CS 344: Design and Analysis of Computer Algorithms **Rutgers: Spring 2022**

Midterm Exam #1

February 24, 2022

Name: _____ *NetID:* _____

Instructions

1. Do not forget to write your name and NetID above, and to sign Rutgers honor pledge below.
2. The exam contains 4 problems worth 100 points in total *plus* one extra credit problem worth 10 points.
3. This is a take-home exam. You have exactly 24 hours to finish the exam, starting from Thursday, February 24, 9am EST until Friday, February 25, 9am EST.
4. The exam should be done **individually** and you are not allowed to discuss these questions with anyone else. This includes asking any questions or clarifications regarding the exam from other students or posting them publicly on Piazza (**any inquiry should be posted privately on Piazza**). You may however consult all the materials used in this course (video lectures, notes, textbook, etc.) while writing your solution, but **no other resources are allowed**.
5. Remember that you can leave a problem (or parts of it) entirely blank and receive 25% of the grade for that problem (or part). However, this should not discourage you from attempting a problem if you think you know how to approach it as you will receive partial credit more than 25% if you are on the right track. But keep in mind that if you simply do not know the answer, writing a very wrong answer may lead to 0% credit.

   The only **exception** to this rule is the extra credit problem: you do not get any credit for leaving the extra credit problem blank, and there is almost no partial credit on that problem.
6. **You should always prove the correctness of your algorithm and analyze its runtime.** Also, as a general rule, avoid using complicated pseudo-code and instead explain your algorithm in English.
7. You may use any algorithm presented in the class or homeworks as a building block for your solutions.

Rutgers honor pledge:

On my honor, I have neither received nor given any unauthorized assistance on this examination.

Signature:_____

Problem. #	Points	Score
1	25	
2	25	
3	25	
4	25	
5	+10	
Total	100 + 10	

Problem 1.

(a) Determine the *strongest* asymptotic relation between the functions

$$f(n) = 2^{\sqrt{n}} \quad \text{and} \quad g(n) = n^4,$$

i.e., whether $f(n) = o(g(n))$, $f(n) = O(g(n))$, $f(n) = \Omega(g(n))$, $f(n) = \omega(g(n))$, or $f(n) = \Theta(g(n))$. Remember to prove the correctness of your choice. **(15 points)**

(b) Use the *recursion tree* method to solve the following recurrence $T(n)$ by finding the *tightest* function $f(n)$ such that $T(n) = O(f(n))$. **(10 points)**

$$T(n) \leq 3 \cdot T(n/2) + O(n^2)$$

Problem 2. Consider the algorithm below for finding the sum of numbers in an array A of size $n \geq 1$.

`TOTAL-SUM`$(A[1:n])$:

1. If $n = 1$, return $A[1]$.
2. Otherwise, return `TOTAL-SUM`$(A[1:n-1]) + A[n]$.

We analyze `TOTAL-SUM` in this question.

(a) Use **induction** to prove the correctness of this algorithm. **(15 points)**

(b) Write a recurrence for this algorithm and solve it to obtain a tight upper bound on the worst case runtime of this algorithm. You can use any method you like for solving this recurrence. **(10 points)**

Problem 3. You are given an *unsorted* array $A[1:n]$ of n distinct positive integers. Design an algorithm that in $O(n)$ worst-case time finds the largest number in $\{1, \ldots, n+1\}$ that is missing from A. **(25 points)**

Remember to *separately* write your algorithm (**10 points**), the proof of correctness (**10 points**), and runtime analysis (**5 points**).

Example. For $n = 8$ and $A = [9, 8, 70, 30, 2, 7, 4, 5]$ the correct answer is 6 (because the largest number in $\{1, \ldots, 9\}$ which is missing from A is 6).

Problem 4. We want to purchase an item of price M and for that we have a collection of n different coins in an array $C[1:n]$ where coin i has value $C[i]$ (we only have one copy of each coin). Our goal is to purchase this item using the *smallest* possible number of coins or outputting that the item cannot be purchased with these coins. Design a **dynamic programming** algorithm for this problem with worst-case runtime of $O(n \cdot M)$. **(25 points)**

Remember to *separately* write your specification of recursive formula in plain English (**7.5 points**), your recursive solution for the formula and its proof of correctness (**7.5 points**), the algorithm using either memoization or bottom-up dynamic programming (**5 points**), and runtime analysis (**5 points**).

Example:

- Given $M = 15$, $n = 7$, and $C = [4, 9, 3, 2, 7, 5, 6]$, the correct answer is 2 by picking $C[2] = 9$ and $C[7] = 6$ which add up to 15.
- Given $M = 11$, $n = 4$, and $C = [4, 3, 5, 9]$, the correct answer is that 'the item cannot be purchased' as no combination of these coins adds up to a value of 11 (recall that we can only use each coin once).

Problem 5 (**Extra credit**)**.** You are given an *unsorted* array $A[1 : n]$ of n distinct numbers with the promise that $A[1] > A[2]$ and $A[n] > A[n-1]$. Design an algorithm that in $O(\log n)$ worst case runtime finds an index $i \in \{2, \ldots, n-1\}$ such that $A[i]$ is smaller than both $A[i-1]$ and $A[i+1]$. If there are multiple indices with this property, your algorithm can output one of them. **(+10 points)**

Example: Given $A = [3, 2, 7, 8, 1, 4, 5, 6]$, one right answer is to output $A[2] = 2$, and another one is $A[5] = 1$.

Extra Workspace