Office Use Only

MONASH University

SAMPLE EXAM - 3
Exam Period (August 3018)

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 **x** Clayton ☐ Parkville ☐ Peninsula
☐ Monash Extension ☐ Off Campus Learning **x** 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, or a breach of instructions under Part 3 of the Monash University (Academic Board) Regulations.

AUTHORISED MATERIALS

OPEN BOOK	☐ **YES**	**x NO**
CALCULATORS	☐ **YES**	**x NO**
SPECIFICALLY PERMITTED ITEMS **if yes, items permitted are:**	☐ **YES**	**x 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 average-case time complexity of Quick Sort algorithm? Briefly justify your answer.

   Average-case: $O(N \log N)$;
   Assume that the pivot is always in the middle half of the array after partitioning. In this case, the worst possible partition sizes are N/4 and 3N/4. When pivot is always in the middle half, the height of the tree is $O(log_{4/3}N)$. Since the cost at each level is $O(N)$, the total cost is $O(Nlog_{4/3}N) = O(NlogN)$. When pivot is in the middle half only 50% of the times (average-case), the cost is at most double. So, the average-case cost is $O(NlogN)$.

   (b) (3 marks) Assume that we are calling Bellman Ford algorithm **for every vertex** v of the graph to compute all pairs-shortest distances on a weighted 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 of calling Bellman-Ford V times: $O(V(EV)) = O(EV^2)$,
   For dense graphs, $O(E) = O(V^2)$. Therefore, complexity is $O(V^4)$.

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. What is the min-cut on this flow network and what is its capacity? What is the maximum possible flow in this network?

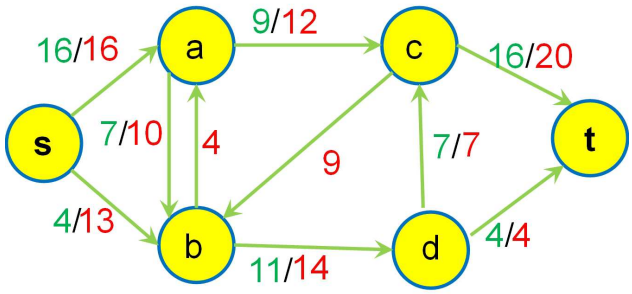


Figure 1: Flow Network

Min-cut is $S = \{s, a, b, d\}$ and $T = \{c, t\}$. Its capacity is 23. The maximum possible flow is 23.

(d) (2 marks) What is the time complexity of Ford Fulkerson algorithm assuming Breadth First Search is used to find augmenting paths? Give a brief reasoning.

Complexity: $O(EF)$ where F is the maximum flow of the network.
Cost for each iteration of while loop is O(V+E) (the cost of BFS and augmenting the flow) and the maximum possible iterations is F because flow increases by at least 1 in each iteration. Thus, the total cost is O(F(V+E)) which is $O(EF)$ because for connected graphs $O(E) >= O(V)$.

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

2. (5 marks) Given a capacity c and a set of items with their weights and values, the 0/1 knapsack problem is to pick items such that their total weight is at most c and their total value is maximized. You **cannot** pick any item more than once. 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 0/1 knapsack is to pick item A and item D with total weight 10kg and total value \$590.

   Write pseudocode for a dynamic programming algorithm to solve the 0/1 knapsack problem.

```
Initialize Memo[ ][ ] to contain 0 for all indices
for i=1 to N:
    for c in 1 to C:
        excludedValue = Memo[i-1][c]
        includedValue = 0
        if weight[i] <= c:
            includedValue = values[i] + Memo[i-1][c - weight[i]]
        Memo[i][c] = max(excludedValue,includedValue)
```


5

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

3. (a) (4 marks) Use mathematical induction to prove the following:

$$\sum_{i=0}^{N} ar^i = a + ar + ar^2 + ar^3 + \cdots + ar^N = \frac{a\left(r^{N+1} - 1\right)}{r - 1}$$

Base case: N = 0

$a = \frac{a(r^{0+1}-1)}{r-1} = \frac{a(r-1)}{r-1} = a$

Inductive step

Assume it holds for a value k, i.e., $\sum_{i=0}^{k} ar^i = \frac{a(r^{k+1}-1)}{r-1}$. We show that it holds for $k+1$, i.e.,$\sum_{i=0}^{k+1} ar^i = \frac{a(r^{k+2}-1)}{r-1}$

$\sum_{i=0}^{k+1} ar^i = a + ar + ar^2 \cdots + ar^k + ar^{k+1}$

$= ar^{k+1} + \sum_{i=0}^{k} ar^i$

$= ar^{k+1} + \frac{a(r^{k+1}-1))}{r-1}$

$= \frac{ar^{k+1}(r-1)+a(r^{k+1}-1)}{r-1}$

$= \frac{ar^{k+2}-ar^{k+1}+ar^{k+1}-a}{r-1}$

$= \frac{ar^{k+2}-a}{r-1}$

$= \frac{a(r^{k+2}-1)}{r-1}$

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

(b) (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(2*N):
            value += i
        return value + mystery(N//2)
```

Recurrence relation:

$T(1) = a$

$T(N) = b * N + T(N/2)$ because the for loop takes $O(N)$ and the function then recursively calls itself with $N/2$.

Solution:

$T(N) = b * N + b * N/2 + T(N/4)$

$T(N) = b(N + N/2 + N/4) + T(N/8)$

$T(N) = b(N + N/2 + N/2^2 + \cdots + N/2^{k-1}) + T(N/2^k)$

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 $O(N)$.

(c) (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 the $O(\log N)$ is required to store the recursive calls in the stack.

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 **19.5** 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.

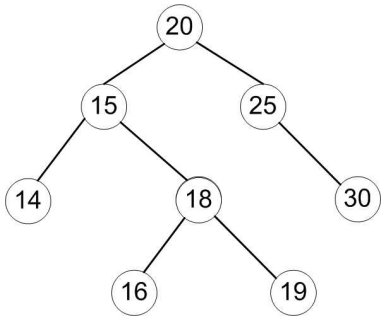
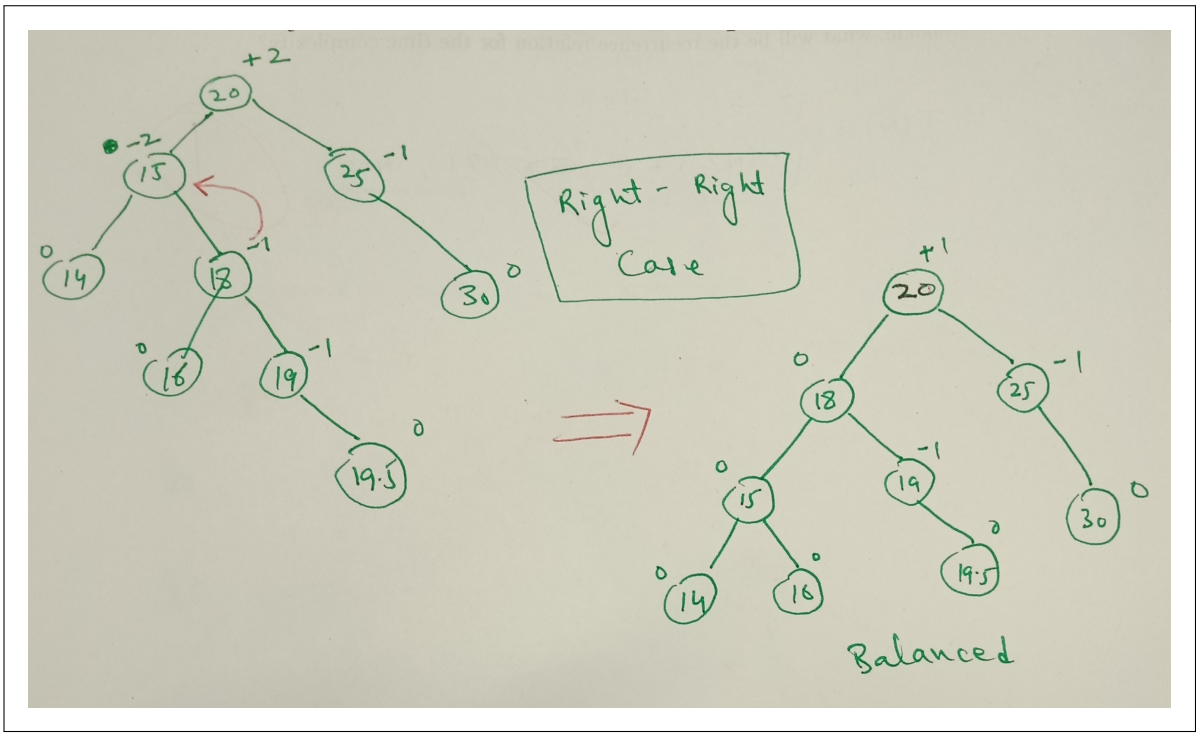


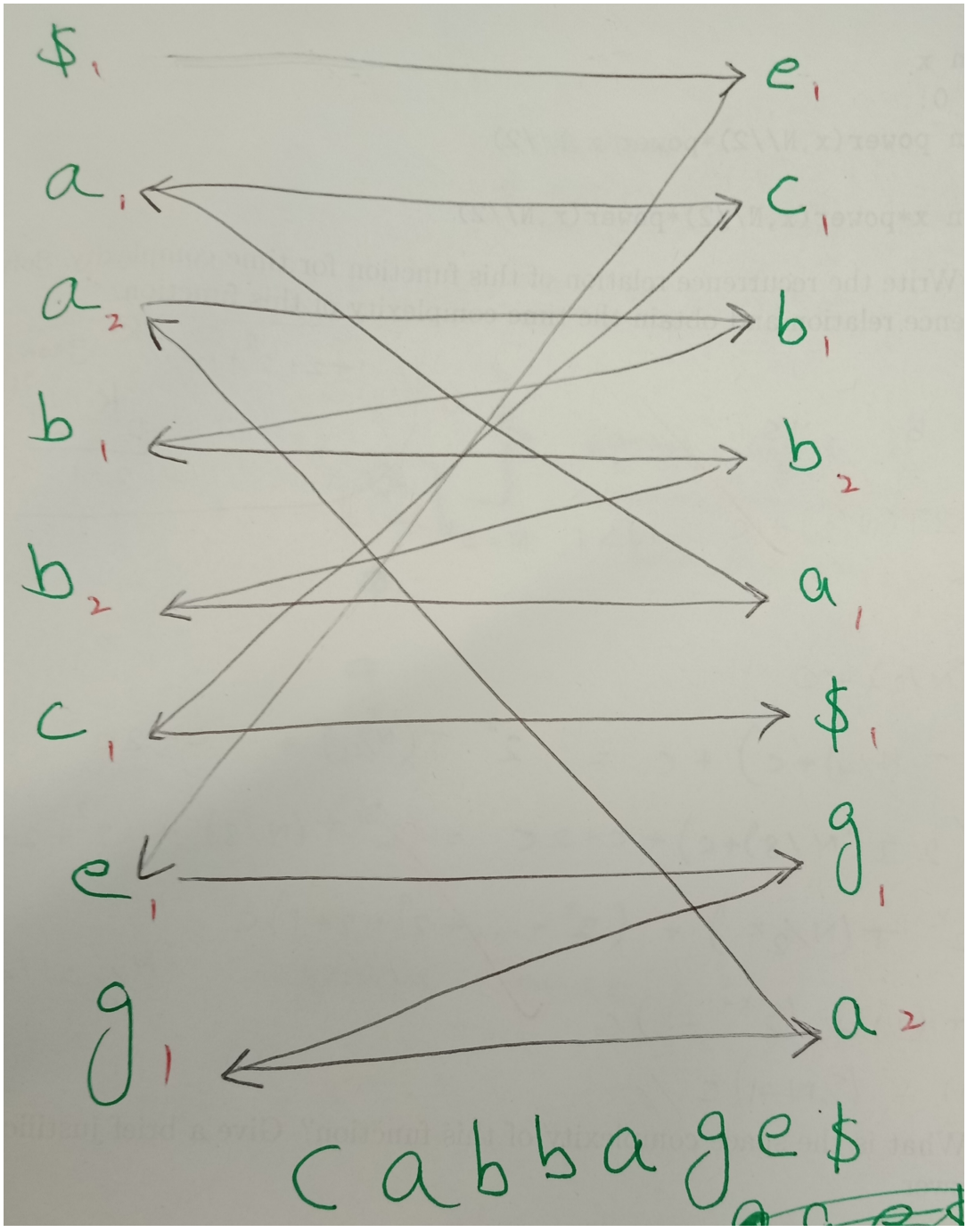
Figure 2: AVL Tree



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

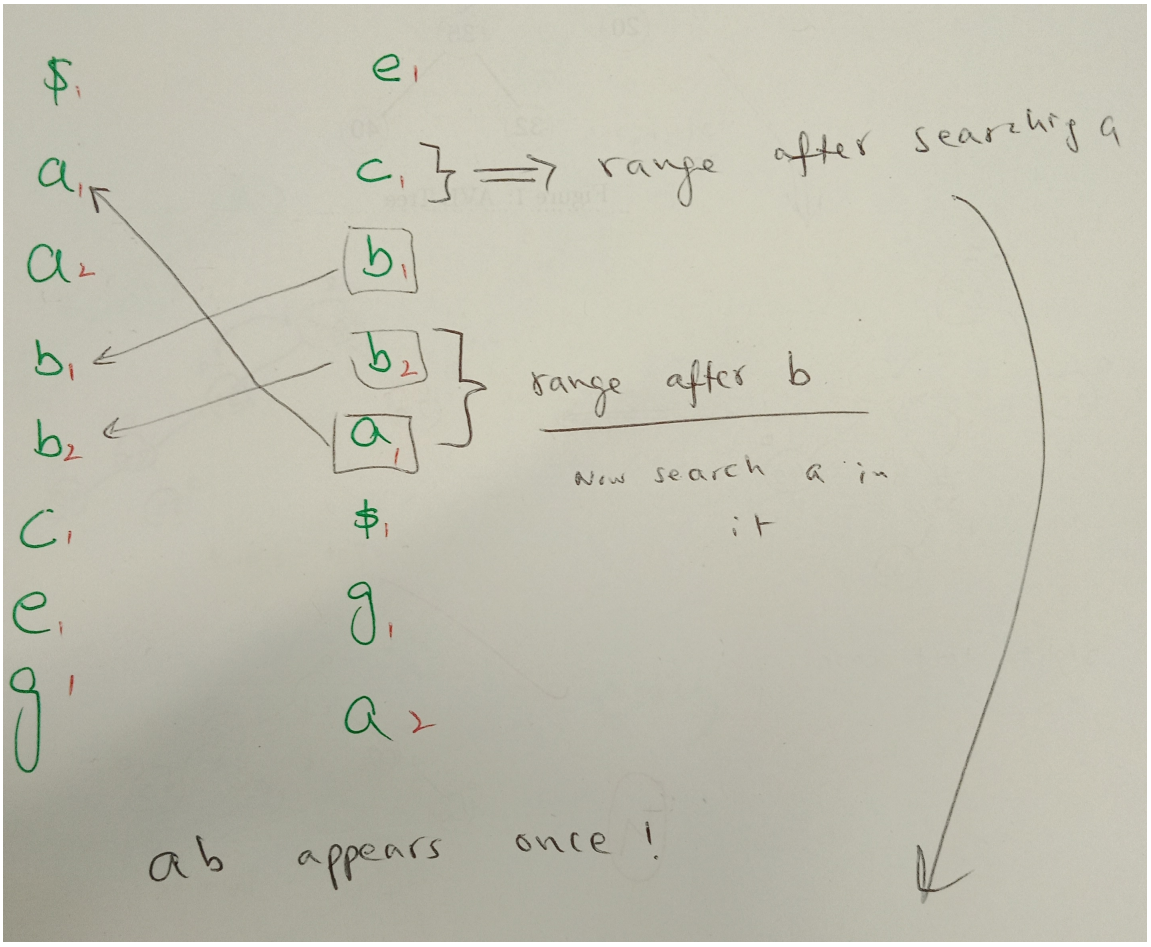
5. The Burrows-Wheeler Transform of a string is "ecbba$ga". Answer the following questions.

   (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 "ab" and count the number of its occurrences in the original string. Clearly demonstrate the range after processing each character in the query pattern and use this to 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 RABBIT? You are not required to show the working – it is sufficient to show ONLY the suffix array.

```
RABBIT$
1234567


7   $
2   ABBIT$
3   BBIT$
4   BIT$
5   IT$
1   RABBIT$
6   T$
[7,2,3,4,5,1,6]
```

(b) (3 marks) What are the space and time complexities of constructing a suffix array using the prefix doubling approach when insertion sort is used to sort at each step? Briefly justify your answer.

Each sorting takes $O(N^2)$ because there are $O(N^2)$ comparisons performed by insertion sort and each comparison takes $O(1)$. The sorting is performed $O(\log N)$ times. So, the total cost is $O(N^2 \log N)$.

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 Prim'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.

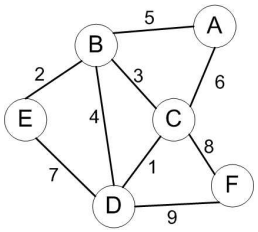
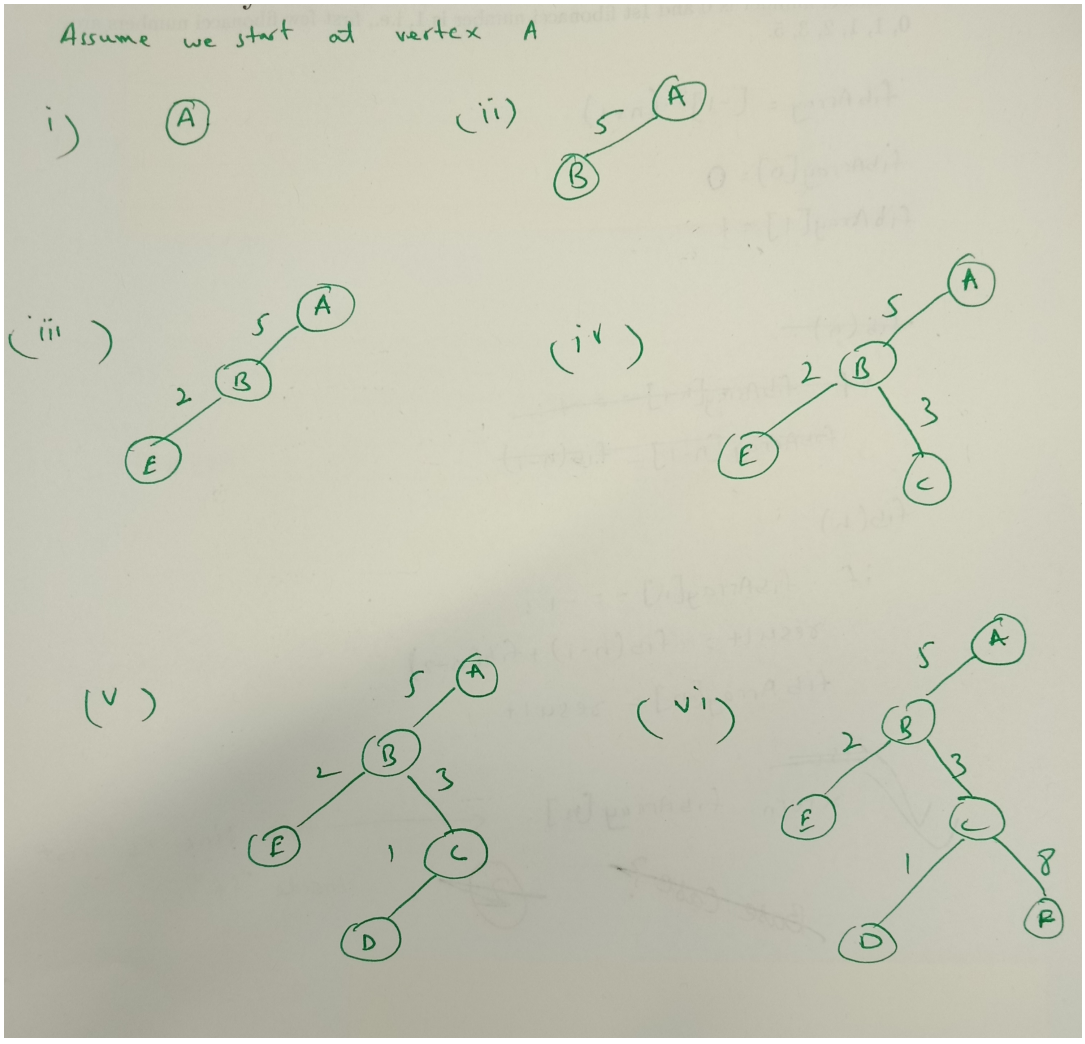


Figure 3: Graph G



The question asks to draw five figures. So, the first figure above is not necessary (and you can choose to start with any vertex of the graph).

5

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

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

Prim's Algorithm: Correctness

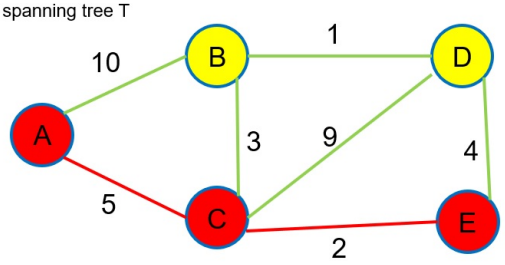
#INV: Finalized is a (growing) subset of a minimum spanning tree

Base Case:

- The invariance is true initially when Finalized is empty

Inductive step:

- Assume Finalized is currently a subset of a MST. We show that after Prim's algorithm adds an edge, invariance still holds
- Suppose the algorithm chooses a vertex E and an edge <C,E> having minimum weight w
- Assume {Finalized Union <C,E>} is **not** a subset of **any** minimum spanning tree. We show that this assumption is wrong.
- Let M be a minimum spanning tree that contains Finalized **but** excludes <C,E>.
  - E must be connected to Finalized in M (because M is a spanning tree). Since M does not contain <C,E>, there must be a path that connects Finalized (e.g., red vertices) with E (e.g., see blue edges).
  - Let <C,B> be the first edge on the path that connects Finalized to E.
- If we remove <C,B> from M and add <C,E> we will still get a spanning tree. Let this spanning tree be called T.
- Since the weight of <C,E> is smaller or equal to the weight of <C,B>, the weight of T is smaller than or equal to M. Hence, either M is not a minimum spanning tree or T is also a minimum spanning tree.
- Hence, Finalized after adding <C,E> is a subset of a minimum spanning tree T
  - i.e., the invariance holds after adding the edge <C,E>



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 Bellman-Ford algorithm that finds **shortest distances** between a source vertex to every other vertex in the graph.

```
dist[1...V] = infinity
dist[s] = 0
# STEP 2: Iteratively estimate dist[v] (from source s)
for i = 1 to V-1:
    for each edge <u,v,w> in the whole graph:
        est = dist[u] + w
        if est < dist[v]:
            dist[v] = est
            pred[v] = u

for each edge <u,v,w> in the whole graph:
    if dist[u]+w < dist[v] :
        return error; # negative edge cylce found in this graph

return dist[...], pred[...]
```

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

$O(V + E)$ is the space complexity and $O(VE)$ is the time complexity.

This is the end of the test.

0