CSE 201
Final Exam Review Problems

Be sure to review all prior homework assignments, midterm exams, and their solutions. Also 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. 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 the handout 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.)

3. Recall the coin changing problem again. Given denominations $d = (d_1, d_2, \ldots, d_n)$ 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 $d = (1, 5, 10, 25)$.

4. **Scheduling to Minimize Average Completion Time:** (This is problem 16-2a on page 402 of CLRS.) Suppose you are given a set $S = \{a_1, a_2, \ldots, a_n\}$ 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 $p_2 = 5$. 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 $(5 + 8)/2 = 6.5$. Another schedule runs a_1 first, followed by a_2. In that case $c_1 = 3$, $c_2 = 8$, and the average completion time is $(3 + 8)/2 = 5.5$. Thus the schedule (a_1, a_2) has a better average completion time as compared to (a_2, a_1). Observe that a schedule is nothing more than a permutation of the set $S = \{a_1, a_2, \ldots, a_n\}$ of tasks.

   Give a greedy 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 runs continuously for p_i units of time. Prove that your algorithm minimizes the average completion time, and state its running time.

5. Let $B = b_1 b_2 \ldots b_n$ 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 7 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.)

6. **Matrix-Chain Multiplication Problem** State and prove a theorem that establishes that the *principle optimality* holds for the Matrix-Chain Multiplication problem. (Input: $p = (p_0, p_1, p_2, \ldots p_n)$ giving the sizes $p_0 \times p_1, p_1 \times p_2, \ldots p_{n-1} \times p_n$ of the matrices in a Matrix-Chain product $A_1 \cdot A_2 \cdots A_n$. Output: a parenthesization of the product that minimizes the number of real number multiplications that must be performed to compute the product.) In other words, show how and why an optimal parenthesization is composed of optimal parenthesizations of subproducts.

7. **Continuous vs. Discrete Knapsack Problem** Given values $v[1 \cdots 6] = (5, 5, 9, 4, 4, 12)$, weights $w[1 \cdots 6] = (1, 4, 3, 4, 1, 6)$ and capacity $W = 9$.

   a. Suppose these data constitute an instance of the discrete knapsack problem. (Determine $x \in \{0, 1\}^6$ that maximizes $\sum_{i=1}^{6} x_i v_i$ subject to the constraint that $\sum_{i=1}^{6} x_i w_i \leq W$.) Solve this problem using dynamic programming.

   b. Suppose these data constitute an instance of the continuous knapsack problem. (Determine $x \in [0, 1]^6$ that maximizes $\sum_{i=1}^{6} x_i v_i$ subject to the constraint that $\sum_{i=1}^{6} x_i w_i \leq W$.) Solve this problem using a greedy algorithm, using the selection function $f(i) = v_i / w_i$.

   c. Regard these data as an instance of the discrete knapsack problem again. Attempt to solve it using a greedy algorithm, using the selection function $f(i) = v_i / w_i$. If at some point in the greedy loop, you cannot include all of an object, skip it and go to the next "best" object, according to f. Does your answer have the same value that you found in part (a)?

8. Recall Kruskal's algorithm for finding a minimum weight spanning tree in a weighted connected graph G.

   - Select a minimum weight edge e in G, and set $F = \{e\}$.
   - Amongst all edges that do not create a cycle with the edges in F, choose one of minimum weight and add it to F.
   - Continue until $|F| = n - 1$, where $n = |V(G)|$.

   Prove that $T = (V(G), F)$ is a minimum weight spanning tree in G.

9. The following sorting algorithm, called BadSort() is a modified version of StoogeSort() from the 2nd edition of CLRS, which seems to have been left out of the 3rd edition.

```
BadSort(A, p, r)   pre: p ≤ r
1. if A[p] > A[r]
2.     A[p] ↔ A[r]  (swap)
3. if p + 1 ≥ r
4.     return
5. else
6.     q = ⌊(r − p + 1)/3⌋
7.     BadSort(A, p, r − q)
8.     BadSort(A, p + q, r)
9.     BadSort(A, p, r − q)
```

   a. Use induction on the length $m = r - p + 1$ of $A[p \cdots r]$ to prove the correctness of BadSort().
   b. Write a recurrence relation for the number of array comparisons performed by BadSort() on an array of length n.
   c. Use the Master Theorem to find an asymptotic solution to this recurrence, and explain what is bad about BadSort().

10. Define $T(n)$ by the recurrence

$$T(n) = \begin{cases} 0 & n = 1 \\ 2T(\lfloor n/2 \rfloor) + n \lg(n) & n \geq 2 \end{cases}$$

    Here lg means $\log_2$.

    a. Prove that the Master Theorem cannot be applied to this recurrence.
    b. Use the Substitution method to prove that $T(n) = O(n\,(\lg n)^2)$.

11. Write a recursive algorithm (modeled on MergeSort()) that determines if an array is sorted, i.e. given an array $A = (A_1, A_2, \ldots, A_n)$ as input, return TRUE/FALSE iff A is/is-not arranged in increasing order. Prove the correctness of your algorithm. Write a recurrence for the number $T(n)$ of array comparisons performed by your algorithm. Check that $T(n) = n - 1$ is the exact solution to your recurrence.