NATIONAL UNIVERSITY OF SINGAPORE

CS2040S—Data Structures and Algorithms

AY2024/2025, Semester 2

Time Allowed: 2 hours

INSTRUCTIONS TO STUDENTS

1. The assessment paper contains **SIX (6) questions** and comprises **TWELVE (12) pages** including this cover page.
2. Weightage of each question is given in square brackets. The maximum attainable score is **70**.
3. This is a **CLOSED** book assessment, but you are allowed to bring **ONE** double-sided A4 sheet of notes for this assessment.
4. Please write your Student Number only on the **ANSWER BOOKLET**. Do not write your name. Write all your answers in the space provided in the **ANSWER BOOKLET**.
5. You are allowed to write with pencils, as long as it is legible.
6. **Marks may be deducted** for i) unrecognisable handwriting, and/or ii) excessively long code.

This page is intentionally left blank.

It may be used as scratch paper.

Question 1: MCQs [12 marks]

A. If we ran mergesort on a **sorted array of distinct** integers, the running time is (pick the smallest among the following options):

1. $O(1)$
2. $O(\log n)$
3. $O(n)$
4. $O(n \log n)$
5. $O(n \log^2 n)$
6. $O(n^2)$

[2 marks]

For the following statements, please indicate if they are **True** or **False**.

B. Consider hashing with chaining with a table size m, and n elements. If $n = m$, we cannot insert any more elements without resizing the table since it is full. [2 marks]

C. Consider a weighted (potentially negatively) directed graph where every node has in-degree 1. There exists such a graph for which running Dijkstra's will produce the wrong set of distances. [2 marks]

D. BFS runs in $O(|V|)$ time when given a complete, unweighted, undirected graph G with the vertex set V. [2 marks]

E. Given an undirected, weighted graph G with a negative cycle, Prim's algorithm does not necessarily output a correct minimum spanning tree. [2 marks]

F. Doing the following post-order traversal on a binary max-heap outputs the elements of the max-heap in sorted order. (Assume the last element is stored at index n).

```
void PostOrder(int current_index){
    if(current_index > n){
        return;
    }
    PostOrder(current_index * 2)
    PostOrder(current_index * 2 + 1)
    Print(current_index)
}
```

[2 marks]

Question 2: Quickfire [8 marks]

In the following questions, choose **one** algorithm that we should use to solve the problem as efficiently as possible (worst case if deterministic, expected running time if randomized):

A. Given an array of n integers, to find the k^{th} smallest item, we should use: [2 marks]

B. Given a directed acyclic graph G, to output a toposort of the graph, we should use: [2 marks]

C. To count the number of components in an undirected graph, we should use: [2 marks]

D. To detect negative cycles in a directed graph, we should use: [2 marks]

Question 3: Anagram Count [15 marks]

A pair of strings (s,z) is called an *anagram* pair if one is just a permutation of the characters of another, for example, ("silent", "listen") is an anagram pair, and so is ("cheater", "teacher"). On the other hand ("one", "ono") is not an anagram pair.

Suppose you are given two strings: a text $T[1..n]$ and a search string $S[1..m]$. **You may assume that $m < n$.** The *anagram substring count* of S in the text T is the number of (contiguous) substrings of T that form anagram pairs with S. For example, let $T =$ "*esleastealaslatet*" and $S =$ "*tesla*". Then observe that, among a total of thirteen contiguous substrings in T, exactly three (more specifically, $T[3..7] = least$, $T[6..10] = steal$, $T[12..16] = slate$) form anagram pairs with S, and thus the anagram substring count of S in T is 3.

Note: For this question, you can assume that all the characters in the strings are letters in the English alphabet (i.e. a to z). You may assume that the i^{th} letter corresponds to the value i. E.g., the letter a corresponds to the value 1, the letter d corresponds to the value 4, the letter z corresponds to the value 26.

A. Give a sketch of an algorithm that, given a pair of strings $s[1..p]$, $z[1..q]$, determines whether it is an anagram pair or not. Your algorithm **must run in worst-case time** $O(p+q)$. Prove that your algorithm runs within worst-case time $O(p+q)$. [4 marks]

Given any text string $T[1..n]$ and a positive integer m, build a data structure that supports the operation `AnagramSubstrCount`(S, T): given any string $S[1..m]$ of length m, returns the anagram substring count of $S[1..m]$ in $T[1..n]$.

B. Describe your data structure. Please specify any variables or data structures (e.g., trees, arrays, linked lists, stacks, queues) you might need. [2 marks]

C. Describe your algorithm (in pseudocode) to initialize (build) your data structure for input text $T[1..n]$ and a positive integer m. (Your algorithm should be as efficient as possible.) [4 marks]

D. What is the running time of your initialization algorithm? (If your algorithm is randomized, mention the expected running time.) [2 marks]

E. Describe your algorithm (in pseudocode) for `AnagramSubstrCount`(S). (Your algorithm should be as efficient as possible. If it is randomised, we will consider the expected running time.) [2 marks]

F. What is the running time of `AnagramSubstrCount`? (If your algorithm is randomized, mention the expected running time.) [1 mark]

Question 4: Islands [14 marks]

Consider an $n \times n$ grid H of tiles, where $n \geq 1$. For $1 \leq i, j \leq n$, the tile on the i^{th} row and j^{th} column comes with height $H[i][j]$. The heights can be any non-negative value.

We can choose to set the water level to be at a certain height h. If we do, all tiles that are height $\leq h$ are considered **flooded**. The remaining tiles are considered to be **safe** (not flooded). Two tiles (r_1, c_1) and (r_2, c_2) are considered to be **connected** if:

1. Both tiles are safe; and
2. ($r_1 = r_2$ and $|c_1 - c_2| = 1$) **or** ($c_1 = c_2$ and $|r_1 - r_2| = 1$)

An **island** is a maximal group of safe tiles such that any two safe tiles in the group can reach each other (we can go from a tile to any another tile if they are connected), and no other tiles outside of that group can reach it. An island can also be just a single tile, if that tile is not connected to any other tile.

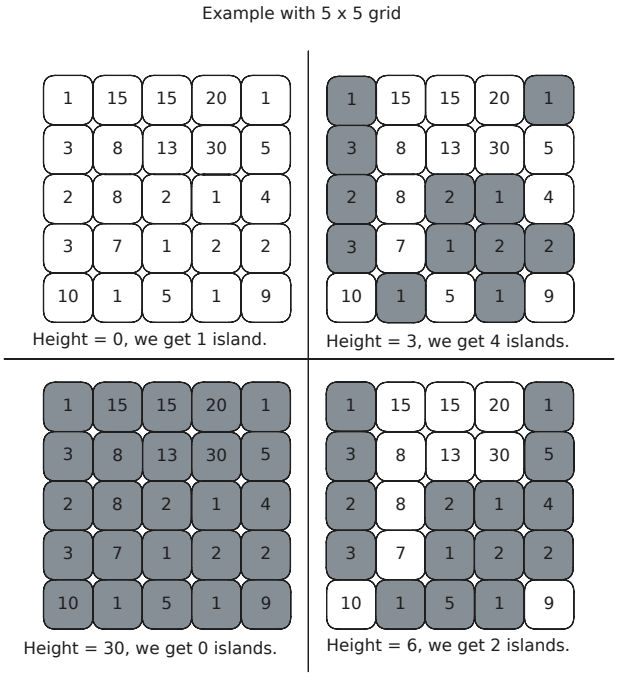


Figure 1: Example with a 5 by 5 grid

Take special note that on the top right example in figure 1, there are 4 islands, because the tile on the bottom left is not connected (diagonally) to the island at the top.

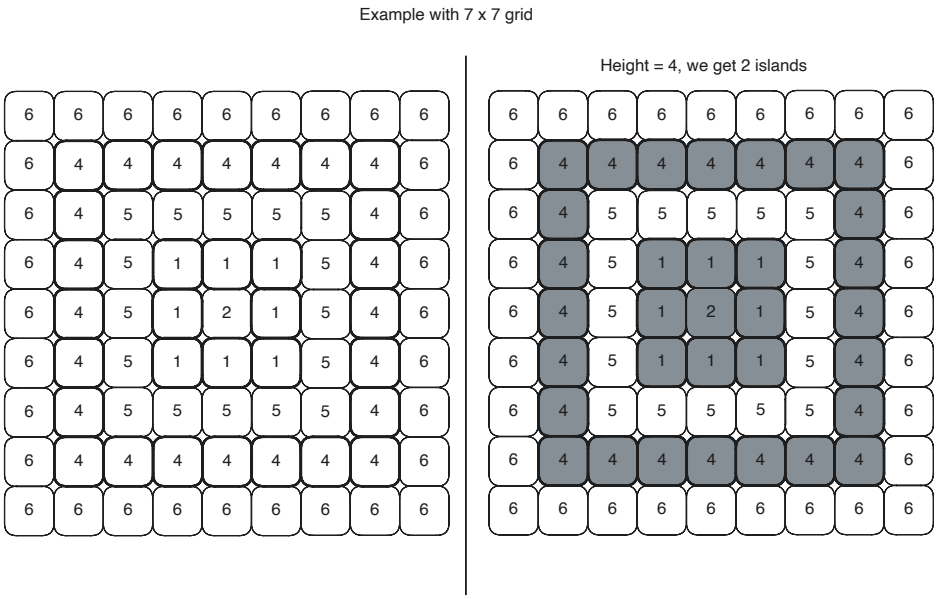


Figure 2: Example with a 9 by 9 grid

Main task: Given as input $H[1..n][1..n]$ and an integer k such that $1 \leq k \leq n^2$. Give an algorithm that is as efficient as possible in finding and outputting the **minimum height** h of which, by setting that to be the water level, there are at least k islands.

A. Describe an algorithm that takes as input $H[1..n][1..n]$ and a height h, and outputs the number of islands. [6 marks]

B. What is the time complexity of your algorithm in Part (A)? If it is deterministic, state the worst case time complexity. If it is randomised, state the expected running time. [2 marks]

C. Describe an algorithm that solves the main task. You may use your answer to Part (A) as a subroutine. [4 marks]

D. What is the time complexity of your algorithm in Part (C)? If it is deterministic, state the worst case time complexity. If it is randomised, state the expected running time. [2 marks]

Question 5: Traffic Trouble [11 marks]

Consider a directed graph G with **positively** weighted edges that represents some internet network. We are going to send packets over the network from source node s. If node u has an edge to node v, then the time taken to send a packet from node u to node v is given by the edge weight $w(u,v)$. Otherwise, node u cannot send a packet to node v.

Nodle can convert an existing link between nodes into a fastlane. When a packet enters a fastlane, the cost of traversing between 2 nodes becomes 0. Nodle can create up to k fastlanes, where k is a non-negative integer. However, it is expensive to create fastlanes, so Nodle would like to create as few fastlanes as possible. At the same time, he only needs to ensure that packets can traverse between a specified pair of nodes in the network can do so within some time limit L.

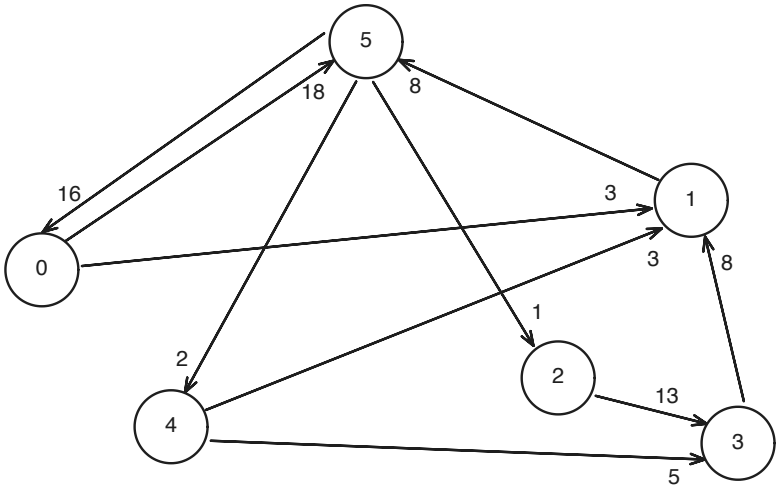


Figure 3: Example of a possible graph

(**Example 1:**) For example, if we wanted to traverse from node 0 to node 3, the shortest path from node 0 to 3, is 18. In order to make this possible within time limit $L = 13$, we can make the path between nodes 1 and 5 into a fastlane and the following path will now take a total time of 10:

1. $0 \rightarrow 1$ with weight 3
2. $1 \rightarrow 5$ with weight 8 (**made free**)
3. $5 \rightarrow 4$ with weight 2
4. $4 \rightarrow 3$ with weight 5

In other words, if we did not make any edges free, we won't be able to make the time limit. So $k = 1$ is the answer, for time limit $L = 13$.

(**Example 2:**) On the other hand, the shortest path from node 0 to node 5 is 11. So to make it within time limit $L = 10$, we can make the path between node 0 to 5 as fastlane and take the following path:

1. $0 \rightarrow 5$ with weight 18 (**made free**)

This suggests that for time limit $L = 10$, $k = 1$.

Your task: Given as input:

1. A simple, directed graph $G = (V, E)$ with **positively** weighted edges.
2. A source node $s \in V$, and target node $t \in V$
3. A time limit L.

Design an algorithm, as efficient as possible, to find the minimum value of k, i.e., the minimum number of fastlanes to use, such that the total time taken for going from node s to node t is at most L. Output IMPOSSIBLE if it is impossible to make it from s to t within time L, irrespective of the value of k.

A. Describe your algorithm that solves the main task. You do not have to re-implement any algorithm covered in class. If you wish to invoke an existing algorithm, mention what it is, and state clearly any modifications you wish to make to those algorithms. [9 marks]

B. What is the time complexity of your algorithm? If it is deterministic, state the worst case time complexity. If it is randomised, state the expected running time. [2 marks]

Question 6: Never Skip Leg Day! [10 marks]

Nodle has a new conundrum. Every day, Nodle has to pick between one of 3 routines to do at the gym: (A) Push, (B) Pull, (C) Legs. (For simplicity we will refer to them as routines A, B, and C respectively). He needs to plan out his sequence of routines for n days. You are given 3 arrays, each containing n integers. The first, second, and third array, each represent the score that Nodle gets for doing routines A, B, and C respectively on each day.

For example, for 5 days, we might be given as arrays:

$$\begin{aligned} score\_A &= [\ \ 1, \ -3, \ \ 2, \ 10, \ \ 2\ ] \\ score\_B &= [\ 16, \ \ 2, \ -5, \ \ 2, \ 51\ ] \\ score\_C &= [\ 61, \ 62, \ 40, \ 30, \ 50\ ] \end{aligned}$$

Nodle needs to pick the sequence of routines to do so that:

1. Every day he only picks **exactly 1** type of routine to do.
2. Every routine is done **at most 2** days in a row.
3. He maximises the total possible score obtainable, which is the sum of the points earned each day.

For example, if Nodle chooses to do the following:

$$\begin{aligned} score\_A &= [\ \ 1, \ -3, \ \boxed{2}, \ 10, \ \ 2\ ] \\ score\_B &= [\ 16, \ \ 2, \ -5, \ \ 2, \ 51\ ] \\ score\_C &= [\ \boxed{61}, \ \boxed{62}, \ 40, \ \boxed{30}, \ \boxed{50}\ ] \end{aligned}$$

Then notice every routine is done at most 2 days in a row (even if routine C was done a total of 4 times, it was never done 3 times in a row), and this sequence earns a total of $61+62+2+30+50 = 205$ points.

Notice that we **cannot** use the following sequence instead:

$$\begin{aligned} score\_A &= [\ \ 1, \ -3, \ \ 2, \ \boxed{10}, \ \ 2\ ] \\ score\_B &= [\ 16, \ \ 2, \ -5, \ \ 2, \ 51\ ] \\ score\_C &= [\ \boxed{61}, \ \boxed{62}, \ \boxed{40}, \ 30, \ \boxed{50}\ ] \end{aligned}$$

Because even though this earns a total of $61+62+40+10+50 = 223$ points, routine C is being done 3 times in a row in this sequence.

For the given example, the maximum obtainable score is $61+62+2+30+51 = 206$, using the sequence: C, C, A, C, B.

Your goal: Given 3 arrays containing n integers each, output the maximum obtainable score.

A. Describe your algorithm that solves the main task. You do not have to re-implement any algorithm covered in class. If you wish to invoke an existing algorithm, mention what it is, and state clearly any modifications you wish to make to those algorithms. [8 marks]

B. What is the time complexity of your algorithm? If it is deterministic, state the worst case time complexity. If it is randomised, state the expected running time. [2 marks]

Question 7: Bonus Question, Optional: [0 marks]

A. You are blindfolded (and you can't see a thing). There are 100 coins lying on a table. Of these coins, 80 are tails-up and 20 are heads-up. You need to divide them into two groups of x and y each (with $x+y=100$). You can pick any coin and flip it (if it is tails, it will become head and if it is head, it will become tail), but you can't see so you don't know which coin you are flipping. In the end, both groups should contain equal number of heads. [0 mark]

— E N D O F P A P E R —

CS2040S—Data Structures and Algorithms

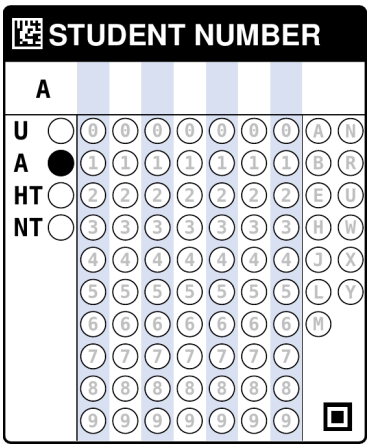
Final Exam Answer Sheet

AY2024/2025, Semester 2

Time allowed: 2 hours

Instructions (please read carefully):

1. Write down your **student number** on the right and using ink or pencil, shade the corresponding circle in the grid for each digit or letter. DO NOT WRITE YOUR NAME!
2. This answer booklet comprises **SIXTEEN (16) pages**, including this cover page.
3. All questions must be answered in the space provided; no extra sheets will be accepted as answers. You may use the extra page behind this cover page if you need more space for your answers.
4. You must submit only the **ANSWER SHEET** and no other documents. The question set may be used as scratch paper.
5. The questions are provided in the question booklet, and if there is any inconsistency in the question, please refer to the question booklet. If there is any inconsistency in the answers, refer to the answer sheet.
6. You are allowed to use pencils, ball-pens or fountain pens, as you like as long as it is legible (no red color, please).
7. Please fill in the bubbles properly. **Marks may be deducted** for unclear answers.



For Examiner's Use Only

Question	Marks
Q1	/ 12
Q2	/ 8
Q3	/ 15
Q4	/ 14
Q5	/ 11
Q6	/ 10
Total	/ 70

Question 1A Running time for mergesort on sorted array of distinct integers [2 marks]

◯ $O(1)$ ◯ $O(\log n)$ ◯ $O(n)$
◯ $O(n \log n)$ ◯ $O(n \log^2 n)$ ◯ $O(n^2)$

Question 1B Cannot insert anymore elements without resizing [2 marks]

◯ True ◯ False

Question 1C Consider a weighted (potentially negatively) directed graph where every node has in-degree 1. There exists such a graph for which running Dijkstra's will produce the wrong set of distances. [2 marks]

◯ True ◯ False

Question 1D BFS runs in $O(|V|)$ time [2 marks]

◯ True ◯ False

Question 1E Prim's algorithm does not necessarily output correct minimum spanning tree [2 marks]

◯ True ◯ False

Question 1F Doing the following post-order traversal on a binary max-heap outputs the elements of the max-heap in sorted order. (Assume the last element is stored at index n).[2 marks]

◯ True ◯ False

Question 2A Given an array of n integers, to find the k^{th} smallest item, we should use: [2 marks]

Question 2B Given a directed acyclic graph G, to output a toposort of the graph, we should use: [2 marks]

Question 2C To count the number of components in an undirected graph, we should use: [2 marks]

Question 2D To detect negative cycles in a directed graph, we should use: [2 marks]

Question 3A Describe your algorithm for anagram pair checking (in pseudocode). Explain why it runs in worst-case time $O(p+q)$. [4 marks]

```
bool AnagramPair(String s, String z){
    /* Your pseudocode here */




}
```

Question 3B Declare your variable or data structures. [2 marks]

```
class DataStructure {
            // Declare any data structures or variables here




}
```

Question 3C Describe your Initialization algorithm (in pseudocode). [4 marks]

```
class DataStructure {
    void Init(String T, int m){
        // Your pseudocode here;









    }
}
```

Question 3D What is the running time of your initialization algorithm? [2 marks]

Question 3E Describe your algorithm (in pseudocode) for AnagramSubstrCount.[2 marks]

```
class DataStructure {
    int AnagramSubstrCount(String S){
        /* Your pseudocode here */









    }
}
```

Question 3F What is the running time for your implementation of AnagramSubstrCount? [1 marks]

Question 4A Give your algorithm that takes as input $H[1..n][1..n]$ and a height h, and outputs the number of islands. [6 marks]

```
int num_islands(int[][] tile_heights, int h){

}
```

Question 4B

[2 marks]

Question 4C Give your algorithm that solves the main task. You may use your answer to the previous sub-part as a subroutine. [4 marks]

```
int max_islands(int[][] tile_heights, int k){

}
```

Question 4D

[2 marks]

Question 5A Your algorithm that solves the main task. [9 marks]

```
int traffic_trouble(ArrayList<ArrayList<int>> adjacency_list, int L){

}
```

Question 5B

[2 marks]

Question 6A Your algorithm that solves the main task. [8 marks]

Question 6B

[2 marks]

Question 7A Describe your strategy here [0 marks]

This page is intentionally left blank.

It may be used as scratch paper.

This page is intentionally left blank.

It may be used as scratch paper.

— END OF ANSWER SHEET —

Question 1A Running time for mergesort on sorted array of distinct integers [2 marks]

○ $O(1)$ ○ $O(\log n)$ ○ $O(n)$
● $O(n \log n)$ ○ $O(n \log^2 n)$ ○ $O(n^2)$

Question 1B Cannot insert anymore elements without resizing [2 marks]

○ True ● False

Question 1C Consider a weighted (potentially negatively) directed graph where every node has in-degree 1. There exists such a graph for which running Dijkstra's will produce the wrong set of distances. [2 marks]

● True ○ False

Question 1D BFS runs in $O(|V|)$ time [2 marks]

○ True ● False

Question 1E Prim's algorithm does not necessarily output correct minimum spanning tree [2 marks]

○ True ● False

Question 1F Doing the following post-order traversal on a binary max-heap outputs the elements of the max-heap in sorted order. (Assume the last element is stored at index n).[2 marks]

○ True ● False

Question 2A Given an array of n integers, to find the k^{th} smallest item, we should use: [2 marks]

Quickselect

Question 2B Given a directed acyclic graph G, to output a toposort of the graph, we should use: [2 marks]

DFS

Question 2C To count the number of components in an undirected graph, we should use: [2 marks]

DFS or BFS

Question 2D To detect negative cycles in a directed graph, we should use: [2 marks]

Bellman-Ford

Question 3A Describe your algorithm for anagram pair checking (in pseudocode). Explain why it runs in worst-case time $O(p+q)$. [4 marks]

```
bool AnagramPair(String s, String z){

    Build frequency vector for s and z
    Check whether they are equal
}
```

Generally any solution that obtains $O(p+q)$ runtime gets full 3.5 marks (out of 4). There were (really) many errorneous solutions like adding up the characters and assuming that two strings are anagrams of each other if and only if the sum is the same. This is definitely not correct. Any solution that is close to $O(p+q)$ (e.g. sorting the characters first) gets 1 partial mark. Super inefficient solutions get 0.

The remaining 0.5 marks are awarded for explanation for the runtime.

Question 3B Declare your variable or data structures. [2 marks]

```
class DataStructure {
    // Declare any data structures or variables here
    Hash table to store frequency vectors of
    all substrings of size m
}
```

Generally:

1. Any solution that uses a hashtable that maps frequency vectors to counts gets full 2 marks.
2. Any solution that uses a 2D array in the correct way gets 1.5 marks.
3. Any solution that uses a tree for character frequency or uses prefix sums gets 1.5 marks.

Marks are deducted for unneccesary space usage: -0.5m

Question 3C Describe your Initialization algorithm (in pseudocode). [4 marks]

```
class DataStructure {
    void Init(String T, int m){
        // Your pseudocode here;
        Start processing T from the beginning.
        Build a frequency vector for the first m-length substring.
        Then, update the vector for each new character.
        Each vector is a 26 log m -bit integer, and thus
        can easily be used as keys.
        Add all of them into a has table H with
        their count (no. of times a frequency vector appears).
    }
}
```

1. Any solution that: adds the frequency vector for each m-length substring of T gets full 4 marks.
2. Uses a tree instead of a hash table gets 3 out of 4 marks.
3. Any solution that does not use Init properly (but is otherwise correct) gets 2 out of 4 marks.

Question 3D What is the running time of your initialization algorithm? [2 marks]

$O(n)$.

Question 3E Describe your algorithm (in pseudocode) for AnagramSubstrCount.[2 marks]

```
class DataStructure {
    int AnagramSubstrCount(String S){
        /* Your pseudocode here */
        Construct frequency vector of S.
        Check the count in the corresponding cell in H.
    }
}
```

1. Full 2 marks for model solution above. Or using radix sort (which basically is building the frequency vector).
2. 1.5 marks for solutions that basically do: "for each frequency vector, check whether equal to frequency vector of S, if so, count++"
3. 1 mark for solutions that basically do: "for each m length substring of T, if substring and S is an anagram pair, then count++"
4. 1.5 marks for solutions that basically do: "sort S, lookup count based on sorted S"

Marks are subtracted for reasons such as:

1. suboptimal frequency vector computation
2. incorrect count due to unique substrings
3. checking based on existence of characters rather than count/frequency of characters
4. sub-optimal/inefficient in general.

Question 3F What is the running time for your implementation of AnagramSubstrCount? [1 marks]

$O(m)$

Question 4A Give your algorithm that takes as input $H[1..n][1..n]$ and a height h, and outputs the number of islands. [6 marks]

Solution 1:

```
int num_islands(int[][] tile_heights, int h){
    1. Treat each tile that is above height h as a node.
    2. Draw an edge between two nodes if they are
        next to each other or above/below each other.
    3. Run BFS/DFS to count the number of components.
    4. Return the count.
}
```

Solution 2:

```
int num_islands(int[][] tile_heights, int h){
    1. For each tile, add 1 to the island count.
    2. Then, check for all 4 neighbours:
        2.1 if the neighbour is above water,
                and they are not part of the same component:
        2.1.1 Subtract 1 from the island count,
                and union the two components.

}
```

Incorrect but partials marks given in case of:

- Running BFS/DFS but not specifying how the graph was constructed. (2 marks)
- Constructing a graph wrongly where it'll return the wrong count, but still using BFS or DFS to do component counting. (2 marks)
- Assuming that UFDS returns the number of sets. This is not an operation that UFDS provides for free. (5 marks)
- Almost correct solutions. (subject to discretion of the grader) (4 marks)
- Partially correct solutions. (subject to discretion of the grader) (2 marks)

Question 4B

[2 marks]

$O(n^2)$

Question 4C Give your algorithm that solves the main task. You may use your answer to the previous sub-part as a subroutine. [4 marks]

Question was designed with a slider (for complexity) in mind:

- $O(max\_height \times n^2)$ or slower. (2 marks)
- $O(n^4 \times n^2)$. (3 marks)
- $O(n^2 \times \log(n))$. (4 marks)

Decent (and most common) solution:

```
int max_islands(int[][] tile_heights, int k){
    1. Collect all possible heights in a list. Sort it.
    2. For each possible height (in increasing order),
        run the solution from part A,
        and if the number of islands is >= k, return this height.

}
```

Optimal solution:

```
int max_islands(int[][] tile_heights, int k){
    1. Sort the tiles by their heights, initially all tiles are not nodes.
    2. Initialise a UFDS where all nodes are disjoint.
    3. Group the tiles based on their heights,
        and consider the tiles in decreasing heights.
    4. For the next possible height, consider all tiles of that height.
    5. For each tile, add 1 to the island count.
    6. Then, check all 4 neighbours, if the neighbour is above water,
            and they are not part of the same component,
            subtract 1 from the island count,
            and union the two components.
    7. If the number of islands currently is >= k,
        then we store this height as the h to output.

}
```

A lot of submissions incorrectly either used binary search or peak finding to find the smallest height h for which there are at least k islands. Their rationale being that "as the water height increases, the number of islands monotonically increases then monotonically decreases". This is incorrect! A 10 by 10 counterexample is where the first row of tiles is given as $[40, 24, 32, 13, 14, 10, 12, 8, 16, 1]$, and where all other tiles on all other rows are height 0. Notice that as the water height increases, the number of islands moves between values 0, 1, 2, 3 multiple times.

For example, at a water height of 10, there are 3 islands, Then at a water height of 12, there are 2 islands. Then at a water height of 13, there are again 3 islands. For this reason, a binary search or peak-finding-based approach will not work. These submissions were given 0 marks for 4C.

Other errorneous solutions also got 0.

Question 4D

[2 marks]

$O(n^4)$

Question 5A Your algorithm that solves the main task. [9 marks]

```
int traffic_trouble(ArrayList<ArrayList<int>> adjacency_list, int L){
    Let n be the number of nodes, create n + 1 copies of the graph,
    call each copy a layer.
    Refer to them as layers $0$, $1$, ..., $n$.

    If in the original graph, node u has an edge to node v.
    For each layer, draw an edge 0 weight edge from node u in
    layer i to node v in layer i + 1.

    Let s0 be the respective node for node s in layer 0,
    and in general let ti be the respective node for node t in
    layer i.

    Run Dijkstra with node s0 on this n + 1 layer graph,
    and check the distances to nodes t0, t1, t2, ..., tn.

    Find the smallest i for which the distance to node
    ti <= L and t(i+1) > L, and return it.
    Otherwise, if no such i exists, output IMPOSSIBLE.
}
```

Note: This is equivalent to the superpowers question, but on a graph instead of the maze.

1. Optimal answer $O(VE\log(V))$: Copy the graph V times, connect free edges (or 0 weighted edges) between levels. Run Dijkstra once. Find the first i such that $dist(s,t_i) <= T$. Output that. (9 marks)
2. Less optimal $O(VE\log^2(V))$: Similar to optimal, but run it $O(\log(V))$ times for binary search for k. (7 marks)
3. Less optimal $O(V^2E\log(V))$: Similar to optimal, but run it $O(V)$ times to linear search for k. (6 marks)
4. Less optimal $O(E^2\log(V))$: Copy the graph E times, and run Dijkstra once. (6 marks)

Marks subtracted for:

1. Minor errors: -1 marks for communication issues or other minor logic errors
2. Major (but recoverable) graph construction errors. -2 marks E.g. instead of copying (u,v) to (u_i,v_i+1), replacing it instead. Or other extra edges.
3. Severe error: -3 marks "Create k copies of the graph", cannot be done because k is unknown.
4. Other severe error: -4 marks Connected a node u in the i^{th} layer to node u in the $(i+1)^{th}$ layer with a 0 weighted edge.
5. Other algorithms: -4 marks for either: BFS, DFS, or DAG SSSP
6. Bellman Ford: -3 marks
7. Broken retrieval for k: -4 marks for not correctly figuring out the value of k after constructing the graph.
8. Critical error: Assuming that the most expensive edge has to be made free. (Greedy approach). This gets 0 marks.

Question 5B

[2 marks]

$O(EV \log E)$

Question 6A Your algorithm that solves the main task. [8 marks]

```
function max_score(N, score_A, score_B, score_C):
# Initialise dp as a N x 3 x 2 array of negative infinities
dp[1][1][1] = score_A[1]
dp[1][2][1] = score_B[1]
dp[1][3][1] = score_C[1]

for i = 2 to N:
    dp[i][1][1] = score_A[i] + max(
        dp[i - 1][2][1], dp[i - 1][2][2],
        dp[i - 1][3][1], dp[i - 1][3][2],
    )

    dp[i][1][2] = score_A[i] + dp[i - 1][1][1]

    dp[i][2][1] = score_B[i] + max(
        dp[i - 1][1][1], dp[i - 1][1][2],
        dp[i - 1][3][1], dp[i - 1][3][2],
    )

    dp[i][2][2] = score_B[i] + dp[i - 1][2][1]

    dp[i][3][1] = score_C[i] + max(
        dp[i - 1][2][1], dp[i - 1][2][2],
        dp[i - 1][1][1], dp[i - 1][1][2],
    )

    dp[i][3][2] = score_C[i] + dp[i - 1][3][1]

return max(
    dp[N][1][1], dp[N][1][2],
    dp[N][2][1], dp[N][2][2],
    dp[N][3][1], dp[N][3][2],
)
```

1. 8 marks for correct answer
2. 6 marks for Minor mistakes in the recurrence
3. 6 marks for giving recurrence but no algo.
4. 6 marks for drawing a general graph without further explanation
5. 4 marks for other less efficient than optimal, eg. Bellman-Ford without stating Toposort
6. 1 marks for anything exponential in complexity
7. 0 marks if complete wrong (e.g. mst, knapsack blackboxed)
8. 0 marks if they only drew an example graph without further explanation, give 0.

Question 6B

[2 marks]

O(n)

1. If they dont get at least 1 mark in Q6a, it is considered as not giving an algorithm, so no marks for complexity.
2. 2 marks for O(n) correct answer or whatever time complexity their correct algorithm uses
3. 1 mark if they state correct answer but with undefined notation, eg. O(V), O(V + E). Not expressed in the correct parameter.
4. 0 marks for wrong answer

Question 7A Describe your strategy here [0 marks]