# 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. Write a dynamic programming algorithm that takes as input the vector $d$ and the number $N$, and returns the *value of an optimal solution*, i.e. the least number of coins necessary to pay $N$ units, or returns $\infty$ if it is not possible to disburse $N$ units using denominations $d$.
   b. Write a recursive procedure that, given the table of sub-instance solutions generated by the algorithm in (a), 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. 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.
   b. Write a recursive algorithm that, given the table of sub-instance solutions generated by the algorithm in (a), 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. 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 *decided in polynomial time* by the algorithm $A(\,)$.
   b. Define the complexity class $P$. (Hint: recall that $P$ is a set of languages.)
   c. Let $A(x, y)$ be an algorithm whose input is two bit-strings $x, y \in \{0,1\}^*$, and whose output is 0 or 1. The string $x$ represents a problem instance, any $y$ is called a *certificate*. Define what it means for a language $L$ to be *verified in polynomial time* by $A(\,,\,)$.
   d. Define the complexity class $NP$. (Hint: again $NP$ is a set of languages.)

7. Given languages $L_1, L_2 \in P$, prove that $L_1 \cap L_2 \in P$ and $L_1 L_2 \in P$.

8. 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.

9. 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.)