1 Lower Bounds on Data Structures via the 3SUM Conjecture

Imagine that you want a data structure for (1) storing a graph with n vertices and m edges, or (2) a collection of m set within a universe of n elements. There are three issues:

- How long will it take to to set up the data structure? This is called *preprocessing*.
- How much space will the data structure need?
- How long will it take to update the data structure? There are many update operations you might allow. For graphs vertex operations (add or delete), or edge operations (add or delete). For sets adding an element, deleting an element, adding a set, deleting a set, merging sets, maintaining the min or max (if the elements are numbers)¿ There are other operations as well.
- How long will it take to answer a query? There are many queries you might be interested in. For graphs reachability, number of (strongly) connected components, others. For sets membership is the the key one, though there are others.
- Assume that you want to make L queries where L is large. It may be that some queries take a long time; however, while doing it you modify the data structure a lot, so that later queries are much faster. We don't want to look at the worst case. We want to say that L queries took $\alpha(n)L$ time. The function $\alpha(n)$ is the *amortized query time*). There are many queries you might be interested in.
- One can also look at *amortized update time*.

In the context of data structures, *fast* is polylog (or even $O(1)$) and *slow* is n^δ. Often there is a tradeoff.

Example 1 *We look at three data structures for a set where (1) the allowed updates are insert and delete, (2) the allowed query is just membership. Assume there are roughly n items and they are numbers. (It may get bigger or smaller as elements are inserted and deleted.)*

1. *Store the numbers in a sorted array. Any insert or delete takes* $O(n)$ *since you may have to move many elements. Queries are* $O(\log n)$ *time by binary search.*

2. *Store the numbers in a linked list. Inserts takes* $O(1)$*. Deletes may depend on how you know what to delete. If you have to search for it then this takes* $O(n)$*. If you somehow have a pointer to the item you want to delete then* $O(1)$*. Queries take* $O(n)$ *since you may need to traverse* $O(n)$ *elements to get to the one you want. Worse–if the element you want is not in the list it will take* $\Theta(n)$ *steps.*

3. *Store the numbers in a binary tree. There are many ways to do this which we do not get into. Updates take* $O(\log n)$ *and queries take* $O(\log n)$*.*

We list some of the known results. They all assume the 3SUM conjecture.

1. Patrascu [7] was the first person to use the 3SUM conjecture to get lower bounds on dynamic data structures. We give one of his results:

   Dynamic Reachability The problem is to find a data structure for directed graphs that allows (1) updates: insertion and deletion of edges (but not vertices), and (2) queries: given vertices u, v determines if there is a directed path from u to v. There exists $\delta > 0$ such that, for any data structure for this problem, either updates or queries take $\Omega(n^{\delta})$. All of the later papers build on this paper.

2. Abboud and V. V. Williams [1] considered many problems where the 3SUM conjecture (or other assumptions) were used to get lower bounds on data structures. We give two of the problems they considered which have the same lower bound assuming the 3SUM conjecture.

   st-Reachability The problem is to find a data structure for a directed graphs and two nodes s, t that allows (1) updates: insertion and deletion of edges (but not vertices), and (2) queries: is there a directed path from s to t?

   Bipartite Perfect Matching The problem is to find a data structure for an undirected bipartite graphs that allows (1) updates: insertion and deletion of edges (but not vertices), and (2) queries: is there a perfect matching?

For both problems the following holds: For all α there is no data structure that does updates in $O(m^{\alpha})$ and queries in $O(m^{2/3-\alpha})$.

3. Kopelowitz et al. [5] considered many problems. We consider one of them.

   The Static Set Disjointness Problem. The Universe U has n elements. (1) store subsets of U statically so there are no updates, (2) queries: given two sets, are they disjoint?

   They show that if query time is $O(1)$ then preprocessing must take $\Omega(n^{2-o(1)})$,

References

[1] A. Abboud and V. V. Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 434–443. IEEE Computer Society, 2014. https://doi.org/10.1109/FOCS.2014.53.

[2] A. Abboud, V. V. Williams, and O. Weimann. Consequences of faster alignment of sequences. In J. Esparza, P. Fraigniaud, T. Husfeldt, and E. Koutsoupias, editors, *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, volume 8572 of *Lecture Notes in Computer Science*, pages 39–51. Springer, 2014. https://people.csail.mit.edu/virgi/hardstrings.pdf.

[3] A. Abboud, V. V. Williams, and H. Yu. Matching triangles and basing hardness on an extremely popular conjecture. *SIAM J. Comput.*, 47(3):1098–1122, 2018. https://doi.org/10.1137/15M1050987.

[4] B. Aronov and S. Har-Peled. On approximating the depth and related problems. *SIAM J. Comput.*, 38(3):899–921, 2008. https://doi.org/10.1137/060669474.

[5] T. Kopelowitz, S. Pettie, and E. Porat. Higher lower bounds from the 3sum conjecture. In R. Krauthgamer, editor, *Proceedings of the*

Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016, pages 1272–1287. SIAM, 2016.
https://doi.org/10.1137/1.9781611974331.ch89.

[6] A. Lincoln, A. Polak, and V. V. Williams. Monochromatic triangles, intermediate matrix products, and convolutions. In T. Vidick, editor, *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, volume 151 of *LIPIcs*, pages 53:1–53:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.
https://doi.org/10.4230/LIPIcs.ITCS.2020.53.

[7] M. Patrascu. Towards polynomial lower bounds for dynamic problems. In L. J. Schulman, editor, *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 603–610. ACM, 2010.
https://doi.org/10.1145/1806689.1806772.