1. (*) In all of my knapsack examples in class, the value of each item \(v_i\) was chosen to be an integer. What would we have to change (if anything) about the algorithm if the value was chosen to be a real number, and how would that affect the runtime?

2. (**(*)) Suppose you are given a sequence of positive numbers \(a_1, a_2, \ldots, a_n\). You would like to find the largest ratio between two of these numbers, where the numerator occurs after the denominator in the sequence. That is, you would like to compute:

\[
\max \left\{ \frac{a_i}{a_j} : i, j \in \{1, 2, \ldots, n\} \text{ and } i > j \right\}.
\]

(a) [11 points] Assuming \(n \geq 2\), describe possibilities for the form of the optimal solution, and prove how you could transform that optimal solution into optimal solutions for subproblems. (That is, do steps 1 and 2 of our standard approach to dynamic programming.)

(b) [9 points] Write pseudocode for a dynamic programming algorithm that finds the maximum ratio.

(c) [3 points] What is the runtime of your algorithm?

3. (**(*)) Given two strings \(x\) and \(y\), create an algorithm to compute their optimal edit distance, where the edit distance the numbers of insertions, deletions, transpositions (switching the order of two adjacent letters), or substitutions in some sequence that changes \(x\) into \(y\). The optimal edit distance is the smallest possible edit distance. For example, we can change SHOAL to COLA by removing \(S\), changing \(H \rightarrow C\), and transposing the \(L\) and the \(A\). Edit distance is useful for spell checking applications, and genomic applications.

(a) [11 points] Let \(x = x_1 \ldots x_n\) and \(y = y_1 \ldots y_m\) be the two strings. Let \(x' = x_1 \ldots x_{n-1}\) and \(x'' = x_1 \ldots x_{n-2}\), and let \(y' = y_1 \ldots y_{m-1}\) and \(y'' = y_1 \ldots y_{m-2}\). Describe how the optimal edit distance of \(x\) and \(y\) depends on the optimal edit distance of subproblems. (Suppose for this part that \(x\) and \(y\) are both long strings.) (In this problem, you are basically doing parts 1 and 2 of our usual approach to dynamic programming problems. The proofs for the relevant cases are all similar to each other; do at least a couple of them.)

(b) [9 points] Write pseudocode for an algorithm to calculate the optimal edit distance of two strings.

(c) [3 points] What is the runtime of your algorithm?