CMSC 858M: Algorithmic Lower Bounds: Fun with Hardness Proofs
Fall 2020

Instructor: Mohammad T. Hajiaghayi
Scribe: Kiarash Banihashem

1 Overview

Previous parts of the book have focused on computational lower bounds that showed, assuming P≠NP, certain problems cannot be solved in polynomial time. While these methods work for a large class of problems, it is often the case that a problem actually *can* be solved in polynomial time, however the polynomial is of a large degree, e.g, $\mathcal{O}(n^3)$ or $\mathcal{O}(n^4)$. Since the input size of many modern problems can be very large, the practicality of these algorithms is limited. As such, for many of these problems, there have been many attempts to improve the running time of these polynomials to obtain better bounds, with most of these attempts leading to little success. A natural question therefore is whether such improvements are actually possible or are there fundamental limits to the best complexity obtainable for these problems, even if they are all solavable in polynomial time.
This chapter shows that this is indeed the case. Taking a similiar approach as the NP-hardness results, it first introduces the *3SUM* problem and conjectures that it cannot be solved in truely sub-quadratic time. Next, it shows that for a large class of problems admite sub-quadratic reductions from the 3SUM problem. Assuming the 3SUM hardness conjecture is true, this implies that these problems cannot be solved in truely sub-quadratic time either.

2 The 3SUM Problem

This section formulates the 3SUM problem and provides an algorithm that can deterministicly solve this problem in quadratic time by successive refinements of a simple $\mathcal{O}(n^3)$ algorithm.

DEFINITION

Given a set of n integers, the 3SUM problem asks whether any 3 of these integers sum to zero.

END OF DEFINITION
THEOREMTheorem 17.2.1
The 3SUM problem can be solved in $\mathcal{O}(n^2)$ time.
END THEOREM
PROOF

Let A be the original input of n integers. First, we consider a simple $\mathcal{O}(n^2 \log(n))$ algorithm. We observe that three integers A_i, A_j, A_k sum to zero, if and only if $A_i + A_j = -A_k$. We therefore compute all pairwise sums of A, i.e, all possible $A_i + A_j$ and sort them into an array B. Since there are $\mathcal{O}(n^2)$ such pairs, this takes $\mathcal{O}(n^2 \log(n^2)) = \mathcal{O}(n^2 \log(n))$ time. Next, for each integer A_k, we check whether $-A_k$ is in B. Since B is sorted, this takes $\mathcal{O}(n \log(n))$ time for each A_k, leading to an $\mathcal{O}(n^2 \log(n))$ time for checking the entire set A.
To obtain an improved $\mathcal{O}(n^2)$ version, we modify the algorithm such that checking whether $-A_k$ equals $A_i + A_j$ for some i, j takes $\mathcal{O}(n)$ time. To do this, we first sort the array A in $\mathcal{O}(n \log(n))$ time. For each A_k, we initalize two pointers i, j on the first and last element of the array respectively. If $A_i + A_j < -A_k$, we increase the counter i and if $A_i + A_j > -A_k$, we decrease the counter j. We keep doing this until either the pointers corss each other or we find $A_i + A_j = -A_k$. In the latter case, we output YES while in the former case, we proceed to test the next A_k. If no suitable A_k is found, we output NO.

END PROOF

The book then states some improved running times that can be obtained for the problem, both in general and with extra assumptions, which we list below.

1. If the integers are restricted to the interval $[-u, u]$, then the problem can be solced in $\mathcal{O}(n + u \log(n))$ time.

2. In the word-RAM model where $\mathcal{O}(\log(n))$-bit words can be manipulated in constat time, there is a random algorithm for 3SUM with time complexity

$$\mathcal{O}\left(\left(\frac{n \log\log(n)}{\log(n)}\right)^2\right)$$

3. There is a randomized algorithm that takes $\mathcal{O}(\frac{n^2 \log\log(n)}{\log(n)})$ time and a deterministic algorithm that takes $\mathcal{O}(\frac{n^2 (\log\log(n))^{\frac{2}{3}}}{(\log(n))^{\frac{2}{3}}}$ time.

4. There is a deterministic algorithm that runs in $\mathcal{O}(\frac{n^2 (\log\log(n))^{\mathcal{O}(1)}}{\log^2(n)})$ time.

5. There exists a *decision tree* algorithm that can solve 3SUM in $\mathcal{O}(n^{1.5} \log(n))$ time though there are no known algorithms for constructing such a decision tree in subquadratic time.

None of the above algorithms are much better than the $\mathcal{O}(n^2)$ time algorithm More formally, the book defines the class of subquadratic time algorithms as follows.

DEFINITION

An algorithm is called subquadratic if there exists $\epsilon > 0$ such that it runs in $\mathcal{O}(n^{2-\epsilon})$ time.

END OF DEFINITION

3 Definition of 3SUM-Hardness

This section formally states the 3SUM-hardness conjecture and defines the notion of subquadratic reductions. These tools will later be used to show hardness results for a variety of problems.

CONJECTURE

There is no subquadraic algorithm for 3SUM.

END OF CONJECTURE

DEFINITION

Let A, B be two sets or functions. We will write $A \leq_{sq} B$ if any subquadraic algorithm for B can be turned into a subquadratic algorithm for A and write $A \equiv_{sq} B$ if $A \leq_{sq} B$ and $B \leq_{sq} A$.

END OF DEFINITION

DEFINITION

A problem A is 3SUM-hard if 3SUM$\leq_{sq} A$.

END OF DEFINITION

While the notions of 3SUM-hard and NP-hard are similar, the book notes two important differences. Firstly, there is no notion of 3SUM-complete as there is no natural class like NP for 3SUM. Secondly, there is no analogous result to the Cook-Levin Theorem. In other words, the 3SUM problem is assumed to be hard and used as such.

4 Variants of 3SUM

This section of the book introduces three variants of the 3SUM problem and shows that they are 3SUM-hard.

The first result shows that the hardness of the 3SUM problem does not stem from the fact that it allows very large integers and the problem remains hard with restrictions on the size of its input.

THEOREM

The 3SUM problem restricted to inputs in $[-n^3, n^3]$ is 3SUM-hard.

END THEOREM

Next, a variant of 3SUM that effectively restricts the choice of i, j, k is considered

THEOREM

Given a set of integers $a_1, \ldots, a_n$, it is 3SUm-hard to determine whether there is an $i \neq j$ such that $a_{i+j} = a_i + a_j$.

END THEOREM

Another variant is the 3SUM' problem in which a_i, a_j, a_k are chosen from three different arrays of integers.

DEFINITION

Given three sets of n integers A, B, C, the 3SUM' problem asks whether there are $a \in A, b \in B, c \in C$ such that $a + b = c$.

END OF DEFINITION

THEOREM

3SUM'$\equiv_{sq}$3SUM

END OF THEOREM

5 3SUM-hard Problems in Computational Geometry

This section introduces some 3SUM-hard problems from computational geometry.

DEFINITION[GEOMBASE] Given n points in Z^2 with y coordinates in $\{0, 1, 2\}$, determine if there exists a non-horizontal line hitting 3 points of the set. END DEFINITION

DEFINITION[GEOMBASE'] Given n points in Z^2 with y coordinates in $\{0, 1, 2\}$ and a parameter ϵ, View the points as holes in the $y = 0, 1, 2$ lines and enlarge them to be ϵ-long, determine if there is a non-horizontal line going through three of these holes. END DEFINITION

DEFINITION[COLLINEAR] Given n points in Z^2, determine if any three are colinear. END DEFINITION

DEFINITION[CONCURRENT] Given n lines, determine if there is a point on three of these lines. END DEFINITION

THEOREM

1. $3\text{SUM}' \equiv_{sq} \text{GEOMBASE} \leq_{sq} \text{GEOMBASE}'$
2. $3\text{SUM} \leq_{sq} \text{COLLINEAR} \equiv_{sq} \text{CONCURRENT}$

END THEOREM

The book also considers many other problems, some of which we list below.

DEFINITION[Cover Box problem (STRIPS)]

Given a set of n strips and an axis-aligned rectangle, determine if a union of the strips covers the rectangle.

END DEFINITION

DEFINITION[Cover trinangle (TCT)]

Given a set of n triangles, determine if the set of triangles covers the target triangle.

END DEFINITION

DEFINITION[Hole in Uinion (HIU)]

Given a set of n trinangles on the plane which may overlap, does this set have a hole, i.e, a closed region within the union that is not covered by any of the traingles?

END DEFINITION
THEOREM
$\mathsf{GEOMBASE} \leq_{sq} \mathsf{STRIPS} \leq_{sq} \mathsf{TCT} \leq_{sq} \mathsf{HIU}$
END THEOREM

6 Other lower bounds

There are also some graph problems that are 3SUM-hard, as shown by the following two theorems.

THEOREM

Given a weighted graph and a number x, we want to determine if some traingle has weight x.

1. There is an $\mathcal{O}(|E|^{\frac{3}{2}})$ algorithm for this problem.
2. If there is an $\mathcal{O}(|E|^{\frac{3}{2}} - \epsilon)$ algorithm for this problem, then there is an $\mathcal{O}(n^{2-\delta})$ algorithm for 3SUM.

END THEOREM

7 DSUM

This section considers a simple generalization of 3SUM called dSUM.

DEFINITION[dSUM]

Given a set of n integers, are there any d of these integers that sum to 0?

END DEFINITION

The next result shows that given some assumptions, assuming the ETH conjecture is true, then dSUM can not be solved in $\mathcal{O}(n^{o(d)})$.

THEOREM

Let $d \leq n^{0.99}$. If dSUM with numbers of $\mathcal{O}(d \log(n))$ bits can be solved in $n^{o(d)}$ time, then 3SAT can be solved in $2^{o(n)}$ time.

END THEOREM

8 Extra problems

1. [?] show that the 3SUM problem can be reduced to an online version of set disjointness (the *multiphase problem*) and show that the problem requires $n^{\Omega(1)}$ update time. The problem can be reduced to many other dynamic problems.
2. [?] show that assuming the 3SUM-hardness conjecture is true, then st-Sub eonnectivity, st-Reach and bipartite perfect matching problem cannot have algorithms with preprocessing time $\mathcal{O}(m^{4/3-\epsilon})$, amortized update time $\mathcal{O}(m^{\alpha-\epsilon})$ and amortized query time $\mathcal{O}(m^{2/3-\alpha-\epsilon})$.

3. [**?**] consider a weaker conjecture that at least one of 3SUM-hardness, APSP and CNF-SAT holds and obtain
   (a) $n^{3-o(1)}$ lower bounds for the amortized update and query times of dynamic algorithms for single-source reachability, strongly connected components, and MaxFlow.
   (b) an $n^{1.5-o(1)}$ lower bound for computing a set of n st-maximum-flow value s in a directed graph with n nodes and $\widetilde{\mathcal{O}}(n)$ edges
4. [**?**] consider the local alignmant problem where given two input strings and a scoring function on pairs of letters, one is asked to find the substrings of the two input strings that are most similar under the scoring function. They show that assuming 3SUM-hardness is true, there is no algorithm for this problem with running time $\mathcal{O}(n^{2-\epsilon})$.
5. [**?**] consider a variant of monochromatic triangle that can be solved in $\mathcal{O}(n^{1.5})$ time is fine-grained equivalent to 3SUM which further implies that there are no strong improvements possible assuming 3SUM-hardness holds. This result is particularly interseting as it gives fine-grained equivalence between natural problems of different running times.
6. [**?**] show that assuming 3SUM-hardness is true, any static data structure for SetDisjointness that answers queries in constant time must spend $\Omega(N^{2-o(1)})$ time in preprocessing, where N is the size of the set system.
7. [**?**] study the problem of finding the "deepest" point in an arrangement of disks, where the depth of a point denotes the number of disks that contain it. They show this problem is 3SUM-hard.

References

[1] A. Abboud and V. V. Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 434–443. IEEE, 2014.

[2] A. Abboud, V. V. Williams, and O. Weimann. Consequences of faster alignment of sequences. In *International Colloquium on Automata, Languages, and Programming*, pages 39–51. Springer, 2014.

[3] A. Abboud, V. V. Williams, and H. Yu. Matching triangles and basing hardness on an extremely popular conjecture. *SIAM Journal on Computing*, 47(3):1098–1122, 2018.

[4] B. Aronov and S. Har-Peled. On approximating the depth and related problems. *SIAM Journal on Computing*, 38(3):899–921, 2008.

[5] T. Kopelowitz, S. Pettie, and E. Porat. Higher lower bounds from the 3sum conjecture. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 1272–1287. SIAM, 2016.

[6] A. Lincoln, A. Polak, and V. V. Williams. Monochromatic triangles, intermediate matrix products, and convolutions. *arXiv preprint arXiv:2009.14479*, 2020.

[7] M. Patrascu. Towards polynomial lower bounds for dynamic problems. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 603–610, 2010.