National University of Singapore

School of Computing

CS3230 - Design and Analysis of Algorithms
Final Assessment

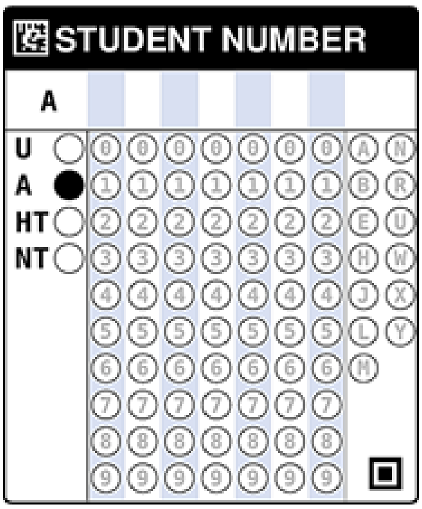
(Semester 2 AY2023/24)

Total Marks: 80 Time Allowed: 120 minutes

INSTRUCTIONS TO CANDIDATES:

1. Do **NOT** open this assessment paper until you are told to do so.
2. This assessment paper contains TWO (2) sections. It comprises THIRTEEN (13) printed pages.
3. This is an **Open Book** Assessment.
4. For Section A, use the bubbles on page 2 (use 2B pencil).
5. For Section B, answer **ALL** questions within the **boxed space**.
   If you leave the box blank, you will get 0.5 mark (**ONLY** for essay questions worth ≥ 2).
   However, if you write at least a single character and it is totally wrong, you will get a 0 mark.
   You can use either a pen or a pencil. Just make sure that you write **legibly**!
6. 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.
7. You can assume that all **logarithms are in base** 2.

Write your Student Number in the box below using **(2B) pencil**:



A Multiple Choice Questions ($16 \times 2.5 = 40$ marks)

Select the **best unique** answer for each question. Each correct answer is worth 2.5 marks.
Write your MCQ answers in the special MCQ answer box below for automatic grading.
We do not manually check your answer.

Write your MCQ answers in the answer box below using (2B) pencil:

No	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
B	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
C	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
D	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
E	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

Page 3 is redacted.

Page 4 is redacted.

Page 5 is redacted.

Page 6 is redacted.

B Essay Questions (40 marks)

B.1 Average-case Comparisons ($2 + 3 + 5 = 10$ marks)

Consider the following algorithm to find the two largest numbers from an n-length array $A[1..n]$ of n distinct numbers.

Input: Given n distinct numbers, $A[1..n]$ contains a random permutation of them.
Begin

1. If $A[1] > A[2]$,
   Then let $a = A[1]$ and $b = A[2]$,
   Else let $a = A[2]$ and $b = A[1]$

   Endif
2. For $i = 3$ to n do:
   2.1. If $b < A[i]$ Then
      2.2. If $a < A[i]$,
         Then let $b = a$ and $a = A[i]$
         Else let $b = A[i]$
      Endif
   Endif

   //**Comment:** a and b store the two largest elements of the subarray $A[1..i]$.
   End For
3. Output a, b.

End

a). (2 marks) For a fixed i, in the For loop iteration with index i, at step 2.1: what is the probability that $A[i] > b$?

(Your answer can be in terms of i. **Hint:** Observe the comment in the pseudocode carefully.).

b). (3 marks) What is the expected number of comparisons made by the above algorithm? (Here, we count only the comparisons made between a, b and the array elements in steps numbered 1, 2.1, and 2.2 and not any comparisons done in the control statement of the For statement).

Give the bound in the form $n + O(f(n))$, where $f(n)$ is as tight as possible.

c). (5 marks) Provide justification for your answers to parts (a) and (b).

B.2 Maximizing Profit ($1 + 3 + 4 + 2 = 10$ marks)

You are a fisherman. Very early this morning, you caught n fishes. These n fishes weigh $w_1, w_2, ..., w_n$ kilograms respectively (you can assume that all n fish weights are Integers between 1 to 1024 kilograms, not necessarily distinct).

In the fish market, there are m fish sellers described as a sequence of m pairs $(x_1, p_1), (x_2, p_2), ..., (x_m, p_m)$. A fish seller j with (x_j, p_j) pair means that this fish seller j is willing to buy x_j number of fish (x_j is also an Integer, $1 \leq x_j \leq n$ (notice x_j could be as large as n), and not necessarily distinct) at p_j SGD per kilogram (p_j is also an Integer, $1 \leq p_j \leq 1024$, and not necessarily distinct).

Design a greedy algorithm to compute the maximum profit (in SGD) that you can get by selling (some, if not all) your n fishes to (some of) these m fish sellers optimally. Note that there are partial marks if your greedy algorithm is correct only when $x_j = 1$ for all m fish sellers (e.g., see part a).1)).

For example, if you caught $n = 3$ fishes with weights $4, 7, 5$ and there are $m = 2$ fish sellers $(1, 10), (3, 9)$, then the optimal strategy is to sell your second fish with weight 7 kilogram to the first fish seller who only wants to buy 1 fish that day at price 10 SGD per kilogram (you get $7 \times 10 = 70$ SGD) and then sell your two other fishes to the second fish seller who can buy up to 3 fishes but you only have two fishes left (you get $4 \times 9 + 5 \times 9 = 81$ SGD). So your total profit is $70 + 81 = 151$ SGD.

a). ($2 \times 0.5 = 1$ mark) **Judge your understanding:**
Just write two output Integers, one for each test case below.
1). $n = 3$, weights $9, 4, 5$, $m = 4$, fish sellers $(1, 2), (1, 1), (1, 6), (1, 3)$
2). $n = 10$, weights $10, 8, 4, 28, 19, 2, 7, 5, 9, 1$, $m = 3$, fish sellers $(2, 4), (1, 5), (4, 3)$

b). (3 marks) Describe the **optimal sub-structure** of this problem and **prove** its correctness.

c). (4 marks) Describe the **greedy choice** that works for this problem and **prove** its correctness. PS: If your greedy choice is correct but your proof is not, you will still get partial marks.

d). (2 marks) Combine the **optimal sub-structure** (in B.2.b) and the **greedy choice** (in B.2.c) to design an algorithm that always outputs an optimal solution. Analyze the time complexity of your greedy solution in terms of n and m. Is it polynomial, pseudo-polynomial, or exponential (choose the best option)?

B.3 Priority Queue (10 marks)

Recall that the priority queue data structure supports the following operations:

- `add(x)`: inserts x into the queue; $O(\log n)$ (worst-case) time complexity (n refers to the number of elements in the queue)
- `top()`: returns the largest element in the queue; $O(1)$ (worst-case) time complexity
- `pop()`: removes the largest element from the queue; $O(\log n)$ (worst-case) time complexity (n refers to the number of elements in the queue)

We now wish to support another operation with the following specifications:

- `remove_larger_than(x)`: removes all items larger than x from the queue

`remove_larger_than(x)` works by repeatedly checking whether `top()` is larger than x, and if so, calls `pop()` until `top()` is $\leq x$. More precisely, you may assume the following implementation:

```
remove_larger_than(x):
    while (priority queue is not empty and top() > x):
        pop()
```

Given an initially empty priority queue, prove that any sequence of n (of the above four types of) operations takes at most $O(n \log n)$ time. You can use any of the three amortized analysis techniques.

B.4 Respectful Coloring is NP-complete! ($1 + 3 + 6 = 10$ marks)

Suppose there are k persons $P_1, P_2, \cdots, P_k$ and n balls $B_1, B_2, \cdots, B_n$. Each person provides his/her preferred coloring of n balls from the set of five colors $\{red, blue, green, yellow, pink\}$. So essentially, each person P_i provides a sequence of colors $c_{i1}, c_{i2}, \cdots, c_{in}$, where $c_{ij} \in \{red, blue, green, yellow, pink\}$ denotes P_i's preferred color for the ball B_j.

Now, there is a painter whose job is to finally color all these n balls. Unfortunately, in his/her color palette, only two colors, *red* and *blue*, are available. The painter would like to color balls using colors available in his/her palette while respecting the color preference of every person for at least one ball. More specifically, we call a $red - blue$ coloring of balls a *respectful coloring* if for each person P_i there exists a ball B_j such that the coloring of B_j is the same as P_i's preferred color for B_j (i.e., c_{ij}). If such a respectful final coloring exists, the painter would like to use that (if multiple red-blue respectful colorings exist, choose one arbitrarily) to color the balls.

Red-Blue Respectful Coloring problem: Given color-preferences of k persons on n balls (as described in the first paragraph), the problem is to decide whether there exists a $red - blue$ respectful coloring or not.

For example, suppose there are 3 persons and 4 balls. The color preferences of three persons are as follows: $P_1 : green, blue, blue, yellow$, $P_2 : red, red, pink, green$, $P_3 : pink, yellow, blue, blue$. Then the answer should be YES since $red, blue, blue, blue$ is a valid $red - blue$ respectful coloring.

a). (1 mark) **Judge your understanding:** There are 4 persons and 3 balls. The color preferences of four persons are as follows: $P_1 : red, blue, yellow$, $P_2 : pink, red, green$, $P_3 : yellow, pink, blue$, $P_4 : blue, pink, red$. Does there exist a $red - blue$ respectful coloring? (Tick one of the following options; if your answer is YES, then also provide a $red - blue$ respectful coloring as a sequence of three colors.)

   i). YES. Your $red - blue$ respectful coloring: ____________________________

   ii). NO.

b). (3 marks) Show that the Red-Blue Respectful Coloring problem is in `NP`.

c). (6 marks) Show that the Red-Blue Respectful Coloring problem is NP-hard.

(You may show a reduction from any of the NP-complete problems introduced in the lectures/ tutorials/ assignments/ practice set.)

Hint: Try a reduction from CNF-SAT or 3-SAT.

– END OF PAPER; All the Best –