CS 361A - Advanced Data Structures and Algorithms

Autumn Quarter, 2005-06
Rajeev Motwani

Reading List on Data Structures

Hashing and Implicit Data Structures

The following is a list of papers which are relevant to the topics discussed in the first few lectures. I do not have the time to present all of these in the class – that would require a whole course in itself! I am also providing brief annotations.

1 Cell Probe Model

1. A. Yao, "Should Tables Be Sorted?," *Journal of the ACM*, v 28 (1981), pp. 615–628.
   This paper introduced the cell probe model and raised the basic issues studied in the following papers.

2. F. Fich and P.B. Miltersen, "Tables should be sorted (on random access machines)," Brics Report RS-95-26, May 1995.
   The title says it all.

3. R.E. Tarjan and A. Yao, "Storing a Sparse Table," *Communications of the ACM*, v 22 (1979), pp. 606–611.
   They gave a scheme using linear space which would guarantee constant time search provided the size of the key space M is polynomially bounded in the size of the data set S.

4. J.L. Carter and M.N. Wegman, "Universal Classes of Hash Functions," *Journal of Computer and System Sciences*, v 18 (1979), pp. 143–154.
   This introduced the notion of 2-universal hash functions.

5. M.L. Fredman, J. Komlos and E. Szemeredi, "Storing a Sparse Table with O(1) Worst Case Access Time," *Journal of the ACM*, v 31 (1984), pp. 538–544.
   They gave a solution for static tables which uses only linear space and provides a constant time access.

6. M. Dietzfelbinger, A. Karlin, K. Mehlhorn, F. Meyer auf der Heide, H. Rohnert and R.E. Tarjan, "Dynamic Perfect Hashing: Upper and Lower Bounds," *IEEE Symposium on Foundations of Computer Science* (1988), pp. 524–531.
   This extends the results of Fredman-Komlos-Szemeredi to dynamic tables but time required for an update (insertion/deletion) is constant in the amortized *and* expected sense. They also prove lower bounds showing that the worst-case amortized time for an update must be at least logarithmic, unless you are willing to increase the search time.

7. M.L. Fredman and J. Komlos, "On the size of separating systems and families of perfect hash functions," *SIAM Journal on Computing*, v 5 (1984), pp. 61–68.
   They provide tight bounds on the optimal sizes of separating systems and perfect hash function families.

8. M. Ajtai, M. Fredman and J. Komlos, "Hash Functions for Priority Queues," *Information and Control*, v 63 (1984), pp. 217–225.
   They use hash functions to implement priority queue operations in Yao's cell probe model, obtaining a constant worst-case time bound on the desired operations. They also analyze the size of hash families for this problem and contrast it with bounds known for perfect hash functions.

9. A. Fiat, M. Naor, J.P. Schmidt and A. Seigel,"Non-Oblivious Hashing," *ACM Symposium on the Theory of Computing* (1988), pp. 367–376.
   They improve the F-K-S result by reducing the extra memory required down to only a logarithmic number of bits. Also, a scheme is provided which gives constant time search with high probability without using any extra space. Finally, they extend these results to multi-key hashing.

10. M.L. Fredman and M.E. Saks,"The Cell Probe Complexity of Dynamic Data Structures," *ACM Symposium on the Theory of Computing* (1989), pp. 345–354.
    Lower bounds are proved on the time required for implementing accesses to dynamic data structures, using the cell probe model of Yao.

11. M. Ajtai, "A Lower Bound for Finding Predecessors in Yao's Cell Probe Model," *Combinatorica*, v 8 (1988), pp. 235–247.

12. J. Gil, F. Meyer auf der Heide and A. Wigderson, "Not All Keys Can be Hashed in Constant Time," *ACM Symposium on Theory of Computing* (1990), pp. 244–253.
    Under a slightly different model of (parallel) hashing they show that although the average time for hashing a key is constant, some key will require logarithmic time. The main reason I am including this paper is that it provides references which will lead you into the area of hashing for parallel computation and PRAM simulation.

2 Implicit Data Structures

An interesting area of research has been the study of implicit data structures. We have had only a brief glimpse of the kind of techniques used for constructing implicit data structures. An interesting version of implicit data structures involves storing items with multiple keys. The following papers will help you to learn more about this area if you are interested.

1. J.I. Munro and H. Suwanda, "Implicit Data Structures for Fast Search and Update," *Journal of Computer and System Sciences*, v 21 (1980), pp. 236–250.

2. J.I. Munro, "An Implicit Data Structure Supporting Insertion, Deletion and Search in $O(\log^2 n)$ Time," *Journal of Computer and System Sciences*, v 33 (1986), pp. 66-74.

3. J.I. Munro, "Searching a Two Key Table Under a Single Key," *ACM Symposium on Theory of Computing* (1987), pp. 383–387.

4. A. Fiat, M. Naor, A.A. Schaffer, J.P. Schmidt and A. Seigel, "Storing and Searching a Multikey Table,"*ACM Symposium on Theory of Computing* (1988), pp. 344-353.

5. A. Fiat and M. Naor, "Implicit O(1) Probe Search," *ACM Symposium on Theory of Computing* (1989), pp. 336-344.

6. A. Borodin, F.E. Fich, F. Meyer auf der Heide, E. Upfal and A. Wigderson, "Tradeoff Between Search and Update Time for the Implicit Dictionary Problem," *Theoretical Computer Science*, v 58 (1988), pp. 57–68.

7. P. Derome, "A New Search Time Update Time Tradeoff for the Implicit Dictionary,".

3 Hash Functions

The following papers study the existence of various kinds of hash functions and analyze their properties. This is by no means a complete list. My intent is to provide you with a starting point if you wish to delve further into this topic.

1. R. Sprugnoli, "Perfect Hashing Functions: A Single Probe Retrieving Method for Static Sets," *Communications of the ACM*, v 20 (1977), pp. 841–850.

2. G. Jaeschke, "Reciprocal Hashing: A Method for Generating Minimal Perfect Hashing Functions," *Communications of the ACM*, v 24 (1981), pp. 829–833.

3. K. Mehlhorn, "On the Program size of Perfect and Universal Hash functions," *IEEE Symposium on Foundations of Computer Science* (1982), pp. 170–175.

4. H.G. Mairson, "The Program Complexity of Searching a Table," *IEEE Symposium on Foundations of Computer Science* (1983), pp. 40–47.