

On the Cell Probe Complexity of Membership and Perfect Hashing

Rasmus Pagh



Department of Computer Science
University of Aarhus, Denmark

Work partly done while visiting Stanford University

Hersonissos, Crete, July 7, 2001

Static Cell Probe Complexity

The cell probe model (Minsky and Papert 1968):

- A data structure is a sequence of s cells.
- Each cell stores a b -bit string.
- The complexity of an operation is the number of cells probed (i.e., computation is not counted).

We will consider the *deterministic worst case* complexity (i.e., over all possible data, for the optimal choice of data structure and deterministic query algorithm).

Of particular interest is the *adaptivity* of the query scheme (i.e., to what extent probes depend on the results of earlier probes).

Membership and Perfect Hashing

Both problems deal with a set $S \subseteq U = \{0, 1\}^b$ of size n .

Membership:

- Data: S
- Query: Is $x \in S$?

(We will consider only data structures of $s = O(n)$ cells.)

Let \mathcal{H}_n be a *perfect family of hash functions* $f_i : \{0, 1\}^b \rightarrow \{1, \dots, r\}$, i.e., such that for any S there exists $f_i \in \mathcal{H}_n$ that is 1-1 on S .

Perfect hashing (\mathcal{H}_n):

- Data: $f_i \in \mathcal{H}_n$
- Query: What is the value of f_i on x ?

(We consider $r = O(n)$ and data structures of $s = O(n/b + 1)$ cells.)

Some Earlier Results

We use $+$ to indicate an *adaptive choice*.

Membership

- When n is large, 1 cell probe suffices (bit vector).
- For small n , $1_{\text{fixed}} + 1$ cell probes suffice (perfect hashing) [Yao, 1978].
- For most parameters, $1_{\text{fixed}} + 1 + 1$ cell probes suffice.
 $O(1)$ adaptive cell probes always suffice.
[Fredman, Komlós & Szemerédi, 1982].

Perfect hashing

- $O(1)$ adaptive cell probes suffice [Schmidt & Siegel, 1989].
- $1_{\text{fixed}} + 1$ cell probes suffice, using $\Theta(n)$ cells [Pagh, 1999].

Overview of New Results (Vanilla Versions)

Results for membership:

- **3** cell probes are necessary in general, and
- $1_{\text{fixed}} + 2$ cell probes suffice.

Results for perfect hashing:

- 1_{fixed} cell probe + **1** *bit* probe suffices, but
- without adaptivity one generally must probe $\Omega(n)$ bits.
- **1** adaptive bit probe is not enough for $r = n$, but
- minimal range can be achieved with one adaptive cell probe.

(Some results on bit probe complexity are not covered in the talk)

Membership: 3 Cell Probes Are Necessary

[Yao, 1978]: $1_{\text{fixed}} + 1$ cell probes are generally not enough if the second cell probe is constrained to read an element of S .

The same turns out to be true with no constraint:

Theorem: For $b \geq 3 \log n$, membership cannot be answered by $1 + 1$ cell probes unless $b = \Omega(n)$.

Corollary: FKS hashing uses an optimal number of cell probes.

Proof overview:

- The first cell probe can be assumed to be fixed.
- Precise bound on how many sets may be represented for each bit pattern in the first cell. (Not many more than in Yao's case.)
- $\Rightarrow 2^{\Omega(n)}$ bit patterns are needed.

Membership Using $1_{\text{fixed}} + 2$ Cell Probes

[Schmidt & Siegel, 1989]: “ k -probe perfect hash functions”, giving a *set* of $k > 1$ cells to probe, require little memory!

(This has also shown up in load balancing and PRAM simulation.)

What happens for $k = 2$?

- Hash table size $(2 + \epsilon)n$ is enough.
- $O(\log n)$ -wise independent hash function values suffice.

This yields a (nonexplicit) $1_{\text{fixed}} + 2$ cell probe membership scheme.

Analysis outline:

- No subset $S' \subseteq S$ hashes to fewer than $|S'|$ cells \Rightarrow
 S can be placed according to a given hash function (Hall's Thm.)
- Bounding the expected number of such obstacles by $O(1/n)$.

Perfect Hashing Using 1_{fixed} Cell + 1 Bit Probe

Idea:

- Get a perfect hash function from the “2-probe perfect hash function” by probing a *single bit* choosing one of the cells.
- Determine which bit to probe using a hash function.

This works for $r = 64n$, using a table of n bits, and an $O(\log n)$ -wise independent hash function.

Analysis outline:

- A perfect hash function cannot be found
 - \Rightarrow A certain 2-SAT instance is not satisfiable
 - \Rightarrow Some series of implications yields $\neg\alpha \Leftrightarrow \alpha$, for a variable α .
- Bounding the expected number of such obstacles by $1/4$.

Nonadaptive Queries Are Exponentially Slower

Switching for a moment to the bit probe model ...

Question: Can we get rid of adaptive bit probes altogether?

Answer: **No.** Nonadaptively, a *constant fraction* of the $O(n)$ bit data structure must be read.

Reason: For large U , many sets will probe within some set of bits too small to encode a perfect hash function.

Perfect hashing (for suitable parameters) has the largest possible gap:

With no adaptive bit probes: $\Omega(n)$ bit probes needed.

With one adaptive bit probe: $O(\log n)$ bit probes suffice.

(Seems to be the first problem for which this is known.)

Perfect Hashing: Adaptivity vs Minimal Range

Question:

With one adaptive bit probe, how much can $r = 64n$ be reduced?

Partial answer:

Not below $1.44n$, regardless of space usage.

Reason:

Few bits are probed for each set

⇒ “Many” data structures are good for each set

⇒ Some data structure is good for “many” sets (impossible).

Minimal Perfect Hashing Using $1_{\text{fixed}} + 1$ Probes

The minimal perfect hashing scheme of [Schmidt & Siegel, 1989] uses a rather high constant number of adaptive cell probes.

One adaptive cell probe suffices (nonexplicitly):

- The first cell probe retrieves a hash function separating S into $O(n/\log n)$ buckets of $O(\log n)$ elements.
- The second cell probe retrieves a minimal perfect hash function for the bucket and an offset.

If made explicit, this could be more practical than current schemes. This seems overlooked in the literature.

Problems to Chew On

Algorithm design:

RAM versions of the practically promising data structures.

Analysis of algorithms:

Precise analysis of “ k -probe perfect hashing” for $k > 2$.

Complexity:

Lower bound on the bit probe complexity of perfect hashing?