National University of Singapore

School of Computing

IT5003 - Data Structures and Algorithms
Final Assessment

(Semester 2 AY2022/23)

Time Allowed: 2 hours

INSTRUCTIONS TO CANDIDATES:

1. Do **NOT** open this assessment paper until you are told to do so.
2. This assessment paper contains THREE (3) sections.
   It comprises FOURTEEN (14) printed pages, including this page.
3. This is an **Open Book** but **not an Open Laptop** Assessment.
4. For Section A, use the OCR form provided (**use 2B pencil for the OCR form**).
   For Section B and C, answer **ALL** questions within the **boxed space** in the answer sheet.
   The answer sheet is at page 11-14.
   You will still need to hand over the entire paper as the MCQ section will not be archived.
   You can use either pen or pencil. Just make sure that you write **legibly**!
5. Important tips: Pace yourself! Do **not** spend too much time on one (hard) question.
   Read all the questions first! Some questions might be easier than they appear.
6. You can use **pseudo-code** in your answer but beware of penalty marks for **ambiguous answer**.
   You can use **standard, non-modified** algorithm discussed in class by just mentioning its name.
7. Please write your Student Number only. Do **not** write your name.

A	0							

This portion is for examiner's use only

Section	Maximum Marks	Your Marks	Grading Remarks
A	36		
B	19		
C	45		
Total	100		

A MCQs ($12 \times 3 = 36$ marks)

Select the **best unique** answer for each question.
Each correct answer worth 3 marks.

The MCQ section will not be archived to open up possibilities of reuse in the future.

[PAGE 3 IS NOT ARCHIVED]

[PAGE 4 IS NOT ARCHIVED]

[PAGE 5 IS NOT ARCHIVED]

[The upper side of PAGE 6 IS NOT ARCHIVED]

B Simpler Questions (19 marks)

B.1 Smallest Sub-List to be Sorted (9 marks)

You are given a list L of n items that is *not sorted* (that's it, at least there is one pair of items that are not in their correct positions). Your job is to determine the (0-based) bounds of the smallest sub-list of L to be sorted in order for the entire list L to be fully sorted (in non-decreasing order). In fact, this is another potential definition of a 'nearly sorted' list, i.e., if the size of the smallest sub-list of L to be sorted is small.

For example if the items are integers and $L = [2, 7, 8, 2, 5, 10, 15]$, then you should return $(1, 4)$ as the answer as index 0 (2) and indices 5..6 (10 and 15) are already in their correct position but integers in indices 1..4 are still in the wrong order.

For this question, the items can be of any data type, not necessarily integers, and for general solution, we can only determine the sorted order by *comparing* the items.

Propose an algorithm (and the associated data structure(s)) that is/are needed to solve this problem and analyze its time complexity in terms of n. To score up to 6 marks, your algorithm should be correct and runs in $O(n \log n)$. To score full (9) marks in this section, your algorithm should be correct and runs in $O(n)$. If you leave the boxed space blank, you will get automatic 1 mark. There is no other partial marks.

B.2 Quack ADT (10 marks)

In class, we have learned **Qu**eue ADT and St**ack** ADT. Now, let's merge them into **Quack** ADT. You can visualize Quack's elements listed from left to right such that three $O(1)$ operations are possible:

1. `QuackPush(x)`: add a new element x to the left end of the Quack,
2. `QuackPop()`: remove and then return the element on the left end of the Quack,
   If the Quack ADT is empty, return None.
3. `QuackPull()`: remove and then return the element on the right end of the Quack.
   If the Quack ADT is empty, return None.

For example, let the current Quack Q contains 5 integers $[4, 7, 1, 8, 9]$ and we perform:

1. `QuackPush(5)`, then Q changes to $[5, 4, 7, 1, 8, 9]$,
2. `QuackPop()`, then Q changes back to $[4, 7, 1, 8, 9]$ and it returns 5,
3. `QuackPull()`, then Q changes to $[4, 7, 1, 8]$ and it returns 9.

B.2.1 Implement Quack ADT with Python `deque` (4 marks)

After going through IT5003, you should immediately notice that Python `deque` data structure is more than enough to implement a Quack ADT efficiently. In this section, your job is to complete the three Quack ADT operations exclusively using Python `deque` operations and all operations must run in $O(1)$ (1 mark each). As an example, the operation `left(self)` has been implemented for you.

B.2.2 Implement Stack ADT with Quack ADT operations (3 marks)

In class, you have also learned about the standard Stack ADT. Now, you have to implement three Stack ADT operations (top, push, and pop), but this time using Quack ADT implementation (`class Quack`) that you have created in the previous Section B.2.1. Note that any other answer without using Quack ADT operations will be marked as wrong answer.

B.2.3 Implement Queue ADT with Quack ADT operations (3 marks)

In class, you have also learned about the standard Queue ADT. Now, you have to implement three Queue ADT operations (front, enqueue, and dequeue), but this time using Quack ADT implementation (`class Quack`) that you have created in the previous Section B.2.1. Note that any other answer without using Quack ADT operations will be marked as wrong answer.

C Applications (15+15+15 = 45 marks)

C.1 Generate Special Integers (15 marks)

Certain integers have special properties, e.g., even integers, odd integers, prime integers, etc. This time, we want to *generate* the list of the first n integers that has form $2^i \cdot 3^j \cdot 5^k$ for non-negative integers i, j, k.

The first 20 of these special integers are: $[2^0 \cdot 3^0 \cdot 5^0 = 1, 2^1 \cdot 3^0 \cdot 5^0 = 2, 2^0 \cdot 3^1 \cdot 5^0 = 3,$
$2^2 \cdot 3^0 \cdot 5^0 = 4, 2^0 \cdot 3^0 \cdot 5^1 = 5, 6, 8, 9, 10, 12, 15, 16, 18, 20, 24, 25, 27, 30, 32, 36]$.

Notice that $2^0 \cdot 3^0 \cdot 5^0 = 1$ is the first such special integer, $2^0 \cdot 3^0 \cdot 5^1 = 5$ is not the fourth such special integer as $2^2 \cdot 3^0 \cdot 5^0 = 4$ appears earlier, and 14 is not a special integer as it has 7 as one of its prime factor (we only take integers that can be expressed as $2^i \cdot 3^j \cdot 5^k$).

C.1.1 One Manual Test Case (2 marks)

If you have understood this question, what are the next 5 special integers (the 21st to 25th)?

C.1.2 Solve This Problem (13 marks)

Propose an algorithm (and the associated data structure(s)) that is/are needed to solve this problem and analyze its time complexity in terms of n. To score full (13) marks in this section, your algorithm should be correct and runs in $O(n \log n)$ (or faster, i.e., the largest test case can go up to $1 \leq n \leq 200\,000$). Hint: Use Priority Queue ADT. To score up to 6 marks, your algorithm should at least be correct (even though it may be slow). If you leave the boxed space blank, you will get automatic 1 mark. There is no other partial marks.

C.2 Cutting Wall (15 marks)

A wall in your house consists of several rows (numbered from 1 to r from top to bottom) of bricks of various integer lengths and uniform height. You want to find a vertical line going from the top to the bottom of the wall that cuts through the fewest number of bricks as you want to install a small electrical cable along that cut. Note that if the line goes through the edge between two bricks (in practice, there is a small gap there), this does not count as a cut — in fact, we want to maximize doing this. You can assume that you cannot perform the cut on the extreme leftmost or extreme rightmost side of your wall.

The input starts with a line that contains an integer r, the number of rows of bricks in your wall, and then r lines. Each of the next r lines describe the r rows of bricks on your wall. Each row contains one or more integers that represent the lengths of the bricks in that row. The sum of all brick lengths in each row is identical – that's it, your wall is of size $r \times c$, where c is the sum of length of bricks on any row ($1 \leq c \leq 10^9$). It is guaranteed that there are at most k bricks in any row and for the purpose of this problem $1 \leq k \leq 100$ (and is usually much smaller than c). For example, if the input is as follows:

```
6
3 5 1 1
2 3 3 2
5 5
4 4 2
1 3 3 3
1 1 6 2
```

Then, your wall would then look like Figure 1:

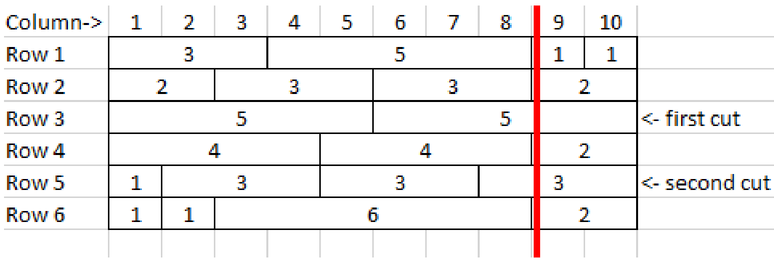

Figure 1: An Example Wall

The optimal answer is 2, by cutting at the (right side of the) 8th column (only bricks in the third and fifth rows from the top have to be cut), as shown above.

C.2.1 One Manual Test Case (2 marks)

If you have understood this question, what is the answer for this test case?
Please give a short explanation as shown above (draw your explanation as with Figure 1).

```
5
1 5 7 1 999999986
2 11 3 999999984
5 5 2 6 999999982
6 5 9 999999980
3 3 3 3 2 8 999999978
```

C.2.2 Solve This Problem (13 marks)

Propose an algorithm (and the associated data structure(s)) that is/are needed to solve this problem and analyze its time complexity in terms of r, c, and/or k. To score full (13) marks in this section, your algorithm should be correct and runs in $O(r \times k)$ (or faster, i.e., the largest test case can go up to $1 \leq r \leq 200\,000$ and $1 \leq k \leq 100$). Hint: Use Hash Table ADT. To score up to 6 marks, your algorithm should at least be correct (even though it may be slow). If you leave the boxed space blank, you will get automatic 1 mark. There is no other partial marks.

C.3 Vitamin Sea (15 marks)

There are lots of countries in the world which are landlocked. That is, they do not have any beach that touches a sea. However, but by going through one other country, a sea can be reached. For example, a person in Luxembourg who is in need of "Vitamin Sea" can reach a sea by passing through the neighbor of Luxembourg, e.g., Belgium.

Your task is to determine how landlocked each country is on a given map. We say that a country is not landlocked (recorded as 0) if it touches water in any adjacent cell in either a horizontal, vertical, or diagonal direction (that's it, a total of 8 directions). If a country is landlocked, you must calculate the minimum number of international borders that one must cross in order to travel from the country to reach any sea. Each step of such a journey must be to a cell that is adjacent in either a horizontal, vertical, or diagonal direction. Crossing an international border is defined as taking a step from a cell in one country to an adjacent cell in a different country.

Note that countries may not be connected to themselves (as in a country formed of islands). In this case, the landlocked value for the country is *the minimal* of each connected region of the country.

You are given two integers r and c in the first line ($1 \leq r, c \leq 1000$), followed by r lines of c characters. Each character is either an alphabet ['A'..'Z'] (one character 'country' name, so there are at most 26 countries in any test case) or '$\sim$' (tilde, that represent the water in the ocean). To simplify this problem, you just need to output which country is the most landlocked and the number of international border(s) to be crossed to do reach a sea from that country. If there are more than one such countries, pick the lowest country character as the output.

For example, if you are given the following test case:

```
9 10
~~~~CBBBC~
~~~~CBYBC~
~~~~CBBBC~
~~~~CCCCC~
~~~~~~~~~~
~EFGHI~~~~
~DAAAJ~~~~
~DAXAK~~~~
~DAAAL~~~~
```

Then you have to output “X 2” as citizens of both ‘X‘ and ‘Y‘ need to cross at least 2 international border in order to reach any ‘∼’ (sea). Since ‘X’ < ‘Y’, we output “X 2”.

C.3.1 One Manual Test Case (2 marks)

If you have understood this question, what is the answer for this test case?
Please give a short explanation as shown above.

```
9 9
~~~~~~~~~
~AAAAAAA~
~ABBBBBA~
~ABAAABA~
~ABACABA~
~ABAAABA~
~ABBBBBA~
~AAAAAAA~
~~~~~~~~C
```

C.3.2 Solve This Problem (13 marks)

Propose an algorithm (and the associated data structure(s)) that is/are needed to solve this problem and analyze its time complexity in terms of r and c. To score full (13) marks in this section, your algorithm should be correct and runs in $O(r \times c)$ (or faster, i.e., the largest test case can go up to $1 \leq r, c \leq 1000$). To score up to 6 marks, your algorithm should at least be correct (even though it may be slow). If you leave the boxed space blank, you will get automatic 1 mark. There is no other partial marks.

The Answer Sheet

*My section B.1 answer (this is a starred question, 1 mark if left blank):

My section B.2.1 answer:

```
from collections import deque
class Quack:
    def __init__(self): # for this section, you MUST use Python deque
        self.d = deque()
    def left(self): # shown as an example
        if not self.d: return None
        return self.d[0]
    def right(self): # returns the rightmost element in O(1) (1 mark)



    def QuackPush(self, x): # implement O(1) QuackPush (1 mark)

        # no return value is needed for QuackPush
    def QuackPop(self): # implement O(1) QuackPop (1 mark)



    def QuackPull(self):  # implement QuackPull (1 mark)
```

My section B.2.2 answer:

```
class Stack(Quack): # for this section, you MUST use Quack ADT
    def top(self): # implement Stack top operation using Quack operation


    def push(self, x): # implement Stack push operation using Quack operation

        # no return value is needed for push
    def pop(self): # implement Stack pop operation using Quack operation
```

My section B.2.3 answer:

```
class Queue(Quack): # for this section, you MUST use Quack ADT
    def front(self): # implement Queue front operation using Quack operation


    def enqueue(self, x): # implement Queue enqueue operation using Quack operation

        # no return value is needed for enqueue
    def dequeue(self): # implement Queue dequeue operation using Quack operation
```

My section C.1.1 answer (the next 5 special integers):

*My section C.1.2 answer (this is a starred question, 1 mark if left blank):

My section C.2.1 answer (the minimum number of cut(s) and a drawing sketch):

*My section C.2.2 answer (this is a starred question, 1 mark if left blank):

My section C.3.1 answer (a character between ['A'..'Z'] and how landlocked that country is):

*My section C.3.2 answer (this is a starred question, 1 mark if left blank):

– END OF PAPER; All the Best –