Exercise 2 bonus points

Read the lecture notes (of the last three lectures), identify as many typos and other mistakes as you can, and add them as a list to your solutions. You get one bonus point for at least one typo/mistake and 2 bonus points for at least five typos/mistakes.

Exercise 1 6 + 6 points

Recall that in the lecture, we generalized $OV$ to the following problem:

$k$–OV: Let $k$ sets $A_1, A_2, \ldots, A_k \subseteq \{0, 1\}^d$ with $|A_1| = |A_2| = \ldots = |A_k| = n$ be given. Decide whether there exist $a^{(1)} \in A_1, a^{(2)} \in A_2, \ldots, a^{(k)} \in A_k$ such that in every dimension the corresponding component of at least one of the vectors $a^{(1)}, a^{(2)}, \ldots, a^{(k)}$ is 0.

Consider the following hypothesis about this family of problems:

$kOVH$: For no $k \geq 2$ and $\varepsilon > 0$, there is an algorithm for $k$–OV running in time $O(n^{k-\varepsilon} \cdot \text{poly}(d))$.

a) In the lecture, we introduced the $q$–Dominating Set problem:

$q$–DomSet: Given a graph $G = (V, E)$, decide whether there is a subset of the vertices $S \subseteq V$ of size $q$, such that for any vertex $v \in V$, either $v \in S$ or $\{u, v\} \in E$ for some $u \in S$.

Prove that $q$–DomSet cannot be solved in time $O(n^{q-\varepsilon})$ for all $\varepsilon > 0$ and integers $q \geq 3$, unless $kOVH$ fails.

b) Consider the following variant of $kOVH$:

$kOVH'$: For no $k \geq 100$ and $\varepsilon > 0$, there is an algorithm for $k$–OV running in time $O(n^{k-\varepsilon} \cdot \text{poly}(d))$.

Show that $kOVH$ and $kOVH'$ are equivalent.
In this exercise, we will prove further results about \( \text{OV} \).

a) Give an algorithm for \( \text{OV} \) running in time \( \tilde{O}(n^2) = O(n^2 \cdot \text{poly log}(n)) \) for vectors of dimension \( d = n^{0.1} \).

b) Show that if \( \text{OV} \) can be solved in time \( T(n, d) \), then given any \( \text{OV} \) instance we can also find an orthogonal pair, if it exists, in time \( O(T(n, d)) \).

c) Adapt the \( \text{OV} \) algorithm from the lecture to also find an orthogonal pair, if it exists, in the same asymptotic running time of \( n^{2 - 1/\text{O(log } c)} \). (Recall that for the algorithm from the lecture, the vectors have a dimension of \( d = c \cdot \text{log}(n) \) with \( c = n^{o(1)} \).) Your solution to this exercise must be different from your solution of part b) above.

Exercise 3  
8 points + 10 bonus points + 5 points

Recall the Longest Common Substring With Don’t Cares problem from the previous exercise sheet:

Longest Common Substring With Don’t Cares: Given a string \( A \) of length \( n \) over some alphabet \( \Sigma \) and string \( B \) of length \( n \) over the alphabet \( \Sigma \cup \{\ast\} \), find the length \( L(A, B) \) of the longest string that is a substring of both \( A \) and \( B \), where a “\( \ast \)” in \( B \) can be treated as any character from the alphabet \( \Sigma \).

In this exercise, we consider only the binary alphabet \( \Sigma = \{0, 1\} \).

a) Let strings \( A \in \{0, 1\}^n, B \in \{0, 1, \ast\}^n \) be such that their longest common substring has length \( L(A, B) \leq c \cdot \text{log}(n) \) with \( c = n^{o(1)} \).

Show how to compute the length \( L(A, B) \) of the longest common substring of \( A \) and \( B \) in time \( n^2 - 1/\text{O(log } c) \).

⋆) Given strings \( A \in \{0, 1\}^n, B \in \{0, 1, \ast\}^n \) and \( \Delta \in \mathbb{N} \). Show how to determine whether the longest common substring of \( A \) and \( B \) has a length of at least \( \Delta \), that is, whether \( L(A, B) \geq \Delta \), and if so, how to compute \( L(A, B) \); both in time \( O(n^2/\sqrt{\Delta}) \).

Hint 1: You may assume you can solve the following problem in time \( O(n \cdot \text{log } m) \): Given a text \( T \in \{0, 1\}^n \) and a pattern \( P \in \{0, 1, \ast\}^m \) (with wildcards), determine all occurrences of \( P \) in \( T \), that is, all indices \( 1 \leq i \leq n - m + 1 \) such that \( T[i..i+m-1] \) and \( P \) match.

Hint 2: Use the following approach: Divide \( T, P \) into blocks and try to find completely matching block pairs to establish a good lower bound on \( L \). Once you were successful, try to extend matching substrings as much as possible.

b) Show that Longest Common Substring With Don’t Cares can be solved in time \( n^2/2^{\Omega(\sqrt{\text{log } n})} \).

(Note: You can use a) and ⋆) even if you didn’t solve them.)