Jordan Allard
CSCI 264-01
Homework 4

HW4 Problem 4 Writeup

a) Heart of the Solution

I) WHAT: S[L, R] = the minimum cost to multiply matrices A[L] through A[R]

II) HOW:

```
S[L, R] = 0 if L = R
S[L, R] = min( S[L, k] + S[k + 1, R] + a[L - 1]*a[k]*a[R] )
      where k = L through R - 1, if L < R
```

III) WHERE: S[1, n]

b) Pseudocode

```
Given an array A of matrix dimensions of length n
For L = 1 to n: S[L, L] = 0
For i = 1 to n:
      For L = 1 to n - i:
            R = L + i
            S[L, R] = infinity
            For k = L to R - 1:
                  tmp = S[L, k] + S[k + 1, R] + a[L - 1]*a[k]*a[R]
                  If tmp < S[L, R] then S[L, R] = tmp
Return S[1, n]
```

c) Proof of Correctness (Explanation of HOW)

This algorithm checks every possible combination of start and end matrices, and will always take the minimum each time. Therefore, it is guaranteed to find the right answer every time.

d) Running Time Estimate

O(n^3)

e) Running Time Estimate Reasoning

The running time estimate for calculating the total cost of the matrix multiplication is O(n^3), since there are three nested for loops that each can run up to n times in a worst case scenario. The reconstruction of the solution is a recursive algorithm where the running time of the non-recursive steps is O(1).

Within the recursion, there are two conditions that call the function again, where either L is increased by 1 or R is decreased by 1. In either case, the distance between L and R decreases by 1, and since L and R can only be up to n spaces apart, the recursion will have a running time of O(n) if it satisfies those conditions every time.

If the conditions are not satisfied, the recursion roughly follows the pattern T(n) = 2*T(n/2) + O(1). Technically the input is not divided by 2 every time, but it is divided evenly between the two calls, meaning that if one receives more input the other will receive less and the running time will be longer and shorter in a proportional manner. Since the running time of the non-recursive steps is O(1), the proportions are constant and thus can be accurately estimated by assuming an even division each time. Using the master theorem, T(n) satisfies Case #1, since O(1) = O(n^((log2/log2) - 1)), and so T(n) evaluates to O(n^(log2/log2)) = O(n).

Thus, no matter what path the recursion takes, the running time will be O(n).