Esercizi su Dynamic Programming

Exercise 1.1 Write an algorithm to find the maximum value that can be obtained with a full parenthesization of the expression

\[ \frac{x_1}{x_2/\frac{x_3}{\ldots \frac{x_{n-1}}{x_n}}}, \]

where \( x_1, x_2, \ldots, x_n \) are positive rational numbers and "/" denotes division.

Exercise 1.2 Give an algorithm that uses the table of additional information \( S[\cdot, \cdot] \) (computed by the Matrix-Chain Multiplication dynamic programming algorithm seen in class) to print the optimal parenthesization for the matrix chain.

Exercise 1.3 Given the string \( A = \langle a_1, a_2, \ldots, a_n \rangle \), we say that \( A_{i..j} = \langle a_i, a_{i+1}, \ldots, a_j \rangle \) is a palindrome substring of \( A \) if \( a_{i+h} = a_{j-h} \), for \( 0 \leq h \leq j - i \). (Intuitively, a palindrome substring is one which is identical to its “mirror” image. For example, if \( A = accaba \), then both \( A_{1..4} = acca \) and \( A_{4..6} = aba \) are palindrome substrings of \( A \).)

(a) Design a dynamic programming algorithm that determines the length of a longest palindrome substring of a string \( A \) in \( O(n^2) \) time and \( O(n^2) \) space.

(b) Modify your algorithm so that it uses only \( O(n) \) space, while the running time remains unaffected.

Exercise 1.4 Design and analyze a dynamic programming algorithm which, on input a string \( X \), determines the minimum number \( p \) of palindrome substrings of \( X, Y_1, Y_2, \ldots, Y_p \) such that \( X = \langle Y_1, Y_2, \ldots, Y_p \rangle \).

Exercise 1.5 Design and analyze a dynamic programming algorithm that, given in input a string \( X \), returns the maximum length of a palindrome subsequence of \( X \). The algorithm must run in time and space \( O(n^2) \).
Exercise 1.6  Given a string of arbitrary integers $Z = \langle z_1, z_2, \ldots, z_k \rangle$ let $\text{weight}(Z) = \sum_{i=1}^{k} z_i$ (note that $\text{weight}(\epsilon) = 0$). Given two integer strings $X = \langle x_1, x_2, \ldots, x_m \rangle$ and $Y = \langle y_1, y_2, \ldots, y_n \rangle$, design a dynamic programming algorithm to determine a Maximum-Weight Common Subsequence (MWCS) $Z$ of $X$ and $Y$.

Exercise 1.7  Design and analyze a dynamic programming algorithm which, on input two nonnegative integers $n$ and $k$, with $n > 0$ and $0 \leq k \leq n$, outputs $\binom{n}{k}$ by performing $\Theta(nk)$ sums. (Hint: Prove that for $0 < k < n$, $\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$.)

Exercise 1.8  Given two strings $X$ and $Y$, a third string $Z$ is a common superstring of $X$ and $Y$, if $X$ and $Y$ are both subsequences of $Z$. (Example: if $X = \text{sos}$ and $Y = \text{soia}$, then $Z = \text{sosia}$ is a common superstring of $X$ and $Y$.) Design and analyze a dynamic programming algorithm which, given as input two strings $X$ and $Y$, returns the length of the Shortest Common Superstring (SCS) of $X$ and $Y$ and additional information needed to print the SCS. The algorithm must run in time $\Theta(|X||Y|)$. (Hint: Use an approach similar to the one used to compute the LCS of two strings.)

Exercise 1.9  Given two strings of integers $Z^1$ and $Z^2$, with $|Z^1| = |Z^2| = k$, we define their discrepancy as $d(Z^1, Z^2) = \sum_{i=1}^{k} |Z^1_i - Z^2_i|$. Design and analyze a dynamic programming algorithm which, on input two (arbitrary) strings of integers $X$ and $Y$, computes the maximum discrepancy obtainable by a subsequence of $X$ and a subsequence of $Y$ of equal length by performing $\Theta(|X||Y|)$ comparisons and sums between integers.

Exercise 1.10  Let $n > 0$. Given a string of $n$ integers $A = \langle a_1, a_2, \ldots, a_n \rangle$, consider the following recurrence, defined for all pairs $(i, j)$, with $1 \leq i \leq j \leq n$:

$$B(i, j) = \begin{cases} a_i & 1 \leq i = j \leq n, \\ \max\{B(i, k) \cdot B(k + 1, j) : i \leq k \leq j - 1\} & 1 \leq i < j \leq n. \end{cases}$$

Design and analyze an iterative bottom-up algorithm that, on input $A$, returns $B(1, n)$ by performing $O(n^3)$ sums.

Exercise 1.11  Let $n > 0$. Assume that a given dynamic programming strategy leads to the following recurrence, defined for all values of $i$ and $j$ with $1 \leq i \leq j \leq n$:

$$C(i, j) = \begin{cases} 1 & (i = 1) \text{ and } (j = n), \\ \sum_{r=1}^{i-1} C(r, j) + \sum_{s=j+1}^{n} C(i, s) & \text{altrimenti}. \end{cases}$$

Design and analyze an iterative bottom-up algorithm that computes all values $C(i, j)$, $1 \leq i \leq j \leq n$. 

2
Exercise 1.12  Given the following bottom-up code:

\[
\text{DP}\_\text{SUM}(n) \\
\text{for } i \leftarrow 1 \textbf{ to } n \textbf{ do } A[i, i] \leftarrow i \\
\text{for } \ell \leftarrow 2 \textbf{ to } n \textbf{ do} \\
\quad \text{for } i \leftarrow 1 \textbf{ to } n - \ell + 1 \textbf{ do} \\
\qquad j \leftarrow i + \ell - 1 \\
\text{return } A[1, n]
\]

write an equivalent memoized code and analyze its running time in terms of sums between integers.

Exercise 1.13  Given a string \( X = \langle x_1, x_2, \ldots, x_n \rangle \), consider the following recurrence \( \ell(i, j) \), defined for \( 1 \leq i \leq j \leq n \):

\[
\ell(i, j) = \begin{cases} 
1 & i = j, \\
2 & i = j - 1, \\
2 + \ell(i + 1, j - 1) & (i < j - 1) \land (x_i = x_j), \\
\sum_{k=i}^{j-1} (\ell(i, k) + \ell(k + 1, j)) & (i < j - 1) \land (x_i \neq x_j).
\end{cases}
\]

Design memoized code to return the value \( \ell(1, n) \) and analyze the code both in the worst case and in the best case, assuming that the only unit-cost operations are character comparisons.