Chapter 17. 3SUM Conjecture

`#project-hardness`

Hyunseo Jeong (Karuna)
jhwest@snu.ac.kr

Seoul National University

March 29, 2023

Overview

What do we study in this Chapter?

- The world of *"Can we solve this when* $N = 100\,000$*?"*
  - This 3SUM problem doesn't seem so, so why don't we just make a whole complexity class out of it?
- 3SUM-hardness of geometric problems
  - This has to do with relating $a + b + c = 0$ with collinear conditions.
- Further interesting results involving 3SUM conjecture
  - Proving lower bounds of dynamic data structures (dynamic reachability, dynamic bipartite matching, ...)
  - More about polynomial lower bounds

3SUM

The classical 3SUM problem everyone knows.

3SUM

Given N integers. Determine if there exist 3 integers a, b, c such that $a + b + c = 0$.

- Homework (no) : Find a $\mathcal{O}(N^2)$ algorithm for 3SUM.

3SUM

In spite of the simplicity of the problem, it turns out that no *truly* subquadratic algorithm solving 3SUM is found.

What do I mean by *truly* here?

Theorem (Baran et al. 2005)

3SUM can be solved in $\mathcal{O}(\frac{n^2(\log\log n)^2}{(\log n)^2})$ time.

Even though the folklore algorithm can be improved, yet no algorithm is faster than $\mathcal{O}(n^{2-\epsilon})$.

Note that $\mathcal{O}(n^{2+\epsilon})$ does not accept $\mathcal{O}(n^2 polylog(n))$ algorithms.
$\mathcal{O}(n^{1.99}\log^{100}(n)) = \mathcal{O}(n^{2-\epsilon})$ but $\mathcal{O}(\frac{N^2}{\log^{100}(n)}) \neq \mathcal{O}(n^{2-\epsilon})$.

3SUM

3SUM Conjecture

Any algorithm for 3SUM requires $\mathcal{O}(n^{2-\epsilon})$ expected time.

3SUM-hardness

A problem B is **3SUM-hard** if a $\mathcal{O}(n^{2-\epsilon})$ algorithm for B induces a $\mathcal{O}(n^{2-\epsilon})$ algorithm for 3SUM.

Note that our conjecture includes both deterministic and randomized algorithms. Sadly, randomization could not help breaking the subquadratic barrier for 3SUM.

From now on, we always mean expected time when we talk about time complexity.

3SUM

There are a few things to note, for the sake of rigour.

- If each integers lie in $[-u, u]$ for $u = \mathcal{O}(N^{2-\epsilon})$, then we can solve 3SUM in subquadratic time using FFT. So we want the integers to be big enough but not too big. In fact, when restricted to $[-N^3, N^3]$, the problem remains to be equivalently hard. Some studies use this version as a definition of 3SUM.
- Things get complicated if we start assuming arithmetic operations as $\mathcal{O}(\log^{\alpha}(n))$. So we assume a Word RAM model of $\mathcal{O}(\log n)$ bits, so that simply we can assume each arithmetic operations take $\mathcal{O}(1)$ time.

Lower bounds on Geometric Problems

Most of the elementary results are based on the relation between $a + b + c = 0$ and collinearity.

First, we state a variant of 3SUM which is easier to handle in most cases.

3SUM'

A problem of given three sets A, B, C and determining whether there exists $a \in A$, $b \in B$, $c \in C$ such that $a + b = c$, is equivalently hard as 3SUM.

proof) It is very easy.

Lower bounds on Geometric Problems

GeomBase

Given n lattice points that lie in either $y = 0, y = 1, y = 2$, determine whether there exists a non-horizontal line that contains at least three points.

Theorem

Geombase is equivalently hard as 3SUM.

proof) $A = \{a | (a, 0) \in S\}, B = \{c | (c, 2) \in S\}, C = \{2b | (b, 1) \in S\}$.
This is a two-way reduction between Geombase and 3SUM'.

This does not seem as a big deal in hindsight, but will play a important role.

Lower bounds on Geometric Problems

Strips

Given n strips and one axis-aligned target rectangle on the plane. Do the strips cover the target rectangle?

Theorem

Strips is 3SUM-hard.

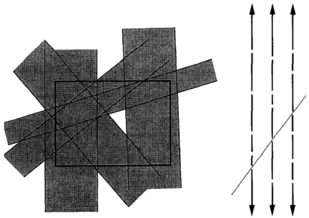

Figure: A Strips instance

Lower bounds on Geometric Problems

Theorem

Strips is 3SUM-hard.

proof) We reduce Strips from Geombase.

- point-line duality : $(a, b) \leftrightarrow y = ax + b$ − preserves intersection.

So we view a Geombase instance as a set of line segments and 6 rays in $\mathbb{R}^2$. Dual of each line segments and rays become a strip or a half-plane, and our goal is to find a point that belongs to neither of the strips or half-planes.

Technical corrections − set the rectangle large enough, and replace half-planes with strips that are large enough to cover the rectangle.

Lower bounds on Geometric Problems

We mention some problems proven 3SUM-hard based on Strips.

Triangle Cover

Given n triangles and one target triangle on the plane. Do the triangles cover the target triangle?

Triangle Measure

Given n triangles on the plane. Compute the area of the union of the triangles.

k-Point Cover

Given n half-planes and $k \in Z$, is there a point covered by at least k half-planes?

Lower bounds on Geometric Problems

There are some unexpected results related to computational biology:

Fixed Angle Chain

A **fixed angle chain** on the plane is a chain of line segments, where at each joint the chain can be flipped to the other direction with same angle.

Fixed Angle Chain

Given a fixed angle chain, the problem of determining whether some flips cause self-intersection is 3SUM-hard.

So this implies that the structure of a polymer cannot be maintained and queried fast – in sublinear time – since answering n queries require $\mathcal{O}(n^{2-\epsilon})$ time.

Lower bounds on Geometric Problems

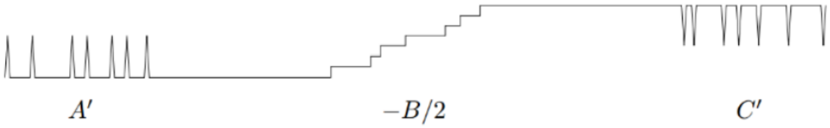

Figure: Reduction from 3SUM' to Fixed Angle Chain

So if 3SUM' is positive, there exists a flip that the teeth align well to make an intersection.

Further results - Dynamic data structures

3SUM is an "algebraic" problem. So the elementary results of 3SUM conjecture is proving lower bounds of "algebraic" problems such as geometry.

So it is very surprising that 3SUM conjecture can be applied to "combinatoric" problems involving graphs or sets. As an example,

Theorem (Pătrașcu, 2010)

Under 3SUM-conjecture, the problem of maintaining a data structure that supports edge insertion and deletion on a directed graph G and queries asking whether there exists a directed path from u to v, requires $\mathcal{O}(n^{0.5-\epsilon})$ time per update and query.

Further results - Dynamic data structures

This kind of "combinatoric" reduction is possible due to the following reduction to a problem with more combinatoric structure:

Convolution 3SUM

Given an array A of n integers, find indices i, j such that $A[i] + A[j] = A[i + j]$.

Theorem (Pătrașcu, 2010)

Convolution 3SUM is 3SUM-hard.

Further results - Dynamic data structures

Then we eventually reduce the "Multiphase problem" from Convolution 3SUM.

Multiphase problem (Dynamic SetDisjointness)

The problem is to construct a dynamic data structure that supports:

1. Given k sets $S_1, \cdots, S_k \subseteq [n]$. Preprocess the data structre in $\mathcal{O}(nk \cdot \tau)$ time.
2. Given a set $T \in [n]$. Update the data structure in $\mathcal{O}(n \cdot \tau)$ time.
3. For given $i \in [k]$, determine whether $S_i \cap T$ is nonempty in $\mathcal{O}(\tau)$ time.

Theorem (Pătrașcu, 2010)

Assuming 3SUM conjecture, if $k = \Theta(n^{2.5})$, the multiphase problem requires $\tau = \Omega(n^{0.5-\epsilon})$.

Further results - Dynamic data structures

Theorem (Pătrașcu, 2010)

Assuming 3SUM conjecture, if $k = \Theta(n^{2.5})$, the multiphase problem requires $\tau = \Omega(n^{0.5-\epsilon})$.

This result can be used to prove polynomial lower bounds for many dynamic problems. For example, the pre-mentioned theorem about dynamic reachability:

Theorem (Pătrașcu, 2010)

Under 3SUM conjecture, data structure for online dynamic reachability requires $\mathcal{O}(n^{0.5-\epsilon})$ time per update and query.

proof) Given a Multiphase problem instance, Add edges $s \rightarrow i$ for all $i \in k$, add an edge $i \rightarrow j$ if S_i contains j, add an edge $j \rightarrow t$ if T contains j. Now ask if there exists a direct path from s to t.

Further results - Dynamic data structures

Theorem (Pătrașcu, 2010)

Convolution 3SUM is 3SUM-hard.

proof) We're given a 3SUM instance, and we assume that Convolution 3SUM can be solved in subquadratic time.

The main idea is to hash the set of integers into an array, preserving the linear structure somehow.

For an odd integer a, we hash an integer x to [first s bits of $(a * x) \bmod 2^w$]. Here w is a word size of a Word RAM model, and s is a constant to be determined later.

Key property of this hashing is the linearity of $(a * x) \bmod 2^w$. Since we take the first s bits, either $h(x) + h(y) = h(x + y)$ or $h(x) + h(y) = h(x + y) + 1$.

Further results - Dynamic data structures

Theorem (Pătrașcu, 2010)

Convolution 3SUM is 3SUM-hard.

Now in the optimistic case where all the element get hashed into different values, we're done. With a careful construction we can solve Convolution 3SUM $\mathcal{O}(1)$ times to solve the 3SUM.

That does not happen all the time, so we have to deal with the collisions. By choosing a uniformly at random, we can view the hashing process as throwing n balls into 2^s bins randomly.

Results from probability theory tells that there are $\mathcal{O}(2^s)$ expected balls that belong to bins with at more than $3n/2^s$ balls. For these elements, we can check in $\mathcal{O}(n)$ time whether they belong to a solution. As a result, we can rule out heavy-loaded bins in $\mathcal{O}(n \cdot 2^s)$ time.

Further results - Dynamic data structures

Theorem (Pătrașcu, 2010)

Convolution 3SUM is 3SUM-hard.

Now each bin has at most $3n/2^s$ balls. We will make $(3n/2^s)^3$ instances of Convolution 3SUM, for each $(i,j,k) \in [3n/2^s]^3$, we use only i-th, j-th, k-th elements for each bins, in total at most $3 \cdot 2^s$ elements.

Now for each instance, every element gets hashed into different values, so we can solve each cases in $\mathcal{O}(n^{2-\epsilon})$ time. So we're done solving the given 3SUM instance, in $\mathcal{O}(n \cdot 2^s + (n/2^s)^3 \cdot n^{2-\epsilon})$ time.

Let $2^s = n^{1-\delta}$ for $\delta < \epsilon/3$, then we have $\mathcal{O}(n^{2-\min(\delta,\epsilon-3\delta)})$, hence we found a subquadratic algorithm for 3SUM, which is a contradiction.

Further results - Dynamic data structures

Then we reduce the Convolution 3SUM into Triangle Listing problem in a graph, which is an "algebraic" problem.

Triangle Listing

Given a (nondirected) graph G, report k triangles.

The proof uses the almost linear hashing from the Convolution 3SUM instance, and build a tripartite graph where the triangles correspond to the indicies i, j such that $h(A[i]) + h(A[j]) = h(A[i+j]) + (0/1)$.

Finally we reduce Triangle Listing to the Multiphase Problem.

Theorem (Pătrașcu, 2010)

Assuming 3SUM conjecture, if $k = \Theta(n^{2.5})$, the multiphase problem requires $\tau = \Omega(n^{0.5-\epsilon})$.

Further results - Polynomial lower bounds

There are many intersting results that I could not introduce in this slide, so I recommend you to read those if you are interested!

Also there are more theory about polynomial lower bound that does not rely on 3SUM conjecture. These are the examples:

- APSP conjecture (Next lecture by koosaga!)
- Triangle detection conjecture
  - Detecting a triangle in a graph G requires $\mathcal{O}(n^{1+\delta-\epsilon})$ time.
- Boolean Matrix Multiplication conjecture
  - Combinatorial BMM requires $\mathcal{O}(n^{3-\epsilon})$ time.
- Orthogonal Vector Conjecture
  - Checking $A \cdot B = 0 \bmod 2$ for two vectors $A, B \in \{0, 1\}^n$ requires $\mathcal{O}(n^{2-\epsilon})$ time.