Basic Research in Computer Science

BRICS RS-99-50 Brodal & Venkatesh: Improved Bounds for Dictionary Look-up with One Error

Improved Bounds for Dictionary Look-up with One Error

Gerth Stølting Brodal
Srinivasan Venkatesh

BRICS Report Series **RS-99-50**

ISSN 0909-0878 **December 1999**

**See back inner page for a list of recent BRICS Report Series publications.
Copies may be obtained by contacting:**

**BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK–8000 Aarhus C
Denmark
Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk**

BRICS publications are in general accessible through the World Wide Web and anonymous FTP through these URLs:

`http://www.brics.dk`
`ftp://ftp.brics.dk`
This document in subdirectory `RS/99/50/`

Improved Bounds for Dictionary Look-up with One Error

Gerth Stølting Brodal* Srinivasan Venkatesh†

December, 1999

Abstract

Given a dictionary S of n binary strings each of length m, we consider the problem of designing a data structure for S that supports *d-queries*; given a binary query string q of length m, a d-query reports if there exists a string in S within Hamming distance d of q. We construct a data structure for the case $d = 1$, that requires space $O(n \log m)$ and has query time $O(1)$ in a cell probe model with word size m. This generalizes and improves the previous bounds of Yao and Yao for the problem in the bit probe model.

Keywords: Data Structures, Dictionaries, Hashing, Hamming Distance

1 Introduction

Minsky and Papert in 1969 posed the following problem, that has remained a challenge in data structure design [8].

> Let S be a set of n binary strings of length m each. We want to construct a data structure for S that supports fast d-queries; that is, given a binary query string q, determine if there is a string in S within Hamming distance d of q.

*BRICS (Basic Research in Computer Science, Center of the Danish National Research Foundation), Department of Computer Science, University of Aarhus, DK-8000 Århus C, Denmark, email: gerth@brics.dk.

†School of Technology and Computer Science, Tata Institute of Fundamental Research, Mumbai 400005, India, email: venkat@tcs.tifr.res.in. This work was done while visiting BRICS at the University of Aarhus.

To date, no efficient solutions are known for this problem for arbitrary n, m and d. Dolev *et al.* [2, 3] and Greene, Parnas and Yao [5] made some progress on the problem for the case when d is large. Manber and Wu [7] considered applications to password security and spellchecking of large files.

The theoretical study of the problem for small d was started by Yao and Yao [11], who considered the case $d = 1$ and presented a scheme in the bit probe model that uses space $O(mn \log m)$ bits and queries requiring $O(m \log \log n)$ bit probes. Brodal and Gąsieniec [1] considered the problem for the case $d = 1$ in the standard unit-cost RAM model with logarithmic word size, and presented a data structure of size $O(mn)$ words supporting 1-queries in $O(m)$ memory accesses.

In this paper, we also consider the case $d = 1$. We give a scheme proving the following result.

Theorem 1 *In a cell probe model with word size m, a data structure exists that achieves query time $O(1)$ using space $O(n \log m)$.*

Translated to the bit probe model, this yields a query time $O(m)$ and space $O(mn \log m)$ scheme. Our scheme improves the query time of Yao and Yao by a factor $O(\log \log n)$. It is also simpler than the scheme given by Yao and Yao in [11].

The model of computation we use is the *cell probe* model with word size m defined in [10]. In this model, each word is assumed to be m bits long. The *space* used by a scheme is measured as the number of words used by the storing scheme. The query algorithm answers queries by making adaptive cell probes and *time* is counted as the number of cell probes made by the query algorithm. The bit probe model is a cell probe model with word size 1.

2 The scheme

To describe our scheme, we will make use of the following results of Fredman, Komlós and Szemeredi [4] and Schmidt and Siegel [9].

Theorem 2 (Fredman, Komlós, and Szemerédi) *Given a set S of n binary strings each of m bits in a cell probe model with word size m, there exists a data structure using $O(n)$ cells that supports membership queries of S in time $O(1)$.*

In the following we refer to the data structure of Fredman, Komlós, and Szemerédi as a FKS data structure.

Definition 1 *A function* $h : \{0, 1, \ldots, \ell - 1\} \rightarrow \{0, 1, \ldots, k - 1\}$ *is* perfect *for* $S \subseteq \{0, 1, \ldots, \ell - 1\}$ *if* h *is 1-1 on* S*. A family* H *is an* (ℓ, n, k)*-family of perfect hash functions if for all* $S \subseteq \{0, 1, \ldots, \ell - 1\}$ *of size* n*, there is an* $h \in H$ *that is perfect for* S*.*

The question of representing efficiently families of perfect hash functions is well studied. Schmidt and Siegel [9] proved the following result in the standard RAM model augmented with multiplicative arithmetic. In this model, a memory access of an $O(m)$ bit word takes unit time.

Theorem 3 (Schmidt and Siegel) *There is a* $(2^m, n, O(n))$*-family of perfect hash functions* H *such that any* $h \in H$ *can be represented in* $\Theta(n + \log m)$ *bits and evaluated in* $O(1)$ *time.*

Though we use the result of Schmidt and Siegel, we note that for our purposes, it suffices to use the simpler scheme of Jacobs and van Emde Boas [6] which requires $O(n \log \log n + \log m)$ bits. Our scheme is based on the following simple idea: If the query string exactly matches with one of the strings in S, then a FKS data structure can be used to check this efficiently. If the query string differs from some string in S by a single bit, then it is sufficient to find such a bit position. We use the Schmidt-Siegel scheme to do this. Given the bit position, we can then flip this bit in the query string and use the FKS data structure for an exact search to verify that the query is within Hamming distance 1 of a string in S.

Let $N(S)$ denote the set of all strings at Hamming distance one of a string in S. Note that $|N(S)| \leq mn$. For any string $x \in N(S)$, we denote by $index(x)$, the position of the bit that was flipped in a string from S to obtain x. If there is more than one choice for $index(x)$, we choose one arbitrarily.

Storing scheme for a set S.

1. Construct a FKS data structure for S.
2. Using the Schmidt-Siegel scheme (Theorem 3), construct a perfect hash function h for $N(S)$ and store h.
3. Construct a look-up table T of size $O(|N(S)|)$ where each entry is $\lceil \log m \rceil$ bits. For $x \in N(S)$, let $T[h[x]] = index(x)$.

Query scheme for a query string q.

1. Check if $q \in S$ using the FKS data structure for S. If $q \in S$, report *yes*.

2. Otherwise, evaluate $h(q)$ and let $i = T[h(q)]$.

3. Flip the ith bit of q and report *yes* if the new string is in S using the FKS structure for S. Otherwise report *no*.

That the scheme works correctly follows from the discussion above. From Theorem 2 and Theorem 3 it follows that in a cell probe model with word size m, the scheme described above uses $O(n \log m)$ cells, and answers queries in $O(1)$ cell probes, and Theorem 1 follows.

References

[1] G. S. Brodal and L. Gąsieniec. Approximate dictionary queries. In *Proc. 7th Combinatorial Pattern Matching*, volume 1075 of *Lecture Notes in Computer Science*, pages 65–74. Springer Verlag, Berlin, 1996.

[2] D. Dolev, Y. Harari, N. Linial, N. Nisan, and M. Parnas. Neighborhood preserving hashing and approximate queries. In *Proc. 5th ACM-SIAM Symposium on Discrete Algorithms*, pages 251–259, 1994.

[3] D. Dolev, Y. Harari, and M. Parnas. Finding the neighborhood of a query in a dictionary. In *Proc. 2nd Israel Symposium on Theory of Computing and Systems*, pages 33–42, 1993.

[4] M. L. Fredman, J. Komlós, and E. Szemerédi. Storing a sparse table with $O(1)$ worst case access time. *Journal of the Association for Computing Machinery*, 31(3):538–544, 1984.

[5] D. Greene, M. Parnas, and F. Yao. Multi-index hashing for information retrieval. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 722–731. IEEE Computer Society Press, 1994.

[6] C. T. M. Jacobs and P. van Emde Boas. Two results on Tables. *Information Processing Letters*, 22(1):43–48, 1986.

[7] U. Manber and S. Wu. An algorithm for approximate membership checking with application to password security. *Information Processing Letters*, 50(4):191–197, 1994.

[8] M. Minsky and S. Papert. *Perceptrons*. MIT Press, Cambridge, Mass., 1969.

[9] J. P. Schmidt and A. Siegel. The spatial complexity of oblivious k-probe hash functions. *SIAM Journal on Computing*, 19(5):775–786, 1990.

[10] A. C. C. Yao. Should tables be sorted? *Journal of the ACM*, 28(3):615–628, 1981.

[11] A. C. Yao and F. F. Yao. Dictionary look-up with one error. *Journal of Algorithms*, 25(1):194–202, 1997.

Recent BRICS Report Series Publications

RS-99-50 **Gerth Stølting Brodal and Srinivasan Venkatesh.** ***Improved Bounds for Dictionary Look-up with One Error*****. December 1999. 5 pp.**

RS-99-49 **Alexander A. Ageev and Maxim I. Sviridenko.** ***An Approximation Algorithm for Hypergraph Max k-Cut with Given Sizes of Parts*****. December 1999. 12 pp.**

RS-99-48 **Rasmus Pagh.** ***Faster Deterministic Dictionaries*****. December 1999. 14 pp. To appear in** ***The Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*****, SODA '00 Proceedings, 2000.**

RS-99-47 **Peter Bro Miltersen and Vinodchandran N. Variyam.** ***Derandomizing Arthur-Merlin Games using Hitting Sets*****. December 1999. 21 pp. Appears in Beame, editor,** ***40th Annual Symposium on Foundations of Computer Science*****, FOCS '99 Proceedings, 1999, pages 71–80.**

RS-99-46 **Peter Bro Miltersen, Vinodchandran N. Variyam, and Osamu Watanabe.** ***Super-Polynomial Versus Half-Exponential Circuit Size in the Exponential Hierarchy*****. December 1999. 14 pp. Appears in Asano, Imai, Lee, Nakano and Tokuyama, editors,** ***Computing and Combinatorics: 5th Annual International Conference*****, COCOON '99 Proceedings, LNCS 1627, 1999, pages 210–220.**

RS-99-45 **Torben Amtoft.** ***Partial Evaluation for Designing Efficient Algorithms—A Case Study*****. December 1999.**