

Exercises 2

Handed Out: Feb 9

Due: Feb 18

1. (20 pts) Solve the following recurrences. Assume in each case that $T(1) = 1$. Provide sufficient details to show how you got the answer. (For example, show how you applied the Master method, or details of any other method you used.)
 - (a) (5 pts) $T(n) = 4T(n/2) + n^3$.
 - (b) (5 pts) $T(n) = T(3n/5) + n$.
 - (c) (10 pts) Solve the following recurrence **both** using the Master Method and the basic recurrence expansion method.

$$T(n) = \sqrt{n}T(\sqrt{n}) + n$$

Why do you think you get different answers? Which answer is the correct one?

2. (15 pts)

What is the optimal way to compute $A_1A_2A_3A_4A_5$, where the dimension of A_1 is 10×4 , the dimension of A_2 is 4×5 , the dimension of A_3 is 5×20 , the dimension of A_4 is 20×2 , and the dimension of A_5 is 2×50 ? Show the final parenthesizing for the matrix chain multiplication.

3. (15 pts) Given three strings A , B , and C , you want to find the *longest subsequence* that is common to *all* three strings. Give a polynomial time algorithm for this problem. Present the pseudo-code for your algorithm in the manner of of longest common subsequence for two strings discussed in the class. Discuss the worst-case time complexity of your algorithm.
4. (20 pts) Let $G = (V, E)$ be a directed graph with nodes v_1, v_2, \dots, v_n . We say that G is an *ordered graph* if it has the following properties: (i) every directed edge (v_i, v_j) satisfies $i < j$; that is, each edge goes from a lower-index node to a higher-index node, and (ii) each node except v_n has at least one edge leaving it.

We want to compute the *longest path* in G starting at v_1 , where the length of a path is the number of edges in it.

- (a) Show that the following greedy algorithm does not always work (give a counterexample).

Start at v_1 . When at a node v_i , among all out-going edges of v_i , take the one that goes to the lowest-indexed node. Finish when v_n is reached.

In your counterexample, be sure to show what the correct answer is, and what the algorithm finds.

- (b) Give an efficient (polynomial-time) algorithm for solving this longest path problem, along with a proof of correctness and worst-case runtime analysis.