CSCI 533 Analysis of Algorithms
Homework #3
Due Monday, October 21st at Midnight

Instructions: Graduates do all five questions. Undergraduates should do any four of the five.

1. An $n$-node red-black tree $T$ has lost the color information in all nodes. You may assume the structure and the black-height information has been left intact. Give an $O(n)$ algorithm to recolor the nodes so that $T$ satisfies all the red-black properties.

2. Show that the longest simple path from a node $x$ in a red-black tree to a descendant leaf has length at most twice that of the shortest simple path from node $x$ to a descendant leaf.

3. This problem deals with the optimal order of matrix multiplication: i.e., you want to give an order for which computing $A_1 \cdot A_2 \cdots A_n$ which minimizes the number of scalar multiplications. Suppose that you have already constructed the table which allows you to solve this problem, and have kept this table in memory. You are given a new matrix $A_{n+1}$, and want to compute $A_1 \cdot A_2 \cdots A_n \cdot A_{n+1}$ Describe what you would do, and analyze the time you need to recompute the solution.

4. Your favorite sawmill charges by the length to cut a piece of lumber. For example, to cut a 7-foot piece (the cut may be anywhere) costs 7¢, and to cut a 10-foot piece costs 10¢. As an example, suppose you bring them the following 29-foot board with the cuts to be made as follows:

   ![Diagram of a 29-foot board with cuts](image)

   If you make the cuts in the (non-optimal) order, cut 1, cut 2, cut 3, cut 4, your cost is $29 + 25 + 15 + 10 = 79¢$. An optimal order is cut 2, cut 1, cut 3, cut 4, with cost $29 + 14 + 15 + 10 = 68¢$.

   At first glance, the following “solution” seems optimal: always make the next cut as close to the middle of some board as possible. Show that this does not always produce an optimal solution. Hint: make up a new example, don’t use the one here.

5. Design and analyze an efficient solution to solve the sawmill problem above. Hint: use dynamic programming.