NATIONAL UNIVERSITY OF SINGAPORE
SCHOOL OF COMPUTING

Final Assessment for CS3230

April 26, 2022 Time Allowed: 80 minutes (5:15–6:35pm)

INSTRUCTIONS:

- This paper consists of **FOUR** questions for a total of 60 points. Please read instructions to each question carefully.
- This is an **OPEN BOOK/NOTES** examination.
- Do **NOT** use the internet for help while taking the examination. Any evidence otherwise will result in **severe penalties**.
- Do **NOT** seek the help of others during the examination. Any evidence otherwise will result in **severe penalties**.
- You have **15** minutes (**6:35pm** to **6:50pm**) to upload the scan of your hand-written solutions to LumiNUS at `Final/Submissions`.
- You need to generate a **separate PDF file** for each of the four questions on the assessment. The four files must be named as `Axxxxxxxy_Q1.pdf`, `Axxxxxxxy_Q2.pdf`, `Axxxxxxxy_Q3.pdf` and `Axxxxxxxy_Q4.pdf` where `Axxxxxxxy` is your student number.
- Submit a file even if you do not attempt a particular question. It is **your own responsibility** to ensure that the PDFs correctly contain your solutions.
- Recall that lg denotes the logarithm with base 2.
- For "Design and analyze an algorithm" questions, you must analyze both the **correctness** and the **running time** of your proposed algorithm.

QUESTIONS:

1. (12 points) For each question in this part, write "True", "False", or "Don't know". Each correct answer is worth 2 points; the answer "Don't know" guarantees 1 point. No need to give justifications.

   (a) $n \lg n + n^2 = \Theta(n^2)$

   (b) $5^{\lg \lg n} = O((\lg n)^2)$

   (c) $n^n = \omega((2n)!)$

   (d) According to our current state of knowledge, there may exist a problem that is in P but not in NP.

   (e) If there is a polynomial-time algorithm for 3-SAT, then there is certainly also a polynomial-time algorithm for Set Cover.

   (f) There is currently no known polynomial-time algorithm for Knapsack.

2. (15 points) For each of the five questions in this part, provide the requested answer in simplest form using the Θ-notation. No need to give justifications.

   For questions (a)–(d), solve the following recurrences by providing tight asymptotic bounds. You can ignore the fact that $n/2$, $n/3$, $n/5$, $n/7$, and $\sqrt{n}$ may not be integers.

   (a) (3 points) $T(n) = 4T(n/2) + n^2 \lg n$

   (b) (3 points) $T(n) = T(n/3) + T(n/5) + T(n/7) + n$

   (c) (3 points) $T(n) = T(\sqrt{n}) + 3230$

   (d) (3 points) $T(n) = T(n-3) + 3n$

   For question (e), suppose you perform a sequence of n operations. The i-th operation costs i^2 if i is a power of 2, and costs 3 otherwise.

   (e) (3 points) What is the (asymptotic) amortized cost per operation?

3. (10 points) You have n handwritten documents that you want to be typed up, and n friends whom you can hire to help. For $i \in \{1, 2, \ldots, n\}$, document i contains w_i words, and friend i can type at a rate of r_i words per minute. Assume that w_i and r_i are positive integers for all i. You can assign each friend exactly one document to type up (so each document must be assigned to exactly one friend), and must pay each friend a number of dollars exactly equal to the number of minutes that this friend takes to type up the assigned document (note that this number is not necessarily an integer).

   Naturally, you want to minimize your total payment. Design an analyze an algorithm that computes the smallest total amount of money that you have to pay. You should try to minimize the running time of your algorithm (as a function of n), and must state this running time using the O-notation.

4. (23 points) There are n sheets of paper, each with a front side (clearly marked "Front") and a back side (clearly marked "Back"). For $i \in \{1, 2, \ldots, n\}$, sheet i has a nonnegative integer a_i written on its front side and a nonnegative integer b_i written on its back side. Let $A = \sum_{i=1}^{n} a_i$ and $B = \sum_{i=1}^{n} b_i$.

   A set of sheets $S \subseteq \{1, 2, \ldots, n\}$ is called *fabulous* if $\sum_{i \in S} a_i \geq A/2$ and $\sum_{i \in S} b_i \geq B/2$. (In words, a fabulous set of sheets has the property that the sum of the numbers on the front side of these sheets is at least half of the sum of numbers on the front side of all n sheets, and similarly for the back side.)

   (a) (10 points) In the problem FabulousHalf, you want to decide whether there exists a fabulous set of size exactly $n/2$. Prove that FabulousHalf is NP-complete.

   (You may assume, without proof, that the following problem called PartitionEqual is NP-complete: Given nonnegative integers $x_1, \ldots, x_n$, decide whether they can be partitioned into two parts with equal sum in such a way that each part consists of exactly $n/2$ of the n integers.)

   (b) (10 points) Design and analyze an algorithm that computes the minimum size of a fabulous set. Your algorithm should run in time polynomial in n, A, and B. You must state the running time using the O-notation.

   (c) (3 points) Upon hearing what you showed in parts (a) and (b), your friend Bob tells you the following: "In order to solve FabulousHalf in polynomial time, you can use your algorithm from part (b) to compute the minimum size of a fabulous set. The answer to FabulousHalf is 'Yes' if and only if this minimum size is at most $n/2$. Since you have shown in part (a) that FabulousHalf is NP-complete, this means you have shown that P = NP."

   Explain why Bob's reasoning is wrong.

   (**Note:** Even if you did not actually solve parts (a) and/or (b), you can still attempt part (c) by assuming the statements from those two parts.)