Orthogonal Vectors and the Subquadratic World

Deepanshu Kush

June 9, 2020

Outline of the talk

- Motivation: Fine-grained complexity
- Orthogonal Vectors (OV) problem definition and OV Conjecture
- Connection to SETH
- Connection to other quadratic time problems
- A couple reductions, and examples
- The cubic cousin, APSP (if time permits)

Motivation: Fine-grained Complexity

String Problems

- Edit distance:
  - between two strings, defined as the min number of insertions, deletions or substitutions of symbols needed to transform one string into another
  - applications in computational biology, natural language processing and information theory
  - Simple quadratic-time DP algorithm, but even the best one takes $\tilde{O}(n^2)$ time
- Longest Common Subsequence (similar story)
- Sequence local alignment:
  - Given two sequence of bases

    AGGCTATCACCTGACCTCCAGGCCGATGCCC
    TAGCTATCACGACCGCGGTCGATTTGCCCGAC

    Compute an "alignment":

    -AGGCTATCACCTGACCTCCAGGCCGA--TGCCC---
    TAG-CTATCAC--GACCGC--GGTCGATTTGCCCGAC
  - Again DP takes $O(n^2)$ time

String Problems (contd.)

So, no $\tilde{O}(n^{2-\varepsilon})$ algorithm known for Edit distance, Longest Common Subsequence or Sequence local alignment

Turns out this is a recurring theme for many problems in computational geometry too, like given n points, checking if some 3 are collinear (and many more geometry problems, as we shall see today)

Even for certain graph problems like calculating the diameter of a sparse ($O(n)$ edges) graph

So could the hardness of solving these string problems and geometry problems in truly sub-quadratic time be related? Are we stuck on all of them for the **same underlying reason**?

If so, parallels with NP-hardness? Knapsack, TSP, colouring, Independent Set, Candy Crush(!) are all NP-hard, or in other words, harder than SAT

Could there be a "SAT" for TIME($\tilde{O}(n^2)$), that all these problems are "harder" than? Such a thing as "sub-quadratic" reduction? Like many-one/Turing reductions between NP-complete problems

Identifying one key quadratic problem

- Orthogonal Vectors! (Like "SAT" for TIME($\tilde{O}(n^2)$)?)
- Turns out we can show that a multitude of problems are *OV-hard:*

Graph diameter [RV'13,BRSVW'18],

eccentricities [AVW'16],

local alignment, longest common substring* [AVW'14], Frechet distance [Br'14], Edit distance [BI'15], LCS, Dyn. time warping [ABV'15, BrK'15], subtree isomorphism [ABHVZ'15], Betweenness [AGV'15], Hamming Closest Pair [AW15], Reg. Expr. Matching [BI16,BGL17], and a ton more!

The Orthogonal Vectors Problem: Definition and Hardness Conjecture

OV: Problem Description

- Two vectors $u, v \in \{0,1\}^d$ (or binary strings of length d) are orthogonal if $\sum_{i\in[d]} u_i \cdot v_i = 0$
- Sum is considered over $\mathbb{R}$ (not $\mathbb{F}_2$)
- Equivalently, they are orthogonal if $\bigvee_{i\in[d]} u_i \wedge v_i = 0$ (there is no position at which both vectors have a 1)

Problem:

- Input: Two lists A, B of n d-dimensional $0-1$ vectors
- Output: "Accept" iff there is an orthogonal pair $(u, v) \in A \times B$

*problem has been considered over other discrete structures too, like the rings $\mathbb{Z}_N$ (esp. in Williams, Yu (SODA '14)), but not our focus today

What is d?

- Obvious brute-force running time of $O(n^2 \cdot d)$
- If d is sufficiently smaller than n (for e.g., $d \ll \log n$), we must have redundant vector copies in each list, so we can weed them out first and then brute-force
- In particular, it follows that if $d \leq (1-\varepsilon)\log n$ for some *constant* $\varepsilon > 0$, then there is a $O(n^{2-\varepsilon} \cdot d) = \tilde{O}(n^{2-\varepsilon})$ time algo for $OV_{n,d}$
- Natural question: What about $d = c \log n$ for *any* constant c?
- Specifically, is there a **universal** constant $\varepsilon > 0$ so that for every constant c, $OV_{n,c \log n}$ can be solved in $\tilde{O}(n^{2-\varepsilon})$ time?
- **Orthogonal Vectors Conjecture (OVC) [R. Williams, Theor. Comp. Sci. '05]: No, there is not!**

Remarks:

- Think of this regime ($d = O(\log n)$) as the *smallest* possible for which $OV_{n,d}$ becomes interesting. OVC says that *even* in this case, "truly sub-quad. time" is impossible
- Note the order of quantifiers here! Because for a *given* constant c, $\tilde{O}(n^{2-\varepsilon_c})$ is possible, for ε_c depending on c

Connection to SETH: why we believe in OVC

Strong Exponential Time Hypothesis: Introduction

- $k - CNF - SAT$:
  - Input: Boolean variables $x_1, \ldots, x_n$ and a formula in the conjunctive normal form i.e. of the form $C_1 \wedge \cdots \wedge C_m$ where each C_i is the logical OR of at most k variables (or their negations)
  - Output: "Accept" iff there exists an assignment to these variables on which this formula evaluates to 1
- Obvious $O(2^n \cdot mn)$ algorithm
- SETH asserts that we can't do much better for arbitrary k. More precisely:
- **SETH**: for every $\varepsilon > 0$, there is a k such that $k - CNF - SAT$ on n variables, m clauses cannot be solved in $2^{(1-\varepsilon)n} \cdot \mathrm{poly}(m)$ time
- Equivalently, if there is a $2^{(1-\varepsilon)n} \cdot \mathrm{poly}(m)$ time algorithm for some $\varepsilon > 0$ that can solve SAT on CNF Formulas (for all k) on n variables and m clauses , then SETH is false

SETH implies OVC!

- Contrapositive: Want to show that a “fast” algo for OV yields “fast” algo for SAT
- In other words, given a SAT instance on n variables $x_1, \dots, x_n$ and m clauses $C_1, \dots, C_m$, want to construct an OV instance on which we can apply this supposed “fast” algo
- This OV instance will have lists A, B of size $N = 2^{n/2}$, consisting of binary strings (vectors) of length m
- How to define these vectors? Use “split and list”. Split variable set into halves: $\{x_1, \dots, x_{n/2}\}$ and $\{x_{n/2+1}, \dots, x_n\}$. A then consists of vectors u_α, where α is a partial assignment that assigns bits to the first half of variables. B consists of the set of v_β
- $u_\alpha(i) = \begin{cases} 1, \text{ if } \alpha \text{ does not satisfy } C_i \\ 0, \text{ otherwise} \end{cases}$ $\quad v_\beta(i) = \begin{cases} 1, \text{ if } \beta \text{ does not satisfy } C_i \\ 0, \text{ otherwise} \end{cases}$
- So u_α, v_β are orthogonal iff $\alpha \cup \beta$ satisfies all the clauses
- Note that it takes $O(2^{n/2} \cdot m)$ time to go from a given SAT instance to defining these lists A, B
- If there is an algo that solves $OV_{N,d}$ in $\tilde{O}(N^{2-\varepsilon})$ time, then SAT, after above reduction, on any k can be solved in time

$$O\left(2^{n/2} \cdot m + \left(2^{n/2}\right)^{2-\varepsilon}\right) = O\left(2^{\left(1-\frac{\varepsilon}{2}\right)n}\right)$$

- This contradicts SETH!

Connection to Other Quadratic Time Problems

More than just hardness: “completeness”!

- Not only are many problems OV-hard, a decent number are even known to be “sub-quadratically equivalent” to OV (typically, latter is difficult to show than just hardness)
- That is, $\tilde{O}(n^{2-\varepsilon})$ time algorithm for one implies $\tilde{O}\left(n^{2-\delta}\right)$ time algorithm for the other, and vice-versa
- Of these, one simple & immediate example is *Subset Query*: given n “query” sets $S_1, \dots, S_n \subseteq [d]$ and n “database” sets $T_1, \dots, T_n \subseteq [d]$, is $S_i \subseteq T_j$ for some i, j?
- Note that $S_i \subseteq T_j$ iff $S_i \cap \overline{T_j} = \emptyset$
- To see the equivalence with OV, simply think of the sets as binary vectors and flip the bits of the “database” vectors
- The reduction is clearly $O(n \cdot d)$ time and so, sub-quadratic equivalence follows

Chen and Williams (SODA '18): An equivalence class for OV

Theorem: Either all of the following can be solved in truly sub-quadratic time, or none of the following:

- (OV) Finding an orthogonal pair among n vectors.

Uses CC

- (Min-IP/Max-IP) Finding a red-blue pair of vectors with minimum (respectively, maximum) inner product, among n vectors.
- **(Exact-IP) Finding a red-blue pair of vectors with inner product exactly equal to a given integer, among n vectors.**
- (Apx-Min-IP/Apx-Max-IP) Finding a red-blue pair of vectors that is a 100-approximation to the minimum (resp. maximum) inner product, among n vectors.

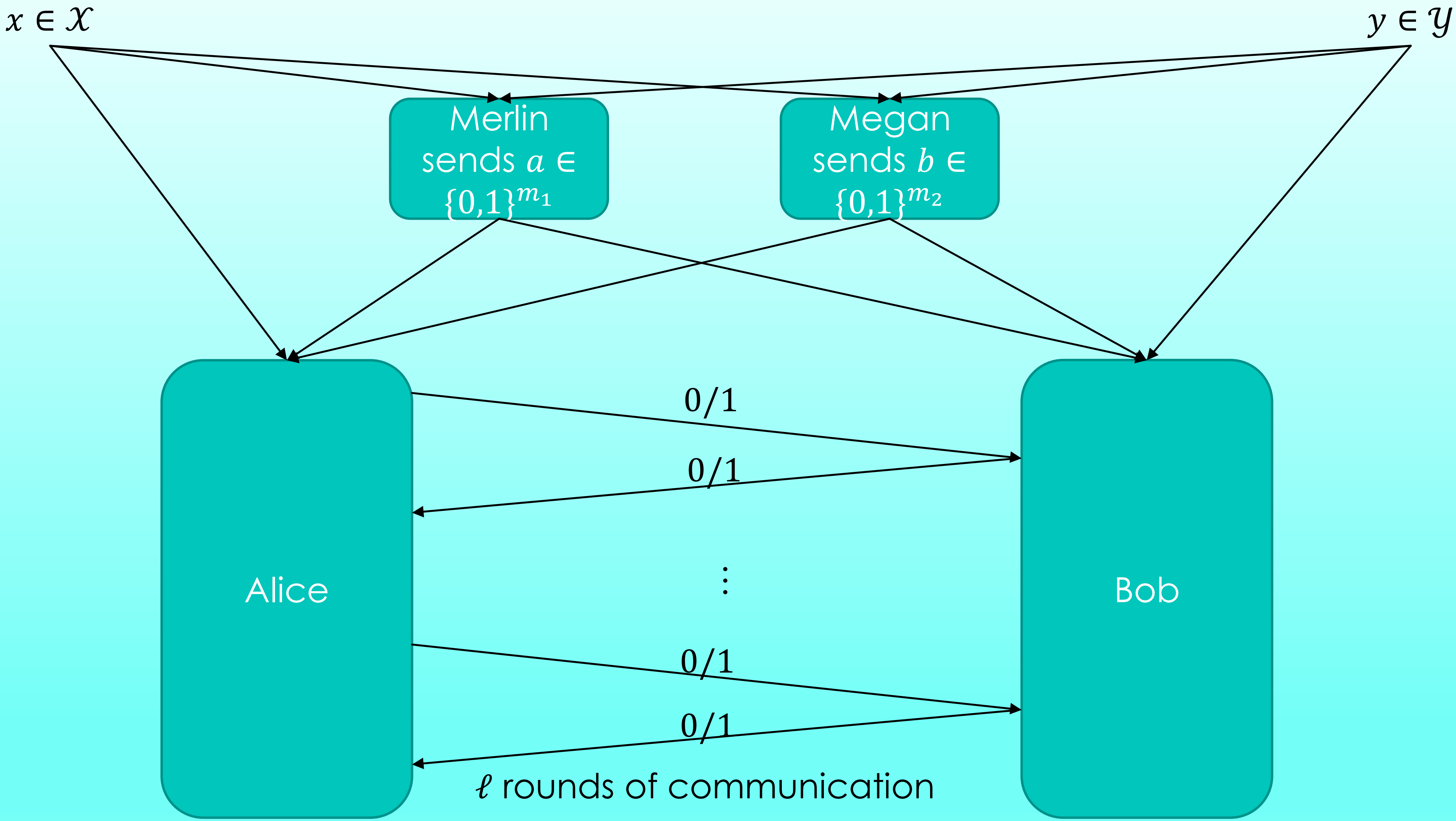
Uses LSH

- (Approximate Bichromatic ℓ_p-Closest Pair) Approximating the ℓ_p -closest red-blue pair (for a constant $p \in [1,2]$), among n points.
- (Approximate ℓ_p -Furthest Pair) Approximating the ℓ_p -furthest pair (for a constant $p \in [1,2]$), among n points.

… and more

Σ_2 Communication Protocols

- Want to compute a function $F : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$
- Like $IP_{d,m}$, which on given two binary strings of length d, checks if their IP is the integer m
- A Σ_2^{cc} protocol Π for F is specified as follows:
  - Two players, Alice (holds input $x \in \mathcal{X}$)and Bob (holds $y \in \mathcal{Y}$)
  - Two "provers" Merlin and Megan
  - Merlin sends a string $a \in \{0,1\}^{m_1}$ and Megan sends a string $b \in \{0,1\}^{m_2}$ (functions of x and y) to both Alice and Bob
  - Alice and Bob then communicate ℓ bits with each other, and Alice decides whether to accept or reject the pair (a, b)
  - $F(x,y) = 1$ iff there exists a string a from Merlin, such that for all strings b from Megan, Alice accepts (a,b) after communications with Bob
  - Protocol Π is computationally-efficient, if both Alice and Bob's response functions can be computed in polynomial time with respect to their input length



Then Alice decides whether to accept or reject (a, b)

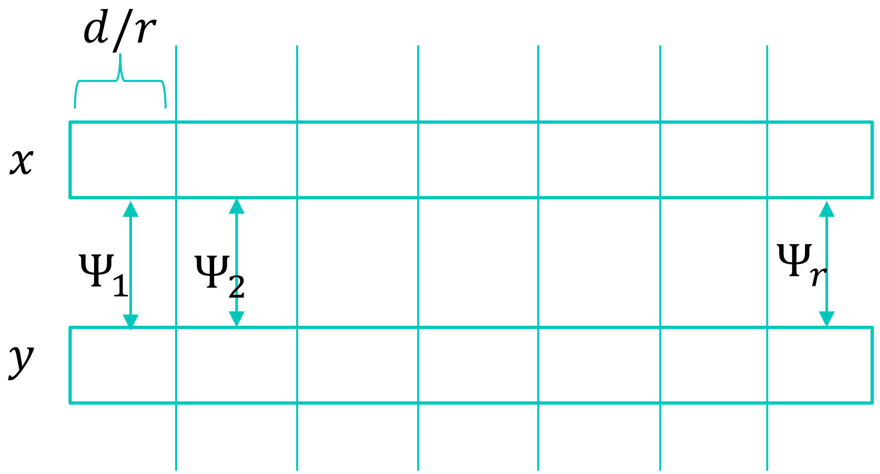
$F(x, y) = 1$ iff $\exists$ string a from Merlin, such that for all strings b from Megan, Alice accepts (a, b)

Efficient Σ_2 protocol ⇒ Sub-quadratic Reduction to OV

- $w_1, w_2, \ldots, w_{2^\ell}$ be all possible communication transcripts between Alice and Bob
- Given $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, generate vector $R_x(a,b)$, $R_y(a,b)$ $\in \{0,1\}^{2^\ell}$ as follows:
  - for all a, $R_x(a,b)_i = 1$ iff transcript w_i is consistent with Alice's input x, and w_i makes Alice reject
  - Similarly, $R_y(a,b)_i = 1$ iff w_i is consistent with (Bob's input y)
- Only one w_i is consistent with both x and y given the pair (a,b) (transcript fixed once x, y, a, b are)
- So $R_x(a,b)$, $R_y(a,b)$ are orthogonal iff Alice accepts the pair (a,b)
- Suppose given Exact-IP_m instance I with sets A and B of n vectors from $\{0,1\}^d$
- Idea is to reduce I to *several* (but not too many) instances of OV: I is a yes instance iff one of these several OV instances is a yes instances
- enumerate Merlin's possible strings $a \in \{0,1\}^{m_1}$; $R_x(a,\cdot)$ denotes the concatenation of all $R_x(a,b)$'s $b \in \{0,1\}^{m_2}$. $R_y(a,\cdot)$ is defined similarly
- A_a be the set of all $R_x(a,\cdot)$, B_a the set of all $R_y(a,\cdot)$
- I is a yes instance if and only if *some* pair (A_a, B_a) is a yes instance for OV

Requires a little argument

Σ_2 Protocol for Exact-IP: Simple Idea



Block them up into r parts of length d/r each. Let Merlin send across (ideally) the vector of IPs of blocks $(\Psi_1, \dots, \Psi_r)$ and Megan send across an index $j \in [r]$.

Alice rejects immediately if the sum of Ψ_i doesn't match with m

Otherwise, Alice checks if the jth block IP (i.e. that of x_j and y_j) matches what Merlin claims it to be i.e., Ψ_j and accepts Merlin & Megan's claim iff Ψ_j is indeed $\langle x_j, y_j \rangle$

Protocol correctly decides Exact-$IP_{d,m}$

Combining things together

- Just left to check that the reduction from Exact-IP to OV is indeed sub-quadratic
- That is, given a universal constant $\delta > 0$ such that for all constants c' , $OV_{n,c'\log n}$ can be solved in $n^{2-\delta}$ time. Need to find a universal constant $\delta' > 0$ such that for all constants c , $IP_{c \log n,m}$ can be solved in $n^{2-\delta'}$ time
- This is done by analysing the efficiency of the Σ_2 protocol described earlier, and that of the process of obtain the reduction to OV from the protocol
- Can be done by setting the right r (turns out to be about $\log n$)

One Fast(er) algorithm for Orthogonal Vectors

(Don’t worry, still not violating OVC)

Fast Algorithm for OV

- Reminder: OVC states that there is no **universal** constant $\varepsilon > 0$ so that for every constant c, $OV_{n,c \log n}$ can be solved in $\tilde{O}(n^{2-\varepsilon})$ time?
- But for a *given* c, one may still hope for $\tilde{O}(n^{2-\varepsilon_c})$ time
- And indeed, Abboud, R. Williams, and Yu (SODA '15) prove the following:
- **Theorem:** For Boolean vectors of dimension $d = c(n) \log n$, OV can be solved in $n^{\left\{2-\frac{1}{O(\log c(n))}\right\}}$ time by a randomized algorithm that is correct with high probability
- T. M. Chan and R. Williams (SODA '16) derandomize this:
- **Theorem:** There is a *deterministic* algorithm for $OV_{n,\, d=\, c(n)\; log\; n}$ that runs in $n^{\left\{2-\frac{1}{O(\log c(n))}\right\}}$ time, provided $d \leq 2^{\{(\log n)^{\{o(1)\}}\}}$

All hail the polynomial method

- Checking if a pair of vectors $(x_i, y_j) \in A \times B$ is orthogonal is the formula
$$E(x_i, y_j) = \wedge_{k=1}^{d} (\neg x_i[k] \vee \neg y_j[k])$$
- Block them up into s parts $A_1, \ldots, A_s$ & $B_1, \ldots, B_s$, each containing n/s vectors (s tbd)
- Write down the formula that evaluates if there is an orthogonal pair in $A_i \times B_j$ (big OR of s^2 pairs of $E(\cdot,\cdot)$)
- Convert that formula into a polynomial, of not-too-large degree! How?
- Razborov & Smolensky in the 80s figured out low-degree "probabilistic" polynomials that "approximate" AND and OR functions really well
- Finally, set s accordingly to use "fast rectangular matrix multiplication" by Coppersmith

  ($\exists$ constant $C \approx 0.172$ s.t. multiplication of an $N \times N^C$ matrix with an $N^C \times N$ matrix can be done using $\tilde{O}(N^2)$ arithmetic operations)

Fast Algorithms for Other Quadratic-Time Problems

- Alman, Chan and R. Williams (FOCS '16) give faster than brute-force algorithms for other problems we've discussed today:

  Min-IP & Max-IP, Exact-IP, Bichromatic-ℓ_p-Closest-Pair, ℓ_p-Furthest-Pair
- This time using *Probabilistic Polynomial Threshold Functions (PTFs)* (so yes, more polynomial method)

All-Pairs Shortest Paths and The Sub-cubic World

A Cubic Cousin and A Quadratic Sibling

- Even more is known about APSP. The following problems are "sub-cubically equivalent" to APSP:

Negative Triangle, Triangle Listing, Shortest Cycle, 2nd Shortest Path, Max Subarray, Graph Median, Graph Radius and Wiener Index

- Main paper is by V. & R. Williams (FOCS '10)
- An excellent resource for all things fine-grained related is the ICM 2018 survey of V. Williams
- 3-SUM: given a set S of n integers from $\{-n^4, \dots, n^4\}$, determine whether there are $x, y, z \in S$ such that $x + y + z = 0$
- a simple $O(n^2 \log n)$ time enumeration algorithm: sort S and then for every $x, y \in S$, check if $-z \in S$ using binary search
- **3-SUM Hypothesis (formulated in early 90s):** 3-SUM on n integers in $\{-n^4, \dots, n^4\}$ cannot be solved in $O(n^{2-\varepsilon})$ time for any $\varepsilon > 0$ by a randomized algorithm
- Actually, it seems that there is more literature on APSP and 3-SUM than on OV
- Many problems in comp. geom. Known to be 3-SUM-hard, including 3-Collinearity Testing
- **No relation known between 3-SUM, APSP, and SETH however**

Thank you for listening!