Notation and preliminaries: The $\tilde{O}(\cdot)$ notation hides polylogarithmic (in n) factors.

1 Not Orthogonal Vector

Consider the following problem and conjecture.

Definition 1.1 (NOV)**.** *Given two sets $A, B \subseteq \{0,1\}^d$ of n binary vectors of dimension d, decide if there is a vector $a \in A$ that is not orthogonal to any vector in B, i.e. whether $\exists a \in A$ s.t. $\forall b \in B : \langle a, b\rangle \neq 0$.*

Conjecture 1 (NOV)**.** *No $O(n^{2-\varepsilon})$ time algorithm, where $\varepsilon > 0$, solves the NOV problem with $d = \Theta(\log^2 n)$.*

First, let us show that this conjecture is useful.

(a) Prove that, assuming the NOV Conjecture, no algorithm can compute the Radius of a sparse graph on n nodes and $\tilde{O}(n)$ edges in $O(n^{1.9})$ time.

Now, let's try to unify it with the OV Conjecture (that is implied by SETH).

(b) Prove that NOV is in $NTIME[\tilde{O}(n)] \cap CoNTIME[\tilde{O}(n)]$,[1] and explain in a few words why this is a barrier for basing the NOV Conjecture on SETH.

Finally, let's show that NOV is in fact a stronger conjecture than OV.

(c) Prove that if OV on two sets of n vectors in dimension $d = \Theta(\log^2 n)$ can be solved in $O(n^{1.9})$ time (refuting the OV conjecture and SETH), then the NOV Conjecture is false.

Hint: Note that you are asked to reduce a certain kind of "detection/finding" version (NOV) to the standard "detection/finding" version (OV). It may be helpful to think about the "listing" version(s) as well.

2 Dynamic SCC

Prove that, assuming SETH, no algorithm can maintain the number of strongly connected components[2] in a directed graph G on n nodes and $m = \tilde{O}(n)$ edges, throughout a sequence of $O(m)$ updates, where each update either adds or removes an edge from G, in a total of $O(n^{1.9})$ time.

[1] A simple way to prove this is to design two algorithms A^{nondet} and $A^{conondet}$. Both solve NOV in $\tilde{O}(n)$ time and are allowed to make guesses. If the input is a "no" instance, algorithm A^{nondet} must reject no matter what it guesses, and if the input is a "yes" instance then there must be a guess that makes A^{nondet} accept. On the other hand, $A^{conondet}$ must always accept if the input is a "yes" instance, while there must exist a guess that makes it reject if the input is a "no" instance.

[2] Recall that a *strongly connected component* is a maximal subset of nodes S such that for any pair $u, v \in S$: u can reach v, and v can reach u. We assume that a node can reach itself, so there is always a (unique) partition of the graph into strongly connected components.