CMPS 201
Final Review Problems

Be sure to review all prior homework assignments, midterm exams, and their solutions. Review all examples covered in class.

1. Suppose $T(n)$ satisfies the recurrence $T(n) = 3T(n/4) + F(n)$, where $F(n)$ itself satisfies the recurrence $F(n) = 5F(n/9) + n^{3/4}$. Find a tight asymptotic bound for $T(n)$. Be sure to fully justify each use of the Master Theorem. (Hint: $\log_9(5) < 3/4 < \log_4(3)$.)

2. Recall that $\{0, 1\}^*$ denotes the set of all bit-strings of any finite length. A language L is a collection of bit-strings, i.e. a subset $L \subseteq \{0, 1\}^*$. Let $A(x)$ be an algorithm whose input is a bit-string $x \in \{0, 1\}^*$, and whose output is 0 or 1.
   a. Define what it means for a language L to be *accepted* by A.
   b. Define what it means for a language L to be *decided* by A.

3. Show that any polynomial time algorithm for the optimization problem SP (SHORTEST-PATH) can be converted to a polynomial time algorithm for the decision problem PATH. (Input: a graph *G*, two vertices *u*, *v* and an integer *k*. Output: Yes if *G* contains a *u*-*v* path of length at most *k*, No otherwise.) Also show how to convert in the other direction, i.e. starting with a polynomial time algorithm for PATH, construct a polynomial time algorithm for SP.

4. Recall the decision problems HAMILTONIAN-CYCLE (HC) and TRAVELING-SALSEMAN-PROBLEM (TSP).

   HC: Given a graph *G*, determine whether or not *G* contains a Hamiltonian cycle (a cycle that visits every vertex in *G*).
   TSP: Given a complete graph K_n, a weight function $d: E(K_n) \to \mathbb{R}$, and a bound $b \geq 0$, determine whether or not K_n contains a Hamiltonian cycle of total weight no more than *b*.

   Recall also the mapping f:HC → TSP that takes instances of HC to instances of TSP, defined as follows. Given a graph *G* with $|V(G)| = n$, identify $V(G)$ with $V(K_n)$, define $d: E(K_n) \to \mathbb{R}$ by

   $$d(u, v) = \begin{cases} 1 & \text{if } \{u, v\} \in E(G) \\ 2 & \text{if } \{u, v\} \notin E(G), \end{cases}$$

   and let $b = n$.

   a. Prove that if G is a Yes instance of HC, then $f(G)$ is a Yes instance of TSP.
   b. Prove that if $f(G)$ is a Yes instance of TSP, then G is a Yes instance of HC.
   c. Explain how $f(G)$ can be computed in polynomial time. (Make some assumption as to how G will be represented, such as adjacency-list, adjacency-matrix, or incidence-matrix.)

5. Suppose we are given 4 gold bars (labeled 1, 2, 3, 4), one of which *may* be counterfeit: gold-plated tin (lighter than gold) or gold-plated lead (heavier than gold). Again the problem is to determine which bar, if any, is counterfeit and what it is made of. The only tool at your disposal is a balance scale, each use of which produces one of three outcomes: tilt left, balance, or tilt right.

a. Use a decision tree argument to prove that at least 2 weighings must be performed (in worst case) by any algorithm that solves this problem. Carefully enumerate the set of possible verdicts.
b. Determine an algorithm that solves this problem using 3 weighings (in worst case). Express your algorithm as a decision tree.
c. Find an adversary argument that proves 3 weighings are necessary (in worst case), and therefore the algorithm you found in (b) is best possible. (Hint: study the adversary argument for the min-max problem discussed in class to gain some insight into this problem. Further hint: put some marks on the 4 bars and design an adversary strategy that, on each weighing, removes the fewest possible marks, then show that if the balance scale is only used 2 times, not enough marks will be removed.)

6. (This is Problem 34.1-6 page 1061 of CLRS, see pages 1057-58 for definitions.) Show that the class P, viewed as a set of languages, is closed under union, intersection, concatenation, complement, and Kleene star. That is, if $L_1, L_2 \in P$, then $L_1 \cup L_2 \in P, L_1 \cap L_2 \in P$, $L_1L_2 \in P, \overline{L_1} \in P$, and $L_1^* \in P$.

7. Recall the coin changing problem again. Given denominations $d = (d_1, d_2, \dots, d_n)$ and an amount N, determine the number of coins in each denomination necessary to disburse N units using the fewest possible coins. Assume that there is an unlimited supply of coins in each denomination. Prove that the greedy strategy works for any amount N with the coin system $d = (1, 5, 10, 25)$.

8. **Scheduling to Minimize Average Completion Time:** (This is problem 16-2a on page 402 of CLRS.) Suppose you are given a set $S = \{a_1, a_2, \dots, a_n\}$ of tasks, where task a_i requires p_i units of processing time to complete, once it has started. You have one computer on which to run these tasks, and the computer can run only one task at a time. Let c_i be the *completion time* of task a_i, that is, the time at which task a_i completes processing. Your goal is to minimize the average completion time, that is to minimize the quantity $(1/n)\sum_{i=1}^{n} c_i$. For example, suppose there are two tasks, a_1 and a_2, with $p_1 = 3$ and $p_2 = 5$, and consider the schedule in which a_2 runs first, followed by a_1. Then $c_2 = 5$, $c_1 = 8$, and the average completion time is $(5 + 8)/2 = 6.5$.

   Give an algorithm that schedules the tasks so as to minimize the average completion time. Each task must run non-preemptively, that is, once task a_i is started, it must run continuously for p_i units of time. Prove that your algorithm minimizes the average completion time, and state the running time of your algorithm.

9. Let $B = b_1b_2 \dots b_n$ be a bit string of length n. Consider the following problem: determine whether or not B contains 3 consecutive 1's, i.e. whether B contains the substring "111". Consider algorithms that solve this problem whose only allowable operation is to peek at a bit.

   a. Suppose $n = 4$. Obviously 4 peeks are sufficient. Give an adversary argument showing that in general, 4 peeks are also necessary. (Hint: this is similar to problem 5 on hw7, and has a similar solution.)
   b. Suppose $n \geq 5$. Give an adversary argument showing that $4 \cdot \lfloor n/5 \rfloor$ peeks are necessary. (Hint: divide B into $\lfloor n/5 \rfloor$ 4-bit blocks separated by 1-bit gaps between them. Thus bits 1-4 form the first block, and bit 5 is the first gap. Bits 6-9 form the next block and bit 10 is the next gap, etc.. Any leftover bits form a separate block. Now run the adversary from part (a) on each of the 4-bit blocks.)