CSE 102
Midterm 2
Review Problems

1. Determine the largest integer k such that if there is a way to multiply 3×3 matrices using k multiplications (not assuming commutativity of multiplication), then there is an algorithm to multiply $n \times n$ matrices (where n is an exact power of 3) in time $o\left(n^{\lg(7)}\right)$. What would the run time of this algorithm be? (Hint: Proceed as in the analysis of Strassen's algorithm, but recur on submatrices of size $\frac{n}{3} \times \frac{n}{3}$.)

2. **Coin Changing Problem** 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*.

3. **Discrete Knapsack Problem** 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.

4. **Shortest Path Problem** 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.

5. **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.

6. Find an optimal parenthesization of a matrix-chain product whose sequence of dimensions is $(5, 10, 3, 12, 5, 50, 6)$. Show the complete tables $m[i, j]$ and $s[i, j]$ described in the handout on Dynamic Programming.

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.