Google Research Award Proposal:

Data Structures

Erik D. Demaine (PI)

Mihai Pătraşcu (PhD Student)

Data structures are essential components of computer systems in general and Google in particular. We believe this area of research is in an auspicious position where practical and theoretical goals are well aligned, implying that deep algorithmic ideas can also have significant practical impact.

We exemplify with a few examples from our past research, which address problems of universal value, and should have important applications in real systems.

Cache-oblivious B-trees: B-trees are a fundamental tool for representing large sets of data in external memory. But what *is* "external memory"? Modern computers have complicated memory hierarchies, including L1 cache, L2 cache, main memory, disk, and often network storage. Even if one decides to concentrate on one level of the hierarchy, choosing the optimal branching factor involves nontrivial tuning.

A surprising, clean alternative is to design a B-tree which works in the optimal $O(\log_B n)$ time without knowing the memory block size B! Then the B-tree will work optimally on all levels of the memory hierarchy simultaneously. Our initial paper [BDFC05] showing that this is possible has been very influential in the further study of cache-obliviousness.

Bloomier filters: Suppose we want to represent a set S of items, and answer queries of the form "is $x \in S$?". If we can tolerate some false positives, a Bloom filter can solve the problem with less memory than it actually takes to store S. This is important in many applications, making Bloom filters a well-known and widely deployed data structure.

Quite often, however, we are not merely interested in membership ($x \in S$), but want to return some associated $data[x]$. Do there exist "Bloomier filters" which can retrieve associated data without representing S? Quite surprisingly, the answer is *yes*, even when elements can be inserted and deleted into S dynamically [DMPP06]. Interestingly, queries are deterministic and always return the correct $data[x]$ for any $x \in S$, even though the data structure does not have space to remember the current set S.

Range reporting in 1D: Suppose we must represent a set S of integers, subject to the query: report one value from an interval $[a, b]$, or determine that $S \cap [a, b] = \emptyset$. Note that after we find one value from the given range, we can trivially find all of them by walking a linked list representing S.

Interestingly, it is possible to answer the query faster than searching for where a or b fit in S [MPP05]. One can build a data structure of linear size that answers a query in *constant* time, *regardless* of the number of bits in the integers!

While the practicality of cache-oblivious B-trees has been investigated with encouraging results [BFCK06, LFN02], the more recent results mentioned above have not yet been tested in practice.

Fortunately, these are based on clean algorithmic ideas, and we believe that they will have a positive impact in actual system.

Google seems like an ideal testing ground for innovative technical ideas, like the above. We would greatly enjoy interacting with Google engineers to identify applications of such ideas, and disseminate cutting-edge data structural research for possible inclusion into actual systems.

In addition, we would be very interested in addressing new fundamental problems of particular relevance to Google. In recognizing such problems, interacting with Google engineers could be invaluable. Even in past work, though, we have considered several problems with immediate connections to search engines:

Representing tries: Tries are the standard representation of strings for the purpose of searching, but direct strategies for implementing them become inefficient for large data. We have been investigating optimal ways to store tries in external memory [ABD$^+$], as well as ways to represent tries using less space [BDM$^+$05].

Computing set intersections: We have developed *adaptive* strategies for computing set intersections [DLOM00], motivated by searching for Web pages that simultaneously contain several keywords. We have followed up on the theoretical algorithms with experimental work [DLOM01] on the problem, using real data from a small web crawl, and actual user queries provided by Monika Henzinger from Google.

Dictionaries on parallel disks: We have shown that if a dictionary is distributed on a number of parallel disks (a natural setup at Google), dictionary operations can be supported *deterministically* in constant time [BHP$^+$06]. This makes it possible to implement huge file systems using hashing (as opposed to slower B-trees), but without introducing randomization.

Finally, we believe an integral part of academic research is to establish lower bounds on how efficiently certain problems can be solved. While not of direct value to systems, this can have significant influence in shaping practical thought, similar to the influence of NP-completeness on work in optimization. While our work in this area is quite broad, we mention a few problems of significant relevance to Google:

Index size: We were the first to show a lower bound on the size of the index needed for efficient text retrieval [DLO03].

Searching with wild cards: We have shown the currently best lower bound on searching with wild cards [PT06], though this is not yet close to the known upper bounds.

Dimensionality reduction: Clustering and similarity-search problems on high-dimensional data (of direct applications to approximate searching, as well as many applications in machine learning) are often solved by projecting the problem on a random subspace of lower dimension, hoping that relevant distances are preserved. We have shown [AIP06] that this dimensionality reduction technique is the optimal way to solve several of the central problems where it is commonly applied.

We look forward to collaborating with Google to identify practically important data structural problems, developing algorithmic solutions to solve them, and establishing optimality through lower bounds. We believe that our arsenal of techniques for data structure design and lower bounds will prove useful for the problems at hand, as our track record should show. We have previously collaborated with S. Muthukrishnan (now a researcher at Google New York) on algorithmic problems, and would be happy to initiate this collaboration through him.

References

[ABD+] Stephen Alstrup, Michael A. Bender, Erik D. Demaine, Martin Farach-Colton, Theis Rauhe, and Mikkel Thorup. Efficient tree layout in a multilevel memory hierarchy. *Algorithmica*. To appear. arXiv:cs.DS/0211010.

[AIP06] Alexandr Andoni, Piotr Indyk, and Mihai Pătraşcu. On the optimality of the dimensionality reduction method. In *Proceedings of the 47th IEEE Symposium on Foundations of Computer Science*, pages 449–458, 2006.

[BDFC05] Michael A. Bender, Erik D. Demaine, and Martin Farach-Colton. Cache-oblivious b-trees. *SIAM Journal on Computing*, 35(2):341–358, 2005.

[BDM+05] David Benoit, Erik D. Demaine, J. Ian Munro, Rajeev Raman, Venkatesh Raman, and S. Srinivasa Rao. Representing trees of higher degree. *Algorithmica*, 43(4):275–292, December 2005.

[BFCK06] Michael A. Bender, Martin Farach-Colton, and Bradley Kuszmaul. Cache-oblivious string B-trees. In *Proceedings of the 25th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 233–242, 2006.

[BHP+06] Mette Berger, Esben Rune Hansen, Rasmus Pagh, Mihai Pătraşcu, Milan Ružić, and Peter Tiedemann. Deterministic load balancing and dictionaries in the parallel disk model. In *Proceedings of the 18th Annual ACM Symposium on Parallelism in Algorithms and Architectures*, pages 299–307, Cambridge, MA, 2006.

[DLO03] Erik D. Demaine and Alejandro López-Ortiz. A linear lower bound on index size for text retrieval. *Journal of Algorithms*, 48(1):2–15, August 2003.

[DLOM00] Erik D. Demaine, Alejandro López-Ortiz, and J. Ian Munro. Adaptive set intersections, unions, and differences. In *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 743–752, San Francisco, California, January 2000.

[DLOM01] Erik D. Demaine, Alejandro López-Ortiz, and J. Ian Munro. Experiments on adaptive set intersections for text retrieval systems. In *Proceedings of the 3rd Workshop on Algorithm Engineering and Experiments*, volume 2153 of *Lecture Notes in Computer Science*, pages 91–104, Washington, DC, January 2001.

[DMPP06] Erik D. Demaine, Friedhelm Meyer auf der Heide, Rasmus Pagh, and Mihai Pătraşcu. De dictionariis dynamicis pauco spatio utentibus (*lat.* on dynamic dictionaries using little space). In *Proceedings of the 7th Latin American Symposium on Theoretical Informatics (LATIN 2006)*, pages 349–361, Valdivia, Chile, March 20–24 2006.

[LFN02] Richard E. Ladner, Ray Fortna, and Bao-Hoang Nguyen. A comparison of cache aware and cache oblivious static search trees using program instrumentation. In *Experimental Algorithmics: From Algorithm Design to Robust and Efficient Software*, volume 2547 of *Lecture Notes in Computer Science*, pages 78–92, 2002.

[MPP05] Christian Worm Mortensen, Rasmus Pagh, and Mihai Pătraşcu. On dynamic range reporting in one dimension. In *Proceedings of the 37th ACM Symposium on Theory of Computing*, pages 104–111, 2005.

[PT06] Mihai Pătraşcu and Mikkel Thorup. Higher lower bounds for near-neighbor and further rich problems. In *Proceedings of the 47th IEEE Symposium on Foundations of Computer Science*, pages 646–654, 2006.