Exercise 1 (9 bonus points) For each of the following problems, determine whether it can be solved in strongly subquadratic time (i.e. in time \( O(n^{2-\varepsilon}) \) for some \( \varepsilon > 0 \)).
Prove your claims by giving either an algorithm running in strongly subquadratic time or a hardness proof that rules out such an algorithm under some conjecture discussed in the course.

a) (3 bonus points) Longest Palindrome Subsequence: Given a string \( S \) of length \( n \), find the longest subsequence that is a palindrome (i.e., a sequence of characters which reads the same backwards and forwards).

b) (3 bonus points) Non-Dominating Vectors (Constant Dimension): Given a set \( A \subseteq \mathbb{Z}^d \) of \( n \) integer vectors, \( d = O(1) \), compute the set \( A' \subseteq A \) of non-dominated vectors. (A vector \( a \in A \) dominates another vector \( a' \in A \) if \( a_i \geq a'_{i} \) for all \( 1 \leq i \leq d \) and \( a \neq a' \)).

c) (3 bonus points) Non-Dominating Vectors (Low Dimension): Given a set \( A \subseteq \mathbb{Z}^d \) of \( n \) integer vectors, \( d = \log^3 n \), compute the set \( A' \subseteq A \) of non-dominated vectors.

Exercise 2 (6 bonus points) The Minimum Consecutive Sums Problem is defined as follows:

**MCSP**: Given \( n \) integers \( x_1, x_2, \ldots, x_n \), determine for any \( 1 \leq k \leq n \) the minimal sum of any \( k \) consecutive of these integers, i.e., compute for any \( 1 \leq k \leq n \) the number

\[
\min\{x_i + \ldots + x_{i+k-1} \mid 1 \leq i \leq n - k + 1\}.
\]
Prove that \((\min,+)\)-Convolution and MCSP are equivalent in the following sense:
\[
(MCSP, n^2) \leq_{fg} ((\min,+)\text{-Convolution}, n^2) \leq_{fg} (MCSP, n^2).
\]

**Exercise 3** *(5 bonus points)* From your algorithms classes you may know the problem of finding a string \(P\) (often called *pattern*) in another string \(T\) (often called *text*). This well-known problem is often called *Pattern Matching*; there are algorithms for this problem that run in time \(O(|P| + |T|)^1\).

Instead of finding a single pattern string \(P\), we are now interested in finding *any substring* of \(T\) that can be generated by a given *regular expression*. Formally, consider the following problem:

**RegExPatternMatching**: Given a regular expression \(R\) of size \(m\), and a text \(T\) of size \(n\), determine if any substring \(P\) of \(T\) can be derived from \(R\).

In general, there is no algorithm running in time \(O((mn)^{1-\epsilon})\) (for any \(\epsilon > 0\)) for **RegExPatternMatching** unless OVH fails. However, for *specific classes* of regular expressions, there are faster algorithms to solve this problem. Consider *homogeneous regular expressions*:

A regular expression \(R\) is called *homogeneous of type* \(\ldots o_1 o_2 \ldots o_l \ldots\) (where \(o_i \in \{\circ, *, +, |\}\)) if there exist \(a_1, \ldots, a_p\), characters or homogeneous regular expressions of type \(o_2 \ldots o_l\), such that \(R = o_1(a_1, \ldots, a_p)\).

For example, the regular expression \([(a \circ b \circ c) | b | (a \circ b)]^*\) is homogeneous of type \("*|\circ"\), the regular expression \((a^*) | (b^+)\) is not homogeneous.

a) *(1 bonus point)* Give an \(O(m + n)\) time algorithm for **RegExPatternMatching** where the regular expression is homogeneous of type \("\circ"\) or of type \("*|\circ"\).

b) *(4 bonus points)* Prove that there is no \(O((mn)^{1-\epsilon})\) algorithm (for any \(\epsilon > 0\)) for **RegExPatternMatching** where the regular expression is homogeneous of type \("|\circ |"\) unless OVH fails.
   Prove the same result for homogeneous regular expressions of type \("|\circ *"\).

---

1\(^{*}\)See for example Knuth, Morris, and Pratt’s algorithm.