Practice problems

Problem (greedy-choice proof)

Assume that you are given two unsorted arrays \( A = \{a_1, a_2, \ldots, a_n\} \) and \( B = \{b_1, b_2, \ldots, b_n\} \) of \( n \) positive integers. We want to determine an ordering of the elements of \( A \) and \( B \) such that \( W = \prod_{i=1}^{n} a_i b_i \) is maximized. Consider the following greedy algorithm.

Algorithm GREEDY\( (A, B) \): sort \( A \) and \( B \) in decreasing order; return \( (A, B) \)

Show that the greedy-choice property holds for the algorithm GREEDY (no need to prove that the problem has the optimal substructure property).
Problem (greedy-choice proof)

Consider the following problem. Given a set \( \{x_1, x_2, \ldots, x_n\} \) of points on the real line, determine the smallest set of unit-length closed intervals that contains all the given points. We propose the following greedy algorithm.

Sort the points, and start from the smallest point \( x \). Repeat the following loop: add a unit-interval whose left endpoint is at \( x \), then go over the points (starting at \( x \)) until a point \( y > x + 1 \) is found (or no more points remain). If such a \( y \) is found, let \( x = y \), and repeat.

Analyze the time complexity of the algorithm, and prove or disprove the optimality of the algorithm. If you think the algorithm does not always achieve the optimal solution, show an example. Otherwise, prove the greedy choice (no need to prove the optimal substructure).

Dynamic Programming
Dynamic programming (1D design)

- Given a rod of length \( n \), and a set of prices \( p_i \), \( i = 1, 2, \ldots, n \) for a rod piece of length \( i \), give a dynamic programming algorithm that determines the maximum profit obtainable by cutting up the rod and selling the individual pieces.
Dynamic programming (2D design)

We are given a list of $n$ items with sizes $s_1, s_2, \ldots, s_n$. A sequential bin packing of these items is an assignment of items to bins such that in each bin the items are consecutive. (That is, each bin has items $s_i, s_{i+1}, \ldots, s_j$ for some indices $i < j$.) Bins have unbounded capacities. The load of a bin is the sum of the elements in it. Give an algorithm that determines a sequential packing of $n$ items into $k$ bins for which the maximum load of a bin is minimized. Analyze the time-complexity and space-complexity.
Solution

**Answer:** Let’s define \( S[i, j] \) be the minimum cost of a bin packing for item \( s_1, s_2, \ldots, s_i \) into \( j \) or less bins (where the cost of a bin packing is defined as max load of the bins). We have:

\[
S[i, j] = \begin{cases} 
q_i & i = 1, j = 1 \\
\sum_{k=1}^{i} n_k & i > 1, j = 1 \\
\min_{k \leq j \leq i} \max(k \cdot (j - 1), \sum_{k+1}^{j} n_k) & i > 1, j > 1
\end{cases}
\]