Implicit $O(1)$ Probe Search *

Amos Fiat
Department of Computer Science
Tel-Aviv University
Tel-Aviv, Israel

Moni Naor
IBM Research Division
Almaden Research Center
650 Harry Rd
San Jose, CA 95120

Abstract

Given a set of n elements from the domain $\{1, \ldots, m\}$, we investigate how to arrange them in a table of size n, so that searching for an element in the table can be done in constant time. Yao ("Should Tables be Sorted", JACM, 28(1981) pp. 615-628) has shown that this cannot be done when the domain is sufficiently large as a function of n.

We give a constructive solution when the domain m is polynomial in n, the number of elements, and give a nonconstructive proof for m no larger than exponential in poly(n). We improve upon a result of Yao and give better bounds on the maximum m for which implicit $O(1)$ probe search can be done. We achieve our results by showing the tight relationship between hashing and certain encoding problems which we call rainbows.

Key words: hashing, perfect hashing, spatial complexity, Ramsey Theory, randomness in computation

AMS(MOS) subject classification: 68P05, 68P10, 68Q05, 68R05, 68R10

*Most of this work was performed while both authors were at UC Berkeley. The work of the first author was supported by a Weizmann Postdoctoral Fellowship and by NSF Grants DCR 84-11954 and DCR 85-13926. The work of the second author was supported by NSF Grants DCR 85-13926 and CCR 88-13632.

1 Introduction

The problem addressed in this paper is searching a full table: A set $S \subset \{1,\ldots,m\}$ of size n is to be stored in a table T of size n, where every table entry holds a single element of S. Given $x \in \{1,\ldots,m\}$ the goal is to locate x in the table or to indicate that $x \notin S$, while probing the table as few times as possible. We assume that n and m are known to the searcher.

Yao [8] has shown that if no storage is available in addition to the table T, then there is no table organization that enables an element to be located in less than $\log n$ probes. We refer to a table organization that requires no additional storage as an *implicit* scheme. Yao's proof assumes that the domain size m is much larger than the number of elements n. This immediately raises the following two questions:

1. For what values of m (relative to n) does an implicit $O(1)$ probe search scheme exist?

2. Given that an implicit scheme does not exist, how much additional storage is required to ensure $O(1)$ search?

In [4] Fiat, Naor, Schmidt and Siegel show that if $m = O(n)$ then search can be performed in $O(1)$ time without any additional storage. As for the second question, Fredman, Komlós and Szemerédi [5] show that one probe search can be performed with $O(n\sqrt{\log n} + \log\log m)$ additional bits of storage. IN [4] an $O(1)$ probe scheme is given that requires only $O(\log n + \log\log m)$ additional bits of storage.

We give an implicit $O(1)$ probe search scheme for a domain of size m which is polynomial in the number of elements n. We prove that an implicit scheme exists whenever m is bounded by $2^{\mathrm{poly}(n)}$, based upon a probabilistic construction. It then follows from [4] that $O(\log\log m)$ additional bits are sufficient for any m.

We provide a refinement to Yao's theorem mentioned above which yields a better bound on the maximum m for which implicit $O(1)$ probe search schemes exist. Our proof technique gives a lower bound tradeoff between the number of probes and the size of the domain that allows implicit search. In particular, if $O(1)$ probe implicit search is possible then we can show that

$$m \leq \left. 2^{2^{2^{\cdot^{\cdot^{2^n}}}}} \right\} O(1).$$

Yao's proof technique obtains a tower whose height depends on n.

These results are obtained by means of a class of structures we call *rainbows*. A t-sequence over a set U is a sequence of length t, without repetitions, of elements in U. A (c, m, n, t)-rainbow is a coloring of all t-sequences over $\{1, \ldots, m\}$ with c colors so that for any set $S \subset \{1, \ldots, m\}$, $|S| = n$, all c colors occur in the coloring of the t-sequences over S. We show that the existence of $(c = n, m, n, t = O(1))$-rainbows is essentially equivalent to the implicit $O(1)$ probe search problem for a set of n elements chosen from the domain $\{1, \ldots, m\}$.

The relationship between rainbows and implicit $O(1)$ probe search schemes is specified by the following theorems:

Theorem 1 *For m, n, let $c = \max(n, \log m)$. The existence of a $(c, m, n, t = O(1))$-rainbow yields an implicit $O(1)$ probe search scheme for n elements from the domain $\{1, \ldots, m\}$.*

Theorem 2 *Given an implicit $O(1)$ probe search scheme for n elements chosen from the domain $\{1, \ldots, m\}$, we can construct an $(n, m, n, t = O(1))$-rainbow.*

To motivate rainbows we start in Section 2 by showing how they can be utilized to provide virtual memory.

Theorems 1 and 2 are proved in Sections 3 and 4. Section 5 contains a probabilistic construction for a $(c = n, m = 2^{\mathrm{poly}(n)}, n, t = O(1))$-rainbow, and an explicit construction for a $(c = n, m = \mathrm{poly}(n), n, t = O(1))$-rainbow. Thus, by Theorem 1, achieving the bounds claimed in the beginning of this section.

From their definition it is apparent that rainbows are related to Ramsey Theory. Indeed, the impossibility results we have are derived from Ramsey Theory, and are expressed in terms of Ramsey numbers.

In Section 6 we show bounds on the maximal m, as a function of n, for which an $(n, m, n, t = O(1))$-rainbow exists. Thus, Theorem 2 gives bounds on m for which implicit $O(1)$ probe search schemes exists. Section 6 also discusses the connection between rainbows and colorings of the uniform hypergraph.

Section 7 deals with the relationship between the rainbow structure and other structures, called dispersers, proposed in the literature (for completely different applications). Specifically, we show that if an explicit construction for a certain kind of dispersers is possible, then we can find an explicit construction for an implicit $O(1)$ probe search scheme for m which is $n^{\log n}$. For these dispersers Sipser [7] gave a probabilistic construction.

2 Rainbows Provide Virtual Memory

We now show how Rainbows can be used to simulate additional memory. The virtual memory problem with parameters c, n', t, l is defined below:

Virtual Memory Problem

Given:

- A set $R = \{r_1, r_2, \ldots, r_{n'}\}$, where $1 \leq r_j \leq m$ for $1 \leq j \leq n'$.
- A series of values $v_1, v_2, \ldots, v_l$ where $0 \leq v_i \leq c - 1$ for $1 \leq i \leq l$.

Arrange the elements of R in an array A of size n' (put each element of R in a different location) so that given $1 \leq j \leq l$, v_j can be reconstructed (decoded) quickly, via t accesses to A.

Note that we do not require anything about locating elements of R, only that they will reside somewhere in A.

The next lemma shows the relationship between this problem and the existence of rainbows.

Lemma 1 *Given a (c, m, n, t)-rainbow $\mathcal{C}$, the virtual memory problem for parameters c, n', t, l such that $n` - tl \geq n$ can be solved.*

Proof: Divide the first $t \cdot l$ locations of the array A into blocks of size t. The elements of R should be arranged in A so that the color assigned by $\mathcal{C}$ to the jth block, i.e. to the sequence $\langle A[jt+1], \ldots A[(j+1)t] \rangle$, is v_j. To achieve that, a greedy algorithm can be applied:

Greedy Encoding

- Set $U = R$
- For $j = 1$ to l
  - Find a sequence s colored v_j in U
  - Put the sequence s in the jth block of A
  - $U \leftarrow U \setminus s$
- Arrange U in the rest of A arbitrarily

Throughout the execution of the loop the number of elements in U is $n' - jt \geq n$. Hence there is a sequence in U colored by $\mathcal{C}$, and the find step in the algorithm always succeeds.

This arrangement means that in order to reconstruct v_j, one has to determine the color of the jth block under $\mathcal{C}$ and this can be done via t probes to A. This method is constructive if the color of a sequence under $\mathcal{C}$ can be determined effectively. □

3 Rainbows Yield Implicit $O(1)$ Probe Search

Our goal in this section is to prove Theorem 1. This section is strongly dependent on [4], which is the source of our techniques. A reader not familiar with the paper should be able to understand the major steps explained hereinafter.

We note the following theorem from [4]:

Theorem 3 *[4] n elements from the domain $\{1, \ldots, m\}$ can be arranged in a table of size n so that $O(1)$ probe search is possible, provided $O(\log n + \log\log m)$ additional bits of storage are available.*□

We will concentrate on proving a slightly weaker version of Theorem 1:

Theorem 4 *For m, n, let $c = \max(n, \log m)$. The existence of a $(c, m, n, t = O(1))$-rainbow yields an implicit $O(1)$ probe search scheme for $4n$ elements from a domain of size m.*

Later, we will show that the rainbow construction is robust in that the constants $(4n)$ are irrelevant, a $(c, m, n, t = O(1))$-rainbow can be translated into another $(c^{e_1}, m^{e_2}, n^{e_3}, t' = O(1))$-rainbow for arbitrary fixed exponents $e_1, e_2, e_3 > 0$.

It now might seem trivial to prove Theorem 1, given Theorem 3: Given $4n$ elements from a domain of size m, we encode the $O(\log n + \log\log m)$ bits required by the [FNSS] scheme above by choosing $O(1)$ groups of t elements, and ordering the elements in some fixed set of locations in the table as in the greedy encoding. During the search, the $O(1)$ special elements chosen for the encoding are read, if the search value is not found then the elements are interpreted under the rainbow interpretation as representing the extra bits of storage required by the scheme in [4].

Unfortunately, moving the required elements to their position as required by the encoding ruins the original order suggested in [4]. To prove our claim we must start afresh.

Given a set of n keys, $S \subset \{1, \ldots, m\}$, Fredman, Komlós and Szemerédi [5] describe how to find a perfect hash function $f : \{1, \ldots, m\} \mapsto \{1, \ldots, n\}$ with the property that f is one-to-one and onto when limited to the domain S. This function requires a description of $O(\log\log m) + o(n \log n)$ bits. The description is split into $O(1)$ words of size $O(\log\log m + \log n)$ bits, plus an additional $o(n)$ words of $O(\log n)$ bits each. Evaluating the function f requires reading only $O(1)$ of these words.

We say that S is in the natural order in the table T relative to f if $T[f(x)] = x$, $x \in S$. The natural order is easy to search, given f's description. Another order which is easy to search is obtained by applying an arbitrary permutation $\tau : \{1, \ldots, n/2\} \mapsto \{1, \ldots, n/2\}$ to the first half of the table and applying τ^{-1} to the second half. (For $1 \le i \le n/2$ set $T[\tau(i)] := T[i]$, for $n/2 < i \le n$ set $T[\tau^{-1}(i - n/2) + n/2] := T[i + n/2]$). The idea is that both τ and τ^{-1} are easy to compute. To compute $\tau^{-1}(i)$ simply evaluate $f(T[i])$, to compute $\tau(i)$ evaluate $f(T[i + n/2]) - n/2$. As both τ and τ^{-1} can be computed with one probe to the table, search can be done by performing two probes to the table.

This is a variation of Feldman's involution trick, as presented in [2].

The Method:

- Find an perfect hash FKS-function f for the $4n$ elements, as described in [5].
- Divide the elements into two sets depending whether $f(x) \le 2n$ or $f(x) > 2n$.
- Encode the description of f by arranging the elements x with $f(x) \le 2n$ in the first half of T using the greedy encoding of Section 2. By Lemma 1 this is possible.
- The arrangement defines a permutation τ of the elements in the first half of T, so organize the elements x with $f(x) > 2n$ in the second half of the table as required under τ^{-1}.

Searching for an element now requires decoding $O(1)$ words of the description of the FKS-function. Each decoding requires probing the table at $O(1)$ locations. The natural order can be reestablished by appropriately computing either τ or τ^{-1}, each of which requires one probe plus $O(1)$ probes to read the [5] function description. Overall, search requires $O(1)$ probes. $\Box$

4 Implicit $O(1)$ Probe Search Yields Rainbows

In this section we show that rainbows and $O(1)$ probe search schemes relate in the other direction as well; *i.e.*, given a search scheme we show how to construct a rainbow. More specifically, we prove a refined version of Theorem 2:

Theorem 5 *Given an implicit t-probe search scheme for n elements from the domain $\{1, \ldots, m\}$, an $(n, m, n, t + 2\log t)$-rainbow can be constructed.*

Proof: The sequences are assigned colors based on simulating a search scheme, the colors $1, \ldots, n$ correspond to locations $1, \ldots, n$ in some imaginary search array. The idea is that in a t-sequence there is enough information to simulate a t probe search. *I.e.*, given a t-sequence over $\{1, \ldots, m\}$, $e_1, e_2, \ldots e_t$ we simulate a search for e_1 in the imaginary array, where e_{i+1}, $1 \leq i \leq t-1$, is the element probed at step i. Since the location probed at step i is determined by the search value and the elements probed in steps 1 through $i-1$, we know the location in the imaginary array at each step of the simulation. The color assigned to the sequence is the last location we are to probe.

The only problem with this description is that e_1 might be probed at any of the t steps, not necessarily the last, but our sequences do not have repetitions. We can use $\log t$ bits to indicate the step number, j, at which e_1 is probed. This can be done by dedicating a pair of elements is allocated for each bit of j. If the elements are in order they encode 0, otherwise 1. We assume that these elements are at the end of the sequence, that is elements $e_{t+1}, e_{t+2}, \ldots, e_{t+2\log t}$.

To summarize, the color assigned to the sequence

$$e_1, e_2, \ldots, e_t, e_{t+1}, \ldots, e_{t+2\log t}$$

is the location of e_1 in the array for which the search is being simulated, where e_1 is encountered in the step encoded by $e_{t+1}, \ldots, e_{t+2\log t}$. Sequences that cannot be interpreted in such a fashion are colored arbitrarily.

Claim 1 *Given a set $S \subset \{1, \ldots, m\}$ of size n, and any color $1 \leq c \leq n$, there is a $t + 2\log t$-sequence over S which is colored c.*

Proof: Assume that the set S is arranged in the array A so that implicit t-probe search is possible. Consider the sequence consisting of the elements probed in A when searching for $A[c]$, concatenated to $2\log t$ elements in S not appearing in the probe sequence whose order encodes the step number at which cell c is probed. This sequence is colored c, and consists only of elements in S. $\Box$

5 Rainbow Construction

This section provides an explicit construction of rainbows when the number of colors $c = n$ and the length of the sequence t is a constant. We start with a construction for a domain m that is quadratic in the number of elements n (Lemma 2). The ideas behind this construction are later used in showing how to reduce a problem with domain m to another problem with domain $\sqrt{m}$ (Lemma 3) . This yields an explicit recursive construction for any m that is polynomial in n (Theorem 6). Theorem 6 yields as a corollary that implicit $O(1)$ probe search scheme is possible when m is polynomial in n. We conclude the section by showing that a probabilistic construction is good even when m is exponential in n (Theorem 7).

Lemma 2 *For any prime p, there is an explicit construction of a* $(c = n, m = p^2, n = p + 1, t = 2)$*-rainbow.*

Proof: Consider a 1-1 mapping from all elements $e \in \{1, \ldots, m\}$ to pairs (x, y) such that $0 \leq x, y \leq p - 1$. (For instance, $x = e \pmod p$, $y = (e - x)/p \pmod p$.) Given an element in $\{1, \ldots, m\}$, we will set its value to the value of the mapping.

Color the sequence $\langle u, v\rangle$, $u = (x_1, y_1)$, $v = (x_2, y_2)$, with the color $(y_2 - y_1)/(x_2 - x_1) \pmod p$. If $x_2 = x_1$ then color the sequence $\langle u, v\rangle$ with the color p. We have colored all edges of the full directed graph on m vertices. Note that the sequence $\langle u, v\rangle$ is colored as the sequence $\langle v, u\rangle$, hence we can consider the coloring as that of a complete undirected graph. To prove that this is a good coloring we need the following:

Claim 2 *Consider the edge induced subgraph G_i obtained by choosing all edges of color i. G_i consists of p vertex disjoint cliques of size p.*

Proof: First, note that every vertex $u = (x, y)$ has exactly $p - 1$ directed edges $(u, v_j = (x_j, y_j))$ colored i, for all $0 \leq i \leq p$. For $i = p$ these are simply pairs (x, y_j), $y_j \neq y$; for $i < p$ the x_j and y_j values are the $p - 1$ solutions to the equation $(y_j - y)/(x_j - x) = i \pmod p$.

To show that the undirected induced subgraph consists of cliques, assume that the $\langle u, v\rangle$ and $\langle v, w\rangle$ sequences are colored i: then the $\langle u, w\rangle$ sequence must also be colored i. If $u = (x_1, y_1)$, $v = (x_2, y_2)$ and $w = (x_3, y_3)$ either $i = p$ in which case $y_1 = y_2 = y_3$ and (u, w) is also colored p or $i < p$ in which case $(y_2 - y_1)/(x_2 - x_1) =$

$(y_3-y_2)/(x_3-x_2) = i \pmod{p}$. It now follows that $(y_3-y_1)/(x_3-x_1) = i \pmod{p}$. $\square$

Remark: Note that all vertices u_j, $u_j = (x_j, y_j)$, belonging to the same clique in G_i, have the same value $y_j - ix_j \pmod{p}$. This means that we can identify the clique in G_i containing a vertex u.

We can now resume the proof of the lemma: Given a set $S \subset \{1, \ldots, m\}$ of size $n = p + 1$, for all $0 \leq i \leq p$ at least two elements $u, v \in S$ belong to the same clique in G_i. This means that both sequences $\langle u, v\rangle$ and $\langle v, u\rangle$ are colored i. $\square$

To construct a rainbow for m polynomial in n we use a recursive construction. We explain how to use the construction above to transform the problem from a domain of size m to a domain of size $\sqrt{m}$, by concatenating two elements to each sequence in the $\sqrt{m}$ domain.

Lemma 3 *Given a construction of a $(c = n, m = p, n - 2, t)$-rainbow, p a prime, a $(c = n, p^2, n, t + 2)$-rainbow can be constructed.*

Proof: Let $\mathcal{C}_1$ be a $(p + 1, p^2, p + 1, 2)$-rainbow as described in Lemma 2 and let $\mathcal{C}_2$ be an $(n, p, n - 2, t)$-rainbow that exists by assumption. Our goal is to construct an $(n, p^2, n, t+2)$-rainbow. Given a $t+2$-sequence $\overline{e} = e_1, e_2, \ldots e_{t+2}$, over $\{1, \ldots, m\}$ we use e_1 and e_2 as indicators. If $e_1 > e_2$, then color $\overline{e}$ with the color assigned to (e_1, e_2) under $\mathcal{C}_1$.

Given a set $S \subset \{1, \ldots, m\}$, $|S| = n$, if all $p+1$ colors occur in the coloring of the 2-sequences over S under $\mathcal{C}_1$ then we are done. (In fact, the rainbow contains more colors than required).

Otherwise, at least one color is missing under $\mathcal{C}_1$, but there is at least one color that appears (we assume $n \geq 2$). Therefore, there is a color i such that no pair in S is colored by $\mathcal{C}_1$ with i, but there exist $u, v \in S$ such that (u, v) is colored $i - 1 \pmod{p+1}$ under $\mathcal{C}_1$.

Consider G_i, the edge induced graph defined by edges colored i and introduced above. Every element in S is in a different clique of G_i, otherwise there would have been a pair colored i. The cliques of G_i can easily be indexed as described by the remark at the end of Lemma 2.

If $e_1 < e_2$, we translate $\overline{e}$ to a t-sequence, $\overline{d} = d_1, d_2, \ldots d_t$, over $1, \ldots, \sqrt{m}$. We color $\overline{e}$ by the color assigned to $\overline{d}$ by $\mathcal{C}_2$. Let d_j be the index of the clique of G_i containing e_{j+2}, $1 \leq j \leq t$, and $i - 1 \pmod{p+1}$ is the color assigned to (e_1, e_2) under $\mathcal{C}_1$.

By the discussion above it follows that for every $S \subset \{1,\ldots,m\}$ and for every $1 \le k \le c$ there is a $t+2$-sequence over S that is colored k. Thus we have described a construction for an $(n, p^2, n, t+2)$ rainbow. $\Box$

Since for any integer x there is a prime in $(x, 2x)$, we can apply Lemma 3 recursively, each time reducing the domain from m to $2\sqrt{m}$. Using Lemma 2 as the base case provides us for any $d \ge 1$ with an explicit construction of an $(c = n, m = n^d, n, 2\lceil \log d \rceil + \lceil \log\log d \rceil)$-rainbow. Thus we have

Theorem 6 *For any domain m polynomial in the set size n there exists an $(n, m, n, O(1))$-rainbow. Given a sequence, its color can be determined in $O(1)$ time assuming modular arithmetic in unit time.*

Remark: Note that the proof implies that the existence of rainbows is a robust property, meaning that if p_1, p_2, p_3 are polynomials, and m is as a function of n such that a $(c, m, n, O(1))$-rainbow exists, then a $(p_1(c), p_2(m), p_3(n), O(1))$-rainbow exists as well.

It now follows from Theorem 1:

Corollary 1 *For any domain m polynomial in the set size n there exists an implicit $O(1)$ probe search scheme for which search requires $O(1)$ time, assuming modular arithmetic in unit time.* $\Box$

Probabilistic Constructions: We now turn to probabilistic constructions of rainbows for m which is *exponential* in n. Suppose $m = 2^{n^\ell}$ and consider a random coloring with n colors of all $\ell+2$ sequences over $\{1,\ldots,m\}$. For a set $S \subset \{1,\ldots,m\}$, $|S| = n$, the probability that a specific color is missing in the $\ell+2$ sequences over S is less than

$$(1 - 1/n)^{n(n-1)\ldots(n-\ell-1)}.$$

There are n colors and $\binom{m}{n}$ sets, hence the probability that there exists a set and a color such that the color is missing over the set is less than

$$\binom{m}{n} \cdot n \cdot (1-1/n)^{n(n-1)\ldots(n-\ell-1)} \le 2^{n^{\ell+1}} \cdot n \cdot e^{-n^{\ell+1}} \cdot e^{\ell^2 n^\ell} \ll 1.$$

Therefore we have:

Theorem 7 *For any domain m exponential in the set size n there exists an $(n, m, n, O(1))$-rainbow.* $\Box$

Corollary 2 *For any domain m exponential in the set size n there exists an implicit $O(1)$ probe search scheme.* $\square$

6 Bounds on Rainbows

In this section we give bounds on the maximum m, as a function of n and t, for which a $(c = n, m, n, t)$-rainbow can exist. We will do that by showing the connection between rainbows and colorings of the t-uniform hypergraph. Consider a coloring of all t-subsets (subsets of size t) of $\{1, \ldots, m\}$ with c colors. Ramsey Theory tells us that there exists a function $R(n, t, c)$ such that if $m > R(n, t, c)$ then for any coloring of the t-subsets of $\{1, \ldots, m\}$ with c colors there exists a set $S \subset \{1, \ldots, m\}$ of size n such that all the t-subsets over S are colored with the same color. (See the book by Graham, Rothschild and Spencer [6] for details on Ramsey Theory.)

Theorem 8 *If there exists a (c, m, n, t)-rainbow and $c > t!$ then*

$$m \leq R(n, t, t! + 1)$$

Proof: Let $\mathcal{C}$ be a (c, m, n, t)-rainbow. Define a coloring of the t-subsets of $\{1, \ldots, m\}$ $\mathcal{D}$: for each subset $H \subset \{1, \ldots, m\}$ of size t consider all possible orderings of H. Each of the $t!$ possible orderings receives a color in the rainbow. Since there are more than $t!$ colors in the rainbow we know that there is a color i, $1 \leq i \leq t! + 1$ which none of the orderings receives. $\mathcal{D}$ colors H with the least such i. From Ramsey Theory it follows that if $m > R(n, t, t! + 1)$ then there will be a set $S \subset \{1, \ldots, m\}$ of size n such that all of S subsets of size t are colored under $\mathcal{D}$ with the same color i. Hence, under $\mathcal{C}$ none of the t-sequences over S are colored i, and thus $\mathcal{C}$ is not a rainbow. $\square$

How fast does $R(n, t, t! + 1)$ grow? Let the tower functions $h_i(x)$ be defined as $h_1(x) = x$ and $h_{i+1}(x) = 2^{h_i(x)}$ for $i \geq 1$. That is

$$h_i(x) = 2^{2^{2^{\cdot^{\cdot^{\cdot^{2^x}}}}}} \Big\} i-1.$$

The Stepping Up Lemma in [6], page 91, yields the following: $h_{j-1}(c_1 \cdot n^2) \leq R(n, j, 2) \leq h_j(c_2 \cdot n)$ for some fixed c_1 and c_2. By the method of the proof of Ramsey's Theorem, increasing the number of colors from 2 to $t! + 1$ does not add more than $\log t! + 1$ to the height, i.e. $R(n, t, t! + 1) < h_{t+\lceil \log(t!+1) \rceil}(c_2 n)$. Hence we can conclude that for a $(c \geq t! + 1, m, n, t = O(1))$-rainbow to exist we must have

$$m \leq 2^{2^{2^{\cdot^{\cdot^{\cdot^{2^n}}}}}} \Big\} O(1).$$

Applying Theorem 2, on the connection between rainbow and t-probe search we get that an implicit t-probe scheme can exist only if $m < R(n, t', t'! + 1)$ where $t' = t + 2\log t$. Thus, for an implicit $O(1)$ probe search to exist we must have

$$m \leq 2^{2^{2^{\cdot^{\cdot^{\cdot^{2^n}}}}}} \Big\}O(1).$$

This constitutes a new proof of Yao's theorem [8] with better bounds. His bounds imply that $m < R(2n-1, n, n!)$, which grows much faster. Yao's proof has the advantage that it implies that whenever $m \geq R(2n-1, n, n!)$, the lower bound on the search time is $\lceil \log n \rceil$. Our proof cannot give better bounds than $\Omega(\log n / \log\log n))$, since $t! + 1$ must be less than n.

Any improvement on the lower bounds for rainbows would yield a better lower bound for implicit $O(1)$ probe search. Conversely, constructive implicit $O(1)$ probe search schemes for higher bounds imply better rainbow constructions. The reader can interpret this as either an optimistic or a pessimistic statement.

Undirected Rainbows: We now show that the existence of rainbows is closely related to that of undirected rainbows defined as follows: A (c, m, n, t)-*undirected rainbow* is a coloring of all t-subsets over $\{1, \ldots, m\}$ with c colors so that for any set $S \subset \{1, \ldots, m\}$, $|S| = n$, all c colors appear in the t-subsets over S.

Since the order itself in directed rainbows can determine $t!$ different colors, we know that $(c = t!, m, n, t)$-rainbows exist for any m and n such that $m \geq n$. However, by Ramsey Theory, this is not true for undirected rainbows. On the other hand, the next theorem shows that in order to give bounds on the maximum m for which $(c = n, m, n, O(1))$-rainbows exist, it is enough to consider undirected rainbows.

Theorem 9 *For every t there exists a constant b_t, dependent only upon t, such that a construction for a $(c = n, m, n, t)$-rainbow yields a construction for a $(c = n, m, \lceil \log(t!) + 1 \rceil \cdot n, b_t)$-undirected rainbow.*

Proof: The idea is to provide enough information in the b_t-subset so as to simulate an ordered set. If in addition to a t-subset, $\lceil \log(t!) \rceil$ bits are provided to determine the order in the t-subset, then the color of the subset will be the color of the corresponding t-sequence in the $(c = n, m, n, t)$-rainbow.

Let $\mathcal{C}$ be a $(c = n, m, n, t)$-rainbow. From Theorem 8 we know that $m < R(n, t, t!+1)$ and thus $m < h_{t'}(c_2 n)$ for some t' depending only on t. From the lower bound on $R(n, j, 2)$ of the Stepping Up Lemma there exists a $(2, n, m, t')$ undirected rainbow

$\mathcal{A}$. Let $b_t = t + t' \cdot \lceil \log(t!) \rceil$. Define $\mathcal{C}'$, a coloring of b_t-subsets, as follows: Sort the b_t-subset and partition it into $\lceil \log(t!) \rceil + 1$ subsets of consecutive elements such that the subset of the largest elements is of size t and all the rest are of size t'. Compute the coloring under $\mathcal{A}$ of each of the t'-subsets. Each of the $\lceil \log(t!) \rceil$ t'-subsets supplies one bit under its 2-coloring, and together those bits determine an ordering of the t-subset. The color $\mathcal{C}'$ assigns is the one $\mathcal{C}$ assigns the t-sequence resulting from the t-subset when it is ordered by the encoding given by the smaller subsets.

To see that $\mathcal{C}'$ is indeed a $(c = n, m, \lceil \log(t!) + 1 \rceil \cdot n, b_t)$-undirected rainbow, consider any $S \subset \{1, \ldots, m\}$ of size $\lceil \log(t!) + 1 \rceil \cdot n$. Partition S into $\lceil \log(t!) + 1 \rceil \cdot$ subsets $S_1, S_2 \ldots$, such that each S_i is of size n and all the elements of S_i are smaller than those of S_{i+1}. For any color $1 \leq j \leq c$, $\mathcal{C}$ colors at least one t-ordered subset of $S_{\lceil \log(t!)+1 \rceil}$ with j. The order of this subset determines $\lceil \log(t!) \rceil$ bits $b_1, b_2, \ldots b_{\lceil \log(t!) \rceil}$. In each subset S_i there is a t'-subset colored b_i under $\mathcal{A}$. The b_t subset of S which is the union of all these subsets is colored j under $\mathcal{C}'$. $\square$

7 Construction through dispersers

In this section we show how an explicit construction for dispersers, defined below, yields an explicit construction for rainbows with $m = n^{\log n}$. An (m, n, d, a, b)-*disperser* is a bipartite graph with m nodes on the left side, each with degree at most d, n nodes on the right side with the property that every subset of a nodes in the left side is connected to at least b of the nodes of the right side. These graphs have been used, for instance by Ajtai, Komlós and Szemerédi [1] and Sipser [7], to remove randomness in probabilistic algorithms. Cohen and Wigderson [3] provide a survey of constructions and applications.

Let $m = n^{\log n}$. We first show how to construct a rainbow with $\log n$ colors for such m and n, and then show how to apply it with a $(m, n, \log^2 n, n, n/2)$-disperser to get an $(c = n, m, n, O(1))$-rainbow.

Lemma 4 *There exists an explicit construction for a* $(c = \log n, m, n, O(1))$*-rainbow, if* m *is* $n^{polylog(n)}$

Proof: For $1 \leq x \leq m$ let x_i denote the ith bit of x. Consider the coloring of pairs that assigns the pair (x, y) $\min_{1 \leq i \leq \log m} x_i \neq y_i$, i.e. the first bit in which x and y differ.

Claim 3 *In any set $S \subset \{1,\ldots,m\}$ of size n, the pairs must be colored with at least $\log n$ different colors.*

To see that the claim is true, consider organizing the elements of S in a trie, i.e. in a binary tree where each element appears as a leaf and its value is determined by the path from the root. If a node in the ith level of the trie has two children, then there is a pair $\langle x, y\rangle$, where x is a descendent of the left child and y a descendent of the right child, that is colored i. There must be at least $\log n$ levels in which there is a node with 2 children, since each level at most doubles the number of nodes from the previous one and there are n leaves.

The claim shows that rather than having a set of size n out of a domain of size m, the problem can be reduced to that of a set of size $\log n$ from a domain of size $\log m$. If m is $n^{\mathrm{polylog}(n)}$, then $\log m$ is polynomial in $\log n$, and hence the construction of Theorem 6 can be applied to obtain the required rainbow. $\Box$

Sipser [7] gave a probabilistic construction for an $(n^{\log n}, n, \log^2 n, n, n/2)$-disperser. Given such a disperser D, we now show how to use such dispersers to amplify rainbows, and construct $(c = n, m = n^{\log n}, n, O(1))$-rainbows. Let $\mathcal{C}$ be a $(c = \log^2 n, m, n-1, t)$-rainbow whose existence is assured by the previous lemma and the remark following Theorem 6. Consider a coloring of $t+1$-tuples over $\{1,\ldots,m\}$ defined as follows: The first t elements are used to obtain a color $e \in \{1 \ldots \log^2 n\}$ via $\mathcal{C}$. The $t+1$st element, $v \in \{1,\ldots,m\}$, is treated as a node on the left side of D. e specifies a neighbor of v on the right side of D. The neighbor is the color of the t-tuple. Since any set of n nodes on the left side is adjacent to at least half the nodes on the right side, and since $\mathcal{C}$ is a $(\log^2 n, m, n, t)$-rainbow, it follows that for any set $S \subset \{1,\ldots,m\}$ of size n there is a set $T \subset \{1,\ldots,n\}$ of size at least $n/2$ such that one can specify in this manner any member of T. Using the construction of Lemma 2 this can be amplified to include all the n nodes on the right. This construction gives a $(c = n, m = n^{\log n}, n, 2t+2))$-rainbow. Therefore, an explicit construction for a disperser with those parameters yields an implicit $O(1)$ probe search scheme for $m = n^{\log n}$.

No explicit construction with parameters close to the ones given in [7] is known. The best explicit construction for such expanders is given in Ajtai, Komlós and Szemerédi [1].

8 Conclusions and Open Problems

As a consequence of the results of this paper, the maximal m for which implicit $O(1)$ probe is possible lies between $2^{\mathrm{poly}(n)}$ and a constant height tower of powers. One obvious open problem is to close this gap. Finding an explicit construction for rainbows with m superpolynomial in n is another obvious research direction. A different question is whether rainbows are useful for implicit data representation in other settings.

Acknowledgments

We thank Noga Alon, Joel Friedman, Nati Linial, Mike Luby, Jeanette P. Schmidt, Alan Siegel and Avi Wigderson for helpful discussions and advice. We are grateful to the two anonymous referees for their diligent reading and many useful remarks.

References

[1] M. Ajtai, J. Komlós, E. Szemerédi, *Deterministic Simulation in LOGSPACE*, Proc. 19*th* ACM Symposium on Theory of Computing, 1987, pp. 132-140.

[2] 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 58, 1988, pp. 57–68.

[3] A. Cohen and A. Wigderson, *Multigraph amplification*, manuscript 1989.

[4] A. Fiat, M. Naor, J. P. Schmidt and A. Siegel, *Non-Oblivious Hashing*, Proc. 20*th* ACM ACM Symposium on Theory of Computing, Chicago, pp. 367–376.

[5] 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, Vol 31, 1984, pp. 538–544.

[6] R. L. Graham, B. L. Rothschild and J. H. Spencer, **Ramsey Theory**, Willey 1980.

[7] M. Sipser, *Expanders, Randomness or Time versus Space*, Journal of Computer and Systems Sciences 36, 1988, pp. 379-383.

[8] A.C. Yao, *Should Tables Be Sorted?*, Journal of the Association for Computing Machinery, Vol 28, 1981, pp. 615–628.