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. Let T be a *k*-ary tree with *n* leaves and height *h*. Prove that $h \geq \lceil \log_k(n) \rceil$. (Hint: Let $L(T)$ and $H(T)$ denote the number of leaves and the height (respectively) of the tree *T*, then proceed by induction on $h = H(T)$.)

3. Suppose you are given an unlimited supply of coins in each of the *n* denominations $d = (d_1, d_2, \ldots, d_n)$, and a number *N* of monetary units to be paid out using the least possible number of coins.

   a. Show that this problem exhibits *optimal substructure*, i.e. show how to divide the problem into subinstances of the same problem in such a way that every subinstance solution is a combination of other subinstance solutions.
   b. Write a recurrence relation for the *value of an optimal solution*. Include boundary and out of bounds values.
   c. Write a dynamic programming algorithm that fills in a table of values defined by the recurrence relation you wrote in (b). Your algorithm should take as input the vector *d* and the number *N*, and return the *value of an optimal solution*, i.e. the least number of coins necessary to pay *N* units, or returns ∞ if it is not possible to disburse *N* units using denominations *d*.
   d. Write a recursive procedure that, given the table of sub-instance solutions generated by the algorithm in (c), prints out a list of coins to be disbursed, i.e. print the *optimum solution itself*.

4. Given n objects with values $v = (v_1, v_2, \ldots, v_n)$ and corresponding weights $w = (w_1, w_2, \ldots, w_n)$, a thief wishes to steal a subset of the objects of maximum total value, and whose total weight does not exceed the capacity W of his knapsack.

   a. Show that this problem exhibits *optimal substructure*, i.e. show how to divide the problem into subinstances of the same problem in such a way that every subinstance solution is a combination of other subinstance solutions.
   b. Write a recurrence relation for the *value of an optimal solution*. Include boundary and out of bounds values.
   c. Write a dynamic programming algorithm that takes the vectors *v* and *w* and the number *W*, and returns the *value of an optimal solution*, i.e. the maximum value that can be stolen.
   d. Write a recursive algorithm that, given the table of sub-instance solutions generated by the algorithm in (c), prints out the *optimum solution itself*, i.e. prints out a list of which objects to steal.

5. State and prove a theorem the establishes that the *principle of optimality* holds for the Shortest-Path (SP) problem. (Input: a graph and two vertices (G, u, v). Output: the length of a shortest *u*-*v* path in *G*.) In other words, explain how and why a shortest path is composed of shortest paths.

6. 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. Let $v[1 \cdots 6] = (5, 5, 9, 4, 4, 12)$, $w[1 \cdots 6] = (1, 4, 3, 4, 1, 6)$ and $W = 9$.
   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 that $\{0, 1\}^*$ denotes the set of all bit-strings of any length. A language L is simply 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.

9. Show that any polynomial time algorithm for the optimization problem SP (defined in problem 5 above) 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.

10. Recall the decision problems Hamiltonian Cycle (HC) and Travelling Salesman (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.)