National University of Singapore

School of Computing

CS3230 - Design and Analysis of Algorithms

Midterm Assessment

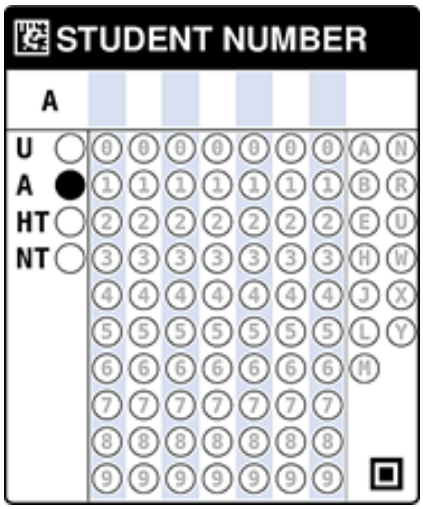
(Semester 2 AY2023/24)

Time Allowed: 80 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 EIGHT (8) printed pages.
3. This is an **Open Book** Assessment.
4. For Section A, use the bubbles on page 4 (use 2B pencil).
5. For Section B, answer **ALL** questions within the **boxed space**.
   If you leave the box blank, you will get an automatic 0.5 mark (**NOT** for Bonus question).
   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 ($10 \times 1.5 = 15$ marks)

Select the **best unique** answer for each question. Each correct answer is worth 1.5 marks.
Write your MCQ answers in the special MCQ answer box on page 4 for automatic grading. We do not manually check your answer.

1. Which of the following statements is **TRUE**?

   a). $n! = \Omega(n^n)$

   b). $2^n = O((\log n)^{\log n})$

   c). $n/\log n = \Omega(n^{0.999})$

   d). $n^{\log n} = O(n^{2000})$

   e). None of the above

2. $(2n)!$ is in

   a). $O((n!) \cdot (n!))$

   b). $O((n!) \cdot (2^n))$

   c). $O((n!) \cdot (n!) \cdot (n!))$

   d). $\Omega((n!) \cdot (n!) \cdot (n!))$

   e). None of the above

3. While computing the time complexity of a recursive algorithm, you get the following recurrence relation: $T(n) = 49T(n/7) + n^{7/2}$. Then which one of the following is **TRUE**?

   a). $T(n) = \Theta(n^{7/2} \log n)$

   b). $T(n) = \Theta(n^{7/2})$

   c). $T(n) = o(n^2)$

   d). $T(n) = \Theta(n^2)$

   e). $T(n) = \Theta(n^2 \log n)$

4. Suppose $T(1) = 1$, $T(n) = T(n-1) + 3n + 7$. Then which one of the following is **TRUE**?

   a). $T(n)$ is in $\Theta(n^2)$

   b). $T(n)$ is in $\Theta(n^3)$

   c). $T(n)$ is in $\Theta(n)$

   d). $T(n)$ is in $\Omega(2^n)$

   e). None of the above

5. Which of the following statements is **TRUE**?

   a). Longest Common Subsequence of any two arbitrary input sequences is always unique

   b). Any algorithm to compute n-th Fibonacci number (mod n) must take $\Omega(n)$ time

   c). The running time of a dynamic programming algorithm is always upper bounded by the number of distinct subproblems

   d). Any comparison-based algorithm to search an integer in an n-length sorted integer array must take $\Omega(\log n)$ comparisons

   e). None of the above

6. Consider the recurrence relation: $T(n) = T(0.9n) + n$. (Assume, $T(n) \leq 99$, for $n \leq 17$.) What is the height of the corresponding recursion tree?

   a). $\Theta(1)$

   b). $\Theta(\log \log n)$

   c). $\Theta(\log n)$

   d). $\Theta(n^{0.1})$

   e). $\Theta(n)$

7. Consider the deterministic algorithm for quick sort. Which of the following statements is **TRUE**?

   a). For all large enough n, for every possible input of n numbers, quick sort algorithm performs at least $0.001n^2$ comparisons

   b). For all constant $c > 0$, for all large enough n, for all possible input of n numbers, quick sort algorithm performs at least cn^2 comparisons

   c). For some constant $c > 0$, for all large enough n, for all possible input of n numbers, quick sort algorithm performs at most cn^2 comparisons

   d). For some constant $c > 0$, for all large enough n, for all possible input of$n$ numbers, quick sort algorithm performs at most $cn \log n$ comparisons

   e). None of the above

8. Suppose you are given as input an (undirected) graph $G = (V, E)$ with n nodes and m edges, where each vertex has a unique label from $\{1, 2, \cdots, n\}$. Next, consider the following randomized procedure: Select a vertex $v \in V$ uniformly at random and then return the label of v as output. Let X be the random variable denoting the output of this randomized procedure. What is the expected value of X?

   a). $m/2$

   b). $2m/n$

   c). $n/2$

d). $1/n$

e). $(n+1)/2$

9. For $n > 0$, Strassen's multiplication algorithm for multiplying two $n \times n$ integer matrices does

   a). $O(n^{\log_2 7})$ number of integer multiplications, $\Omega(n^3)$ number of integer additions

   b). $\Omega(n^3)$ number of integer multiplications, $O(n^{\log_2 7})$ number of integer additions

   c). $O(n^{\log_2 7})$ number of integer multiplications, $O(n^{\log_2 7})$ number of integer additions

   d). $\Theta(n^3)$ number of integer multiplications, $\Theta(n^{\log_2 7})$ number of integer additions

   e). None of the above

10. Suppose $Pr(X)$ denotes the probability of event X happening. Suppose A and B are two events. Which of the following statements is always **TRUE**:

    a). $Pr(A \text{ or } B) > max(Pr(A), Pr(B))$

    b). $Pr(A \text{ or } B) \geq Pr(A) + Pr(B)$

    c). $Pr(A \text{ or } B) \leq Pr(A) + Pr(B)$

    d). $Pr(A \text{ and } B) = Pr(A) \cdot Pr(B)$

    e). None of the above

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

No	1	2	3	4	5	6	7	8	9	10
A	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯
B	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯
C	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯
D	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯
E	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯

B Essay Questions (25 marks)

B.1 One Defective Ball ($5 + 4 = 9$ marks, plus 2 bonus marks)

Recall the question from Tutorial 4, where you had to find one heavier ball from a group of 243 balls. This question generalizes the problem. Suppose now we have n balls, and one of them is defective (either heavier **or lighter** than other $n - 1$ balls). Assume $n \geq 10$.

You have a balance, which allows you to check when given two sets A, B of balls (placed on either side of the balance respectively), whether:

1. A and B weigh equally, or
2. A is heavier than B, or
3. A is lighter than B.

Using the above balance, you need to devise an algorithm to detect which ball is defective and also whether it is heavier or lighter than other balls. As each weighing is charged, your algorithm needs to do it in as small a number of weighings as possible (in the worst case, in terms of n). For this problem, you need to give an optimal answer rather than in Θ notation. Your marks depend on whether your algorithm correctly finds the defective ball and how close your answer is to the optimal, though you are only required to prove upper bound (not lower bound).

Though you are allowed to use any method to solve the above question, the following parts guide you toward the answer in a structured/simpler way. (**Note:** If you decide not to follow the following guided path, you may use all the boxes for your own method; just specify at the top that you are not following the guided path.)

Remark: For the questions below, you should clearly specify your algorithm for how you are selecting the balls to be put on balance (and which side of the balance you place them on). However, you are not required to give any pseudo code.

a). (5 marks) Suppose you are given m balls, each colored RED or `green`. Among these balls, one is defective. Also you are also given one additional non-defective ball. Suppose you already know that:

- If the defective ball is RED, then it must be heavier than the other balls, and
- If the defective ball is `green`, then it must be lighter than the other balls.

What is the largest value of m, such that you can determine the defective ball among the m colored balls, in the worst case, using at most i weighings?

Hint: In each round, think about how many (and which) balls you want to put on the balance (and on which side of the balance).

b). (4 marks) Suppose you have n balls, where you have no other information (i.e., they are *not colored* as in part a).). Among these balls one is defective. You still have that extra one additional non-defective ball. Now, what is the largest value of n, such that you can determine the defective ball among the n balls (as well as whether it is heavier **or lighter** than the other balls), in the worst case, using at most i weighings?

Hint: In each round, think how many balls you want to put on the balance (and on which side).

Let $T(i)$ denote the largest number of balls from which you can find the defective ball using i weighings (in the worst case). Then try to express $T(i)$ in terms of $T(i-1)$ and result from part a), and then solve for $T(i)$.

c). **(Bonus 2 marks)** Now modify your algorithm and analysis for part b). when you are *not given* the additional non-defective ball (solving this variant will then solve the original question), to determine the number of weighings needed to find the defective ball among the n balls (where $n \geq 10$).

B.2 Average Number of Inversions ($3 + 5 = 8$ marks)

Consider a sequence $a_1, a_2, \cdots, a_n$ of n distinct integers, where each $a_i \in \{1, 2, \cdots, n\}$. We call a pair (a_i, a_j) *inverted* if $i < j$ but $a_i > a_j$. For instance, the sequence $2, 3, 8, 5, 4$ contains three inverted pairs $(8, 5), (8, 4)$ and $(5, 4)$.

Let π be a permutation of $\{1, 2, \cdots, n\}$ chosen uniformly at random from the set of all permutations of $\{1, 2, \cdots, n\}$. Visualize this permutation π as a sequence $a_1, a_2, \cdots, a_n$ of n distinct integers.

a). Fix two integers $i, j \in \{1, 2, \cdots, n\}$. What is the probability that (a_i, a_j) forms an inverted pair? (Provide an answer with a proper explanation.)

b). Suppose you are given a sorting algorithm that, given as input a permutation π with k inverted pairs, runs in $\Theta(k)$ time. Analyze the average-case complexity of this sorting algorithm. More specifically, what is the expected running time of this sorting algorithm when provided with as

input a permutation of $\{1, 2, \cdots, n\}$ chosen uniformly at random from the set of all permutations of $\{1, 2, \cdots, n\}$? (Express the expected running time as a function of n using $\Theta(\cdot)$ notation.)

Note: If you cannot answer part (a), you may still answer this question (without any further penalty) assuming the answer to part (a) is p and then express the expected running time as a function of n, p using $\Theta(\cdot)$ notation.

B.3 Stock Market ($5 + 2 + 1 = 8$ marks)

You have monitored the stock price of a certain popular company that you really like for the last N days: $P_1, P_2, \cdots, P_N$ (every P_i satisfies $1 \leq P_i \leq 777$ SGD). The stockbroker charges you a fixed rate of C ($0 \leq C \leq 30$) for every stock purchase (there is no cost/fee when you sell your stock to the stockbroker). Today is day $N + 1$; you wondered how much money you would have made if you had invested optimally during the previous day $[1..N]$. You are very risk averse so that at any given time in day $[1..N]$, you will never have more than one stock of the company. Note that there is a possibility that the answer is 0 SGD, i.e., it is best not to invest at all. Also, note that at the end, you need not hold any stock.

Example 1: If $N = 6$, $C = 10$ and $P = \{\mathbf{100}, 120, \underline{130}, 80, 50, 40\}$, the optimal strategy is to **buy one stock at day** 1**, paying** 100+10 = 110 **SGD**, wait, and on day 3, <u>sell that one stock at 130 SGD</u>. This way, you earn a profit of $130 - 110 = 20$ SGD. Do not do anything else on day $4, 5, 6$ (as the stock price of that company keeps declining).

Example 2: If $N = 6$, $C = 10$ and $P = \{100, 99, 99, 98, 97, 95\}$, the optimal strategy is to not do anything in these 6 days as the stock prices are non-increasing.

a). Describe the recurrence of this problem: the base case(s) and the general/recursive case(s)!

b). 1). What is the running time of the recurrence above if you implement your recursion verbatim?
2). What is the *faster* running time if you implement the recurrence using Top-Down DP (recursion *with memoization*) or Bottom-Up DP? (Express **both** running times using $\Theta(\cdot)$ notation.)

c). Using the DP solution that you have provided above, be a human computer and compute what is the optimal answer (just give a single integer) if you are given $N = 20$, $C = 30$, and $P = \{10, 80, 20, 40, 30, 50, 40, 60, 50, 70, 60, 10, 200, 199, 198, 197, 196, 195, 194, 193\}$.

– END OF PAPER; All the Best –