COSC 3P03: Design and Analysis of Algorithms
Final, Dec. 14, 2018

Note: your answers must be precise, clear, legible, and as short as possible. Use only the space provided. In cases where proof/explanation/details are required, answers alone (without justification) will get no credit. Also, you may use any algorithms that are covered in classes by simply referencing them without having to write down details. For example, you may simply say "Find the 5th smallest element using the linear-time selection algorithm", etc.

Also: one cheat sheet is allowed. No computer/calculator is allowed.

Name: Student Number:

1. (5) Professor Amongus has shown that a decision problem A is polynomially reducible to an NP-complete problem B. And after many pages of dense mathematics, he has also proven that A can be solved in polynomial time deterministically. Has he just proven that $P = NP$? Why or why not?

2. (10) Given a sequence S of n distinct integers x_1, x_2, ..., x_n, our job is to find a longest increasing subsequence. For example, if the entries of S are 11, 17, 5, 8, 6, 4, 7, 12, 3, then one longest increasing subsequence is 5, 6, 7, 12. We discussed this problem in class when we were talking about dynamic programming and polynomial reductions. Solve the problem by dynamic programming by defining a proper length function and giving its recurrence.

3. (15)
Use the dynamic programming method to fill the table for finding the optimal way to multiply four matrices $M_1 \times M_2 \times M_3 \times M_4$ with dimensions 1×5, 5×5, 5×1, and 1×1. Note that for matrix M_i, its dimensions are $r_{i-1} \times r_i$. You need to

1. write down the recurrence for m_{ij} as defined in class;
2. fill the table for computing m_{ij}'s;
3. give an order in which to multiply the matrices optimally by putting parentheses around these matrices as done in Assignment 3.

4. (25) Let's consider the problem of finding the counterfeit coin from the Test 2 again: Suppose that we have 3 identical looking coins and only one of the coins is heavier than others. Suppose further that you have one balance scale and are allowed one weighing. We know how to find the counterfeit with one weighing as follows: Weigh any two. If one of them is heavier, that is the one. Else, the third one is the one.

Now assume that we have n coins where $n = 3^k$ for some integer $k > 0$, with one of them being heavier than others. We know that the counterfeit can be found with exactly $\log_3 n$ weighings using a divide-and-conquer algorithm. Consider the following algorithm that finds the counterfeit: Divide the n coins into $n/3$ groups of size 3. For each group, weigh any two coins. If one is heavier, we have found the counterfeit and stop. Else, put the third coin into a pile S. At the end of this step, if S has $n/3$ coins, we recursively solve the problem. The recursion stops when S has 3 coins and we then use the algorithm earlier to find the counterfeit. Let $t(n)$ be the **worst case exact** number of weighings required to find the counterfeit coin. (a) What is the recurrence for $t(n)$? Don't forget about the initial condition. (b) Solve the recurrence with all four methods.

IMPORTANT: The recurrence for $t(n)$ has to be for the exact number of weighings in the worst case while the solutions for $t(n)$ are asymptotic big Oh solutions. Also, correct solutions to incorrect recurrence for $t(n)$ will get little credits. Please write your answers to this question on the next page.

5. (10) The set partition problem can be defined as follows: given a set of n positive integers (not necessarily distinct, so the set is actually a multiset) $S = \{x_1, x_2, \cdots, x_n\}$, is there a partition of S into two subsets S_1 and S_2 such that $S_1 \bigcup S_2 = S$ and $S_1 \bigcap S_2 = \emptyset$ (empty set), and $\Sigma_{x_i \in S_1} x_i = \Sigma_{x_j \in S_2} x_j$? For example, $S = \{6, 1, 2, 3, 4\}$ has such a partition where $S_1 = \{6, 2\}$ and $S_2 = \{3, 1, 4\}$ while $S = \{1, 2, 5, 3, 20\}$ does not. This problem is known to be NP-Complete. We also introduced the problem of subset sum in class and in A4. Show that the subset sum problem is NP-Complete.

6. (10) In the following, NPC is the set of all NP-complete problems. Show that if $P \cap NPC \neq \emptyset$, then $P = NP$

7. (10) (a) Let P_1, P_2, ..., P_n be a set of n programs that are to be stored on a memory chip of capacity L. Program P_i requires a_i amount of space. Note that L and a_i's are all integers. Clearly, if $a_1 + a_2 + ...+ a_n \leq L$, then all the programs can be stored on the chip. So assume that $a_1 + a_2 + ... + a_n > L$. The problem is to select a maximum subset Q of the programs for storage on the chip. A maximum subset is one with the maximum number of programs in it. Give a greedy algorithm that always finds a maximum subset Q such that

$$\sum_{P_i \in Q} a_i \leq L.$$

You also have to prove that your algorithm always finds an optimal solution. Note that a program is either stored on the chip or not stored at all (in other words, you do not store part of a program).
(b) Suppose that the objective now is to determine a subset of programs that maximizes the amount of space used, that is, minimizes the amount of unused space. One greedy approach for this case is as follows: As long as there is space left, always pick the largest remaining program that can fit into the space. Prove or disprove that this greedy strategy yields an optimal solution.

8. (10) Given an array of integers $A[1..n]$, such that, for all i, $1 \leq i < n$, we have $|A[i] - A[i+1]| \leq 1$. Let $A[1] = x$ and $A[n] = y$, such that $x < y$. Using the divide-and-conquer technique, describe in English an $O(\log n)$ search algorithm to find j such that $A[j] = z$ for a given value z, $x \leq z \leq y$. Show that your algorithm's running time is $O(\log n)$ and that it is correct.

9. (5) Will the greedy algorithm discussed in class (always select the object with the largest value to weight ratio first) for solving the fractional knapsack problem optimally work for the 0-1 version of the problem? Prove your answer.