COMP 271 Design and Analysis of Algorithms
2003 Spring Semester
Written Assignment # 3
Distributed: April 4, 2003
Revised: April 21, 2033
Due: April 24, 2003.

Note: Please follow the guidelines on doing your own work and avoiding plagiarism given on the class home page.

2nd Note: Problems 3, 4 and 5 are taken from the book *Problems on Algorithms,* by Ian Parberry, Prentice-Hall, 1995.

Prob 1. (10 pts) (CLRS 23.2-1)
Kruskal's algorithm can return different spanning trees for the same input graph G depending upon how ties are broken when edges are sorted. Prove that, for each minimum spanning tree T of G, there is a way to sort the edges of G in Kruskal's algorithm so that the algorithm returns T.

Prob 2. (20 pts) (CLRS 23.2-2)
Suppose that the graph $G = (V, E)$ is represented as an adjacency *matrix.* Give a simple implementation of Prim's algorithm that runs in $O(|V|^2)$ time.

Prob 3. (20 pts)
Let matrices A_1, A_2, A_3, A_4, A_5 have dimensions 8×3, 3×2, 2×19, 19×18, and 18×7, respectively. Use the dynamic progamming algorithm shown in class to find the least number of scalar multiplications needed to calculate the product $A_1A_2A_3A_4A_5$. You should show the completely filled in table and also the final way of parenthesizing the product.

Prob 4. (25 pts)
Suppose $A = \{a_1, a_2, \ldots, a_k\}$ is a set of distinct coin values (all the a_i are positive integers) available in a particular country. The *coin changing problem* is defined as follows: Given integer n find the *minimum* number of coins from A that add up to n assuming that all coins have value in A. You should assume that $a_1 = 1$ so that it is always possible to find some set that adds up to n.

For example, if , $A = \{1, 5, 8\}$ and $n = 26$ the best way of making change uses 4 coins, i.e., $\{5, 5, 8, 8\}$.

Design a $O(nk)$ dynamic programming algorithm for solving the coin changing problem; that is, given inputs A and n it outputs the minimum number of coins in A to add up to n. (It is not necessary to say what the coins are). Prove that your algorithm is correct.

Prob 5. (25 pts)

Arbitrage is the use of discrepancies in currency-exchange rates to make a profit. For example, there may be a small window of time in which 1 U.S. dollar buys 0.75 British pounds, 1 British pound buys 2 Australian dollars and 1 Australian dollar buys 0.70 U.S. dollars. Then, a smart trader can trade one U.S. dollar and end up with $0.75 \times 2 \times 0.7 = 1.05$ U.S. dollars, a profit of 5% (note that this assumes there are no trading costs).

Suppose that there are n currencies $c_1, \ldots, c_n$ and an $n \times n$ table R of exchange rates such that one unit of currency c_i buys $R[i,j]$ units of currency c_j. The *value* of a series of exchanges $c_{i_i}, c_{i_2}, \ldots, c_{i_t}$ is

$$R[i_1, i_2] \cdot R[i_2, i_3] \cdots R[i_{t-1}, i_t].$$

which is the number of units of currency c_t that result when starting with one unit of currency c_1, exchanging it into currency c_2, exchanging what results into c_3, etc. until ending by exchanging units of c_{t-1} in currency c_t.

A series of exchanges yields a profit if it both starts and ends in the same currency and the value of the series is greater than 1.

Devise and analyze a dynamic programming algorithm that, given the $R[\ ]$ table, determines whether or not it is possible to make a profit by trading currencies. You must prove that your algorithm is correct and also give the running time of the algorithm using $O()$ notation.