Dictionary Look-Up with Small Errors

Andrew C. Yao* and Frances F. Yao†

Abstract

Let \( W \) be a set of \( n \) binary strings of length \( m \) each. We are interested in designing data structures for \( W \) that can answer \( d \)-queries quickly, that is, given a binary string \( \alpha \), decide whether there is any member of \( W \) within Hamming distance \( d \) of \( \alpha \). This problem, originally raised by Minsky and Papert [MP], remains a challenge in data structure design. In this paper, we make an initial effort towards a theoretical study of the small \( d \) case. Our main result is a data structure that achieves \( O(m \log \log n) \) query time with \( O(nm \log m) \) space for the \( d = 1 \) case.

1 Introduction

Let \( W \) be a set of \( n \) binary strings of length \( m \) each. We are interested in designing data structures for \( W \) that can answer \( d \)-queries quickly, that is,

\[
\text{Given a binary string } \alpha, \text{ decide whether there is any member of } W \text{ within Hamming distance } d \text{ of } \alpha.
\]

Many variants of this problem are possible; we focus on the decision problem here to simplify some of the details. Note that this data structure problem is related to but not the same as the approximate string matching problem discussed in recent literature (e.g. see Galil and Park [GP], Landau and Vishkin [LV], Ukkonen [U], Tarhio and Wood [TW], Ukkonen and Wood [UW]), where the task is to decide if an input string occurs as an approximate substring of another input string. The \( d \)-query problem we consider here was originally raised by Minsky and Papert [MP], and it has remained a challenge in data structure design. There has been renewed interest in this problem recently because of applications to DNA mapping and document retrieval. In Dolev et

*Department of Computer Science, Princeton University, Princeton, New Jersey 08544. This author was supported in part by the National Science Foundation under grant CCR-9301430.

†Xerox Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, California 94304.
al ([DHLNP][DHP]) and in Greene, Parnas and Yao [GPY], some progress was made for the case when $d$ is relatively large. Manber and Wu [MW] considered the $d = 1$ case from a more practical standpoint. In this paper, we make an initial effort towards a theoretical study of the small $d$ case, focusing on the case $d = 1$.

One can formulate the above question in a general bitwise theoretical model, and ask what space/time bounds are achievable. For the standard dictionary problem, i.e., exact queries with $d = 0$, there is a solution with optimal space $O(nm)$ and optimal time $O(m)$. This is an immediate consequence of a result by Fredman, Komlós, and Szemerédi [FKS] when translated to the bitwise model. (We will refer to this data structure as the FKS-dictionary.) Thus, $d$-queries in general can be solved in optimal space $O(nm)$ by allowing $O\left(\binom{m}{d} \right) m$ query time to exhaustively search for $\binom{m}{d}$ exact queries. At the other extreme of the space/time spectrum, optimal query time $O(m)$ is achievable with space $O(n m \binom{m}{d})$. We will be interested in nontrivial techniques away from these extremes.

The cases of small $d$ and large $d$ seem quite distinct in nature, and may require different techniques for their solutions. In the former case, even $d = 1$ already poses a rather challenging problem. In this paper, we make an initial effort in establishing some nontrivial bounds for this case. (In Manber and Wu [MW], a 1-query is first transformed into $O(m)$ exact queries, and then solved by hashing techniques.) Our main result is a data structure that achieves $O(m \log \log n)$ query time with $O(nm \log m)$ space for the $d = 1$ case.

We consider a bitwise complexity model as suggested in Minsky and Papert [MP] and Elias [E] (or, equivalently, the cell-probe model in Yao [Y] with word size 1). A data structure for $W$ consists of a binary string $D_W$ called the dictionary, and a decision tree $T_{W,\alpha}$ for every $d$-query $\alpha$. Each internal node of $T_{W,\alpha}$ branches according to the outcome of a question of the form “$D_W[j] =$?”, while each leaf contains a yes or no answer to the $d$-query $\alpha$. Such a data structure is said to use space $s$ and query time $t$ for $W$, if $s$ is the size of the dictionary $D_W$ and $t$ is the maximum height of $T_{W,\alpha}$ for any $\alpha$. The maximum of $s$ and $t$ over all sets $W$ (with $n$ binary strings of length $m$) defines the space and time complexity of the data structure, denoted by $s(n,m)$ and $t(n,m)$ respectively. Our main result is as follows.

**Theorem 1** There is a data structure for the $d$-query problem that achieves query time $O(m \log \log n)$ and space $O(nm \log m)$ for the $d = 1$ case.

The next two sections of this paper give a proof for the above theorem. The last section lists some open problems.