COMS E6998-15: Fine-grained complexity (Spring'26) February 24, 2026

Lecture 6: The 3SUM Problem

Instructor: *Josh Alman* Scribes: *Christopher Henry, Anthony Chang, Andrew Mihailoff (2022), Aapeli Vuorinen(2022)*

Project Proposal due March 12!
Choose a subject that is interesting to you!
Possibilities:
1) Either a Research Project or Reading Project-

- Choose either 1 long paper or 2-3 shorter ones.
- Then give a synthesis/presentation.
- Don't just recap, but make something that is valuable to the other students in the class. e.g. Like focus on one result, and expand on why its interesting and where it came from, or one special case of the result -> how it simplifies another proof, etc.
- Like if a paper says "this has applications to 3-SUM" but leaves it there, then try expanding on that.
- Projects often morph from one to another.

- Like if a paper says "this has applications to 3-SUM" but leaves it there, then try expanding on that.
- Projects often morph from one to another.

- (Sometimes a research project is too ambitious / hits a wall, then you can morph it into a reading project)
- You can be very flexible (like an implementation project), but anything that's too diff from the previous two, talk to Josh first maybe.
- Talk to Baitian and/or Josh if you need help, suggestions, feedback, etc!

3 Main deliverables:
1) Project Proposal- March 12

- About 1 page
- What you will do?
- Why you're interested?
- (if relevant) How it's related to the class?

2) In class presentation- April 14,21

- Around 10-15 mins for 1 person, around 20-30 for 2.
- The goal is **NOT** to impress Josh. The #1 goal is for <u>everyone</u> in the room to understand the presentation!!!!
- Baitian is happy to hear practice presentations! If you do so, you will get bonus points!

3) Writeup- May 7

- Around 5-10 pages per person. Here, your goal is to go into more detail that you couldn't in the presentation.
- Goal is to write so that the typical student in the class can understand and follow.

The 3-SUM Problem

We now turn our attention to the 3-SUM Problem, the last of the 3 main conjectures we will discuss in this class (the first two being the SETH / Orthogonal Vectors and All Pairs Shortest Path).

Definition 1 (3SUM problem)**.** *Given sets A, B, C of integers with $A| = |B| = |C| = n$, determine whether there exist $a \in A, b \in B, c \in C$ such that $a + b + c = 0$.*

How fast can we solve this problem? A trivial solution can be found in $O(n^3)$ time, or in time $O(n^2 \log n)$ with a little more elbow grease [1]. The author in [Cha20] gives an $O(\frac{n^2 (loglogn)^3}{(logn)^2})$ time algorithm. How low can we go? The 3SUM Conjecture conjecture posits that there is no truly sub-quadratic solution to the 3SUM problem.

Definition 2 (3SUM Conjecture)**.** 3SUM *defined on $A, B, C \subseteq \{-n^4, \dots, n^4\}$ with $|A| = |B| = |C| = n$ cannot be solved in time $O(n^{2-\varepsilon})$ for any $\varepsilon > 0$.*

Input: Sets A,B,C integers with $|A| = |B| = |C| = n$
Determine if there are $a \in A$, $b \in B$, $c \in C$ s.t. $a + b + c = 0$.
How fast can we solve this?

- Trivially, iterating through all items gives you $O(n^3)$.
- With a little more work, we can sorting a, then doing binary search on each pair of b, c gives you $O(n^2 logn)$.
- Chan '20 gives $O(\frac{n^2 (loglogn)^3}{(logn)^2})$

How low can we go? 3SUM Conjecture says not much more.

3Sum Conjecture: 3SUM with integers from $A, B, C \subseteq \{-n^4, ..., n^4\}$ cannot be solved in time $O(n^{2-\epsilon})$ for any $\epsilon > 0$.

While this is fantastic, the bounds of $|n^4|$ doubtless raises questions. However, we show that the general case can be reduced to the bounded case in $O(n \log n)$ time.

Theorem 3. *For $c > 4$, there is an $O(n \log n)$ time randomized reduction from* 3SUM *on $\{-n^c, \dots, n^c\}$ to* 3SUM *on $\{-n^4, \dots, n^4\}$.*

Proof. We begin by choosing a random prime p s.t. $n \leq p \leq n^{3+\delta}$. We then move from 3SUM to 3SUM $\bmod p$ by reducing as follows: $A' = \{a \mod p \mid a \in A\}, B' = \{b \mod p \mid b \in B\}, C' = \{c \mod p \mid c \in C\}$.
First, observe that if $a + b + c = 0$, then $a + b + c \mod p = 0$. We then make the argument that the probability of a false probability (e.g. $a + b + c \mod p = 0 \mod p$ while $a + b + c \neq 0$) is low. We then show how to solve 3SUM $\bmod p$ to finally arrive at a randomized algorithm for this reduction.

Lemma 4. *For any $t \in \{1, 2, .., 3n^c\}$ there are at most $c + 1$ primes between n and $n^{3+\delta}$ that divide t.*

[1]e.g. sorting A, then conducting binary search on each (b, c) pair s.t. $a = -b - c$

Proof. t is the product of its prime divisors. The product of the primes dividing t and exceeding n must then be less than $3n^c$, so there can be at most $O(c)$ such primes. $\square$

Therefore, the number of primes between n and $n^{3+\delta}$ that could create a false positive is $\leq (c+1)n^3$. We then bound the total number of primes within the range:

Lemma 5. *There are $\Omega(\frac{n^{3+\delta}}{\log n})$ primes between n and $n^{3+\delta}$.*

Proof. This is a consequence of the classical approximation $\pi(n) \approx n/\log(n)$ to the prime-counting function π (the prime-counting function $\pi(n)$ counts the number of primes up to and including n). See [IR82] for a proof. $\square$

Aided by these two lemmas, we first choose some random prime p between n and $n^{3+\delta}$. One can choose such a prime in polynomial time by repeatedly picking a random number in this interval and doing a primality test until one finds a prime. By Lemma 5, a randomly chosen number in this interval is a prime with probability $O(1/\log n)$ and one can test if a given number is prime in time that is polylogarithmic in n [AKS04]. So overall, the expected running time for picking a random prime in this interval is polylogarithmic in n.

To go from 3SUM(mod p) to 3SUM, make the observation that $a+b+c \equiv 0 \mod p$ iff $a+b+c = kp, k \in \{0, \pm1, \pm2, \pm3\}$. Therefore, $\forall k$, run regular 3SUM on sets $\{A, B, C'\}$, where $C' = \{c - kp | c \in C\}$. If 3SUM ever returns true (i.e. $a+b+c' = 0$), then return true. Else, return false. $\square$

As usual, there is also a deterministic algorithm to solve the proof, but we leave it as an an exercise to the reader :).

Finally, we provide a loose process for solving a bounded 3SUM problem in sub-quadratic time using a polynomial method and Fast Fourier Transform, provided the weights are bounded by a $U \ll n^4$.

Theorem 6. 3SUM *with $A, B, C \subseteq \{-U, \ldots, U\}$ can be solved in time $O(n + U\log(U))$.*

Proof. We first add U to each value in A, B, C. Denote by A' the set $\{x + U : x \in A\}$, and similarly for B' and C'. Note that $A', B', C' \subseteq \{0, \ldots, 2U\}$. We now seek $a \in A'$, $b \in B'$, and $c \in C'$ such that $a + b + c = 3U$.

Next define the following three polynomials:

$$p_A(x) := \sum_{a \in A} x^a$$
$$p_B(x) := \sum_{b \in B} x^b$$
$$p_C(x) := \sum_{c \in C} x^c.$$

These are all single-variable polynomials of degree at most $2U$.

We then compute

$$p_A(x) \cdot p_B(x) \cdot p_C(x) = \sum_{a \in A} \sum_{b \in B} \sum_{c \in C} x^{a+b+c},$$

and we check the coefficient of x^{3U}. If the coefficient is 0 (that term does not appear in the polynomial), then there is no $a \in A', b \in B'$, and $c \in C'$ such that $a + b + c = 3U$; whereas if the coefficient is 1, then such a, b, c do exist.

The running time of this algorithm is $\mathcal{O}(n + U \log U)$, the first term coming from creating the polynomials, and the second from computing their products using the Fast-Fourier Transform [OS75]. $\square$

1 Reductions from 3SUM to Problems in Computational Geometry

Now, we turn to reductions from 3SUM to problems in computational geometry. It is interesting to note that these reduction predate much of the machinery in fine-grained complexity theory like the precise formulation of SETH, and fine-grained reductions, which have been developed in the last decade years. We first define the Geombase problem, which we will reduce to many of the upcoming problems. The reductions in this section come from [GO95].

Definition 7 (Geombase Problem). *Given $P \subseteq \mathbb{Z} \times \{0, 1, 2\}$ with $|P| = n$, determine whether there exists a non-horizontal line passing through any three points in P.*

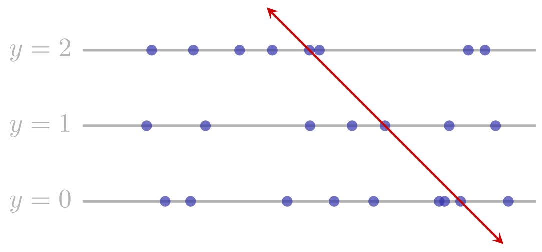

Figure 1: Visual depiction of the Geombase Problem (YES instance).

Theorem 8. *Geombase can be solved in time $T(n)$ if and only if* 3SUM *can be solved in time $\mathcal{O}(T(n))$.*

Proof. Note that $(a, 0)$, $(b, 1)$, and $(c, 2)$ lie on a line if and only if $a + c = 2b$.[2]
Let A, B, C be sets from an instance of 3SUM. Then we construct P our set of points as

$$P = \{(2a, 0) \mid a \in A\} \cup \{(-b, 1) \mid b \in B\} \cup \{(2c, 2) \mid c \in C\}.$$

If we feed this into our algorithm $\mathcal{A}$ for Geombase, we can detect whether there exists a non-horizontal line passing through three points in P; if such a line exists, we can recover $a \in A$, $b \in B$, $c \in C$ such that $a + b + c = 0$. Since constructing P is done in linear time, if an algorithm $\mathcal{A}$ exists solving geombase in $T(n)$ then we have found an algorithm that can solve 3SUM in $\mathcal{O}(T(N))$. $\square$

Definition 9 (Separation Problem). *Given n line segments in a plane. Determine whether there is a line separating the segments into two disjoint, non-empty sets.*

[2]Slope through $(a, 0)$ and $(b, 1)$ is $1/(b - a)$. Slope through $(b, 1)$ and $(c, 2)$ is $1/(c - b)$. Slope must be equal so $1/(b - a) = 1/(c - b) \implies a + b = 2c$.

Theorem 10. *Assuming the* 3SUM *Conjecture, the Separation problem requires* $\Omega(n^{2-\varepsilon})$ *time.*

Proof. We reduce the Geombase problem into the Separation problem. This is illustrated in Figure 1. Given an instance of Geombase, we create line segments along the lines $y = 0$, $y = 1$, and $y = 2$ such that they are between each point of the Geombase instance, and leave a gap of ε between the point and the start of the line segment. Additionally we add vertical line segments at the ends of this grid (at say, $x = \min_{(x_i,y_i)\in P} x_i - 1$ and $x = \max_{(x_i,y_i)\in P} x_i + 1$ from $y = 0$ to $y = 2$) so that no horizontal line will separate the points.

We can now solve Geombase using a Separation algorithm, since any line that separates these line segments cannot be horizontal and must pass through three of the 2ε-size holes corresponding to the points in Geombase. If we pick ε sufficiently small, we have a line that passes through all points in the Geombase instance. (Note that this works as the points in Geombase are all integer.)

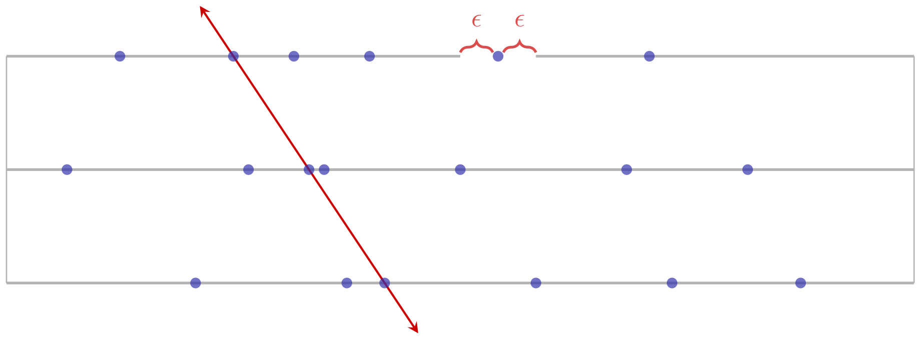

Figure 2: Visual depiction of the Separator to Geombase reduction.

□

Definition 11 (Planar Motion Planning Problem)**.** *Given* n *line segments (obstacles) in plane, a source point, and a target point, determine whether a line segment of length* 1 *(a "robot"), can be moved using only translations and rotations from the source to target without intersecting any of the obstacles.*

Theorem 12. *Assuming the* 3SUM *Conjecture, the Planar Motion Planning Problem requires* $\Omega(n^{2-\varepsilon})$ *time.*

Proof. We again refer to Figure 1 to demonstrate the construction of a reduction from Geombase to Planar Motion Planning.

Similar to the Separation problem, we construct a set of obstacles in the same patterns as the lines in the Separation problem, but this time we scale the lines to be within 1/8 of each other. Furthermore, place the source below this set of lines, and the target above the lines; as well as an enclosure both around the source and the target. This way if it is possible to move the robot using only translations and rotations through the obstacles, this must correspond to a non-horizontal line connecting the points in the Geombase, similar to the Separation reduction. This is because at some point the line, which is longer than 1/4, must pass through three gaps in the obstacles in the middle. □

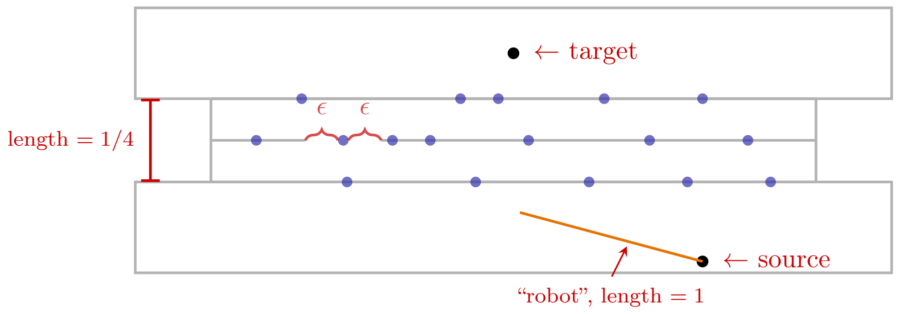

Figure 3: Visual depiction of Geombase to Planar Motion Planning reduction.

A different computational geometry problem is the 3-points-on-a-line problem, which is defined next. It's particularly useful to examine as many computational geometry algorithms start by assuming a set of points are in *general position*, meaning that no three of them are co-linear. In such a case, algorithms often require pre-processing in the presence of co-linear points.

Definition 13. *(3-points-on-a-line problem). Given n points in a plane with integer coordinates, determine if any three of the points lie on a line.*

In order to develop a reduction from 3SUM to 3-points-on-a-line, we first require a different version of 3SUM.

Definition 14 (One-set 3SUM)**.** *Given a set $S \subseteq \{-n^4, \cdots, n^4\}$ ($|S| = n$) of integers, determine whether there distinct $a, b, c \in S$ such that $a + b + c = 0$.*

Lemma 15. *One-set* 3SUM *is sub-quadratic time equivalent to* 3SUM*.*

Proof. 3SUM → One-Set 3SUM. We define the set S as follows

$$S = \{8a + 1 \mid a \in A\} \cup \{8b + 3 \mid b \in B\} \cup \{8c - 4 \mid c \in C\}.$$

We can feed this into our solver for One-Set 3SUM to get a solution for 3SUM.
One-set 3SUM → 3SUM. We can randomly partition the set S into three sets A, B, and C. We feed these three sets into our solver for 3SUM. If a solution exists in S then with constant probability[3] we will find the the solution. Since this probability is constant we can repeat the process to achieve a high success probability. □

Theorem 16. *Assuming the* 3SUM *Conjecture, there is no $\mathcal{O}(n^{2-\varepsilon})$ algorithm for 3-points-on-a-line.*

Proof. Consider some instance of one-set 3SUM with $S \subseteq \{-n^4, \cdots, n^4\}$ and $|S| = n$. We construct an instance of 3-points-on-a-line by creating points X, and show that there exist three points that lie on a line in X, if and only if S has three distinct values that sum to 0.

Let $X = \{(a, a^3) \mid a \in S\}$. We will show that for distinct $a, b, c \in \mathbb{Z}$, $a + b + c = 0$ if and only if the points (a, a^3), (b, b^3), and (c, c^3) lie on a line. To do so, note that these three points lie on a line if and only if

[3] $\frac{3!}{3^3} = \frac{2}{9}$

$$\frac{c-a}{b-a} = \frac{c^3-a^3}{b^3-a^3} = \frac{(c-a)(c^2+ac+a^2)}{(b-a)(b^2+ab+a^2)} \iff b^2+ab = c^2+ac \iff (b-c)(a+b+c) = 0.$$

Since a, b, c are distinct, we must have that (a, a^3), (b, b^3), and (c, c^3) lie on a line if and only if $a+b+c = 0$. □

References

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Annals of Mathematics*, 160:781–793, June 2004. Godel Prize, Fulkerson Prize.

[Cha20] Timothy M. Chan. More logarithmic-factor speedups for 3SUM, (median, +)-convolution, and some geometric 3sum-hard problems. *ACM Trans. Algorithms*, 16(1):7:1–7:23, 2020.

[GO95] Anka Gajentaan and Mark H. Overmars. On a class of O(n^2) problems in computational geometry. *Comput. Geom.*, 5:165–185, 1995.

[IR82] Kenneth Ireland and Michael Rosen. *A classical introduction to modern number theory*, volume 84 of *Graduate texts in mathematics*. Springer, 1982.

[OS75] Alan V. Oppenheim and Ronald W. Schafer. *Digital signal processing*. Prentice-Hall international editions. Prentice-Hall, 1975.