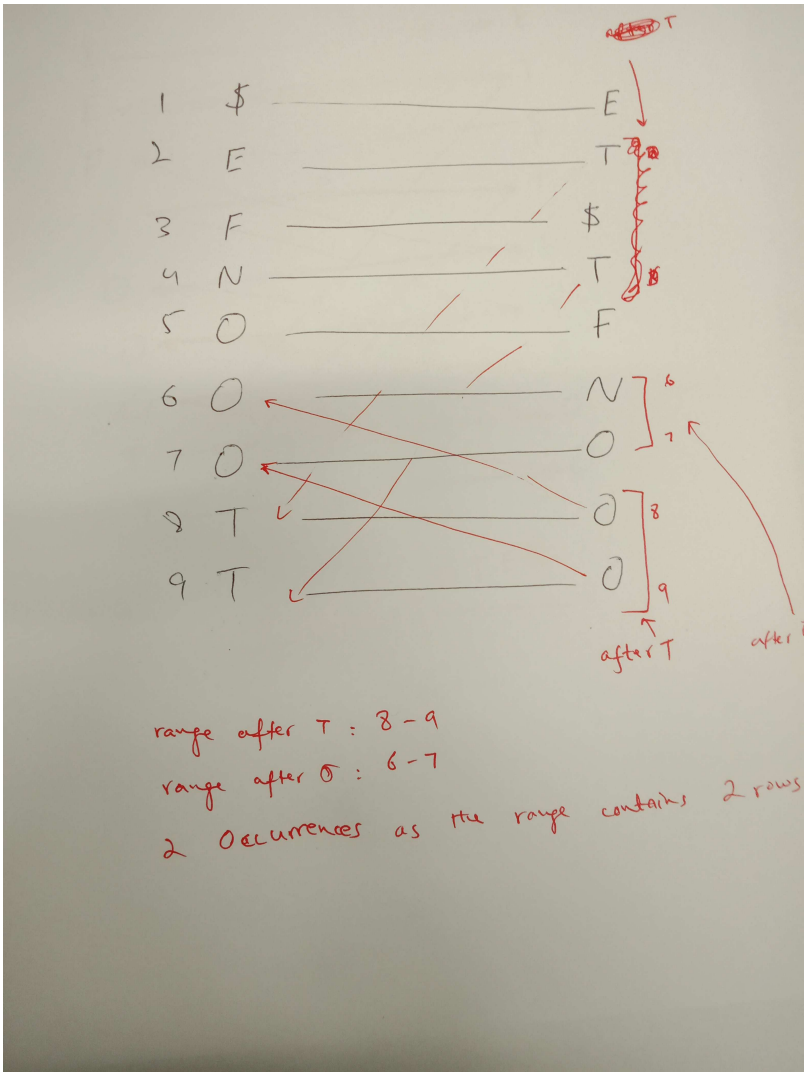
5. The Burrows-Wheeler Transform of a string is "ET$TFNOOO". Answer the following questions.

   (a) (4 marks) Use Last-First columns mapping to reconstruct the original string. The progression of your worked out steps of Last-First columns mapping must be clearly demonstrated in your answer using row numbers and arrows.



This page is intentionally left blank. If needed, you can use this space to write answers.

(b) (4 marks) Using the backwards search strategy, show how you would search the query pattern "OT" and count the number of its occurrences in the original string. Clearly demonstrate how you would update the range after processing each character in the query pattern and how you would count the number of its occurrences.



4

This page is intentionally left blank. If needed, you can use this space to write answers.

6. (a) (3 marks) What will be the suffix array of BAGGAGE? You are not required to show the working – it is sufficient to show ONLY the suffix array.

`[8,5,2,1,7,4,6,3]` OR `[5,2,1,7,4,6,3]` if $ is not considered

(b) (3 marks) What are the space and time complexities of constructing a suffix array using the prefix doubling approach? Briefly justify your answer.

The time complexity is $O(N \log^2 N)$. This is because each sorting takes $O(N \log N)$ as comparison cost is $O(1)$ and sorting is done $O(\log N)$ times.

The space complexity is $O(N)$ because only the original string, suffix array and rank array are required each of which is $O(N)$.

This page is intentionally left blank. If needed, you can use this space to write answers.

7. (a) (5 marks) Consider the graph G shown in Figure 3. Show how Kurskal's algorithm computes the minimum spanning tree in graph G by drawing figures after each edge is added to the tree. You will need to draw five figures.

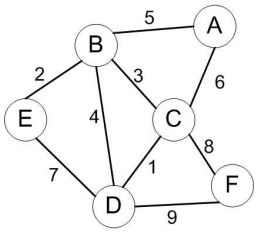
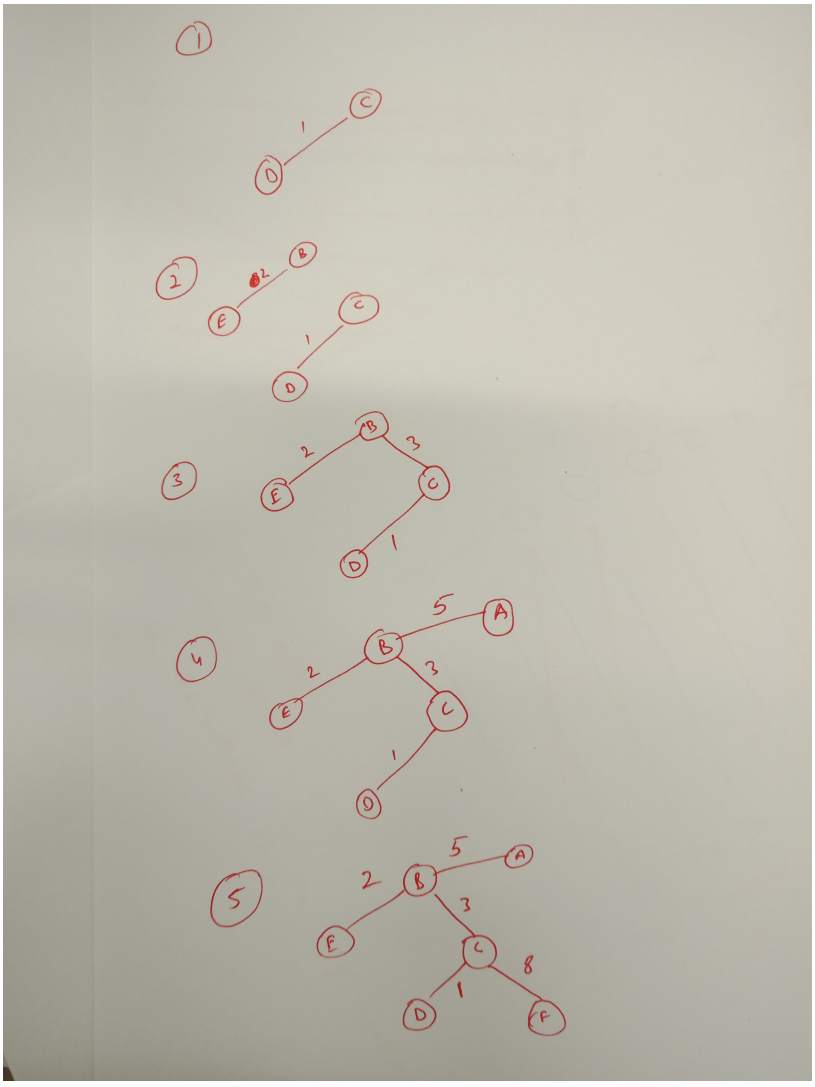


Figure 3: Graph G



5

This page is intentionally left blank. If needed, you can use this space to write answers.

(b) (5 marks) Write the invariant of Kruskal's algorithm and use it to prove that Kruskal's algorithm correctly computes a minimum spanning tree.

Invariant: Finalized is a subset of a minimal spanning tree

- Invariant is initially true when Finalized is empty
- At an arbitrary step, the algorithm combines two sets (UNION_SETS) say S1 and S2 using an edge <D,F>.
- Assume that adding <D,F> is an incorrect choice.
- The sets S1 and S2 must be connected by at least one edge in every spanning tree.
- Let M be a minimum spanning tree that does not contain <D,F>.
- Let <D,G> be the first edge on the path that connects S1 and S2 in the minimum spanning tree M.
- We will get a spanning tree if we add <D,F> in M and remove <D,G>. LetâĂŹs call this spanning tree T.
- The weight of T is smaller or equal to M because the weight of <D,F> is smaller or equal to <D,G>.
- Hence, T is also a minimum spanning tree if M is a minimum spanning tree, i.e., the invariance holds after adding <D,F> in Finalized

5

This page is intentionally left blank. If needed, you can use this space to write answers.

8. 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.

   (a) (4 marks) Write pseudocode for the Floyd-Warshall algorithm that finds **shortest distance** between every pair of vertices in the graph.

```
dist[][] = E
for vertex k in 1..V:
    for vertex i in 1..V:
        for vertex j in 1..V:
            dist[i][j] = min(dist[i][j], dist[i][k] + dist[k][j])
```

   (b) (2 marks) What are the worst-case space and time complexities for Floyd-Warshall algorithm.

$O(V^2)$ is the space complexity and $O(V^3)$ is the time complexity.

This is the end of the test.

0