Lecture 09

An abridged introduction to the cell probe model

1. The cell-probe model is a computational model where we have a machine with m registers, of b bits each. An algorithm in the model is a query tree: each node v in the tree probes a certain register r_v, and branches to its children depending on which of the 2^b values is held in r_v. For this lecture we assume that the branching is deterministic. The time-complexity of such an algorithm is the depth of the tree.

2. A data-structure problem in this model asks for a way of representing a given data (a list, a graph, etc) in the registers of such a machine, so that certain operations can be performed on the data in the shortest possible time.

3. For example, one of the earliest results (Fredman, Komlós and Szeméredi, 1984) shows how to represent a list $L \subseteq [m]$ of n integers, using $O(n)$ registers of bit-length $b = O(\log m)$, so that any query "$x \in L$?" can be answered in $O(1)$ time.[1]

4. We will focus on dynamic data-structure problems. Such problems involve two kinds of operations: *update* operations, and *query* operations. Here are three examples:

4.1 *Dynamic rank.* Maintain a subset of $[m]$ under insertion and removal of elements (*the update operations*) so that we may know how many elements of the set are smaller than a given j (*the query operations*).

4.2 *Dynamic connectivity.* Maintain an undirected graph under addition and removal of edges (*the updates*) so that we may know whether two vertices i and j are connected.

[1] This is only non-trivial provided $2^B \gg n$, because we can always encode all 2^B answers, one in each register.

4.3 *Dynamic reachability.* Maintain a directed graph under addition and removal of edges (*the updates*) so that we may know whether there is a path from i to j.

5. There are many interesting results in this area, both clever and surprising algorithms. The three problems above are chosen because of how they contrast in this respect.

5.1. Dynamic rank is completely understood. It has a solution with $b = \log m$, and both update and query time $O(\log m/\log\log m)$ (Dietz, 1989), and we will prove (after Fredman and Saks, 1989) that this is optimal.

5.2. Dynamic connectivity is almost completely understood. Namely, it has recently finally been proven that $(\log n)^{O(1)}$ worst-case update and query time is enough to solve the problem (Kapron, King, Mountjoy, 2013); this was known for amortized update-time for a while now (Thorup, 2000), and lower bounds of $\log n/\log\log n$ are known.

5.3. Dynamic reachability is far from being understood; for instance, for the single-source version of the problem, sub-quadratic solutions with $n^{.6}$ query-time and $n^{1.6}$ update time are known; (to my knowledge) no sub-quadratic solution is known for the general case. It is generally believed that, for some constant $0 < \varepsilon \leq 2$, n^{ε} update and query time will be impossible, but no-one knows how to prove such a result for any ε. In fact, it is an open problem to show *any* n^{ε}-lower-bound for *any* dynamic data-structure problem.

The chronogram method

☞ The following problem could be called *the rank-problem mod 2*:

6 *Dynamic prefix.* Maintain a bit vector $x \in \{0,1\}^m$ under the operations:

- *change*(i, a), which sets $x_i := a$, and
- *prefix*(j), which returns $\sum_{i \leq j} x_i \mod 2$.

7 *Lower-bound.* No solution to dynamic prefix is possible with both update and query time $\Omega(\log m/\log\log m)$.

☞ This is proven by a technique called *the chronogram method*, which has been invented by Fredman and Saks in the 1989 paper, and has been used many times since to prove lower-bounds on dynamic problems.

Proof. Suppose our sequence of operations is of the form:

$$\mathit{change}(i_1, a_1) \quad \mathit{change}(i_2, a_2) \quad \ldots \quad \mathit{change}(i_n, a_n) \quad \mathit{prefix}(j),$$

for $n = \sqrt{m}$, and some fixed $i_1, \ldots, i_n$ which we will describe later. Let us partition the *change* operations into E *epochs.* Epoch 1 consists of the last $\ell_1 = (\log n)^3$ changes, and epochs 1 to e will consist of the last $\ell_e = (\log n)^{3e}$ changes $\mathit{change}(i_n, a_n), \ldots, \mathit{change}(i_{n-\ell_e}, a_{n-\ell_e})$, so we have $E = \frac{\log m}{6 \log\log m}$ epochs in total (so epoch-1 changes happen *after* epoch-2 changes).

For each value of $a \in \{0,1\}^n$, we may now define $R(a)$ to be the content of the registers after executing all the changes. We may also define $q(a)$ to be the m-bit vector $\mathit{prefix}(1), \ldots, \mathit{prefix}(m)$ when the prefix operation is executed on $R(a)$.

Each $\mathit{change}(i_k, a_k)$ operation writes on some registers — precisely which registers depends on $i_{\leq k}$ and $a_{\leq k}$; whenever it writes on register r, let us *stamp* register r with the index of the current epoch (possibly overwriting the last stamp), so that by the time we run $\mathit{prefix}(j)$, each register is stamped with the last (temporally last, numerically first) epoch that wrote on it. For a given a and a given epoch, let $R^e(a)$ denote $R(a)$ where we revert every e-stamped register to the value it had just before epoch e began. Now define $q^e(a)$ to be the m-bit vector $\mathit{prefix}(1), \ldots, \mathit{prefix}(m)$ when the prefix operation is executed on $R^e(a)$.

7.1 *Important epochs.* Intuitively, $q(a)$ and $q^e(a)$ are close (in hamming distance) if the epoch-e writes have little influence on the output; if $q(a)$ and $q^e(a)$ are far apart, then the epoch-e writes are important in order to know the output when the sequence of changes is given by a.

Let us make that notion more precise, by saying that epoch e is *important* if omitting the epoch-e writes causes the output to be significantly wrong for most changes a, i.e.:

$$\Pr_a \left[\|q(\bar{a}) - q^e(\bar{a})\|_1 \geq \frac{m}{20} \right] > 1/2.$$

Hence by *significantly wrong*, we mean wrong in at least $1/20$ fraction of the coordinates.

7.2. Intuitively, if an epoch is important, then the algorithm should need to query at least some e-stamped register. Let us begin by first proving this rigorously, and then we will finish the proof by showing every epoch is important (which suffices, as there are $E = \Omega(\log m / \log\log m)$ epochs in total).

Indeed, the worst-case time for all *prefix* queries is at least the average:

$$\frac{1}{m2^k} \sum_{a\in\{0,1\}^k} \sum_{j\in[m]} \text{time of } \textit{prefix}(j) \text{ on } R(a)$$

And the time spent by *prefix*(j) on $R(a)$ is at least

$$\sum_{e=1}^{E} \begin{cases} 1 & \text{if } \textit{prefix}(j) \text{ reads some } e\text{-stamped register on } R(a), \\ 0 & \text{otherwise.} \end{cases}$$

Hence the worst-case time is at least:

$$\frac{1}{m2^k} \sum_{e} \sum_{a} \#\{j \mid \textit{prefix}(j) \text{ reads some } e\text{-stamped register on } R(a)\}.$$

Now notice the following: if $q(a)_j \neq q^e(a)_j$, then surely *prefix*(j) must read some e-stamped register on $R(a)$ (otherwise the output of *prefix*(j) would be the same for both $R(a)$ and $R^e(a)$). Hence the term inside the sum is at least

$$\#\{j \mid q(a)_j \neq q^e(a)_j\} = \|q(a) - q^e(a)\|_1.$$

The sum is now at least

$$\frac{1}{m2^k} \sum_{e} \frac{m}{20} \#\left\{a \,\middle|\, \|q(a) - q^e(a)\|_1 \geq \frac{m}{20}\right\} \geq \frac{1}{20} \sum_{e} \Pr_{a}\left[\|q(\bar{a}) - q^e(\bar{a})\|_1 \geq \frac{m}{20}\right],$$

which is, finally, at least $\frac{1}{40}$ fraction of the number of important epochs.

☞ Now we show that, for a suitable choice of $i_1, \ldots, i_n$, every epoch is important.

7.3. For a given e, let $a = a_< a_\geq$, where $a_< = a_{<n-\ell_e}$ denotes the changes previous to epoch e, and $a_\geq = a_{\geq n-\ell_e}$ denotes the changes from epoch e onward. Then it is enough to show that, for every e and $a_<$, at most half of the $a_\geq$ will have

$$\|q(a) - q^e(a)\|_1 < \frac{m}{20}.$$

7.4. The first observation is that, for fixed e and $a_<$, $q^e(a)$ cannot take on many different values. This is because all the writes previous to epoch e are the same — we fixed $a_<$ — and there aren't many changes happening after epoch e — at most ℓ_{e-1} of them.

Indeed, if each change operation takes at most t time-steps, at most $s = 2m2^{bt}$ registers are accessed (read or written) by any change operation.

But because any two $R^e(a)$ (for different $a_{\geq}$) only differ by (the writes made by) ℓ_{e-1} changes, then there are at most

$$\sum_{k=0}^{t\ell_{e-1}} \binom{s}{k} 2^{bj} \leq t\ell_{e-1} s^{t\ell_{e-1}} 2^{bt\ell_{e-1}} \leq 2^{0.1\ell_e}$$

different values for $R^e(a)$, and the same holds for $q^e(a)$. So let V be the number of different $q^e(a)$.

7.5. The second observation is that, provided we pick the i_k correctly, there are few $q(a)$ within a Hamming ball of radius $\leq \frac{m}{10}$. The following claim will suffice: there is a way of picking $i_1, \ldots, i_n$ such that the distance between any two $i, i' \in \{i_{n-k}, \ldots, i_n\}$ is at least $\frac{m}{2k}$. We pick them in reverse as follows: $i_n = \frac{m}{2}$, and then $i_{n-1} = \frac{m}{4}$, $i_{n-2} = \frac{3m}{4}$, and then $i_{n-3} = \frac{m}{8}$, $i_{n-4} = \frac{3m}{8}$, $i_{n-5} = \frac{5m}{8}$, and $i_{n-6} = \frac{7m}{8}$, *etc* (a picture with the unit interval should convince you).

How many different $q(a)$ can be contained within a Hamming ball B of radius $\leq \frac{m}{10}$? Pick any "center" $q(c) \in B$, and let i, i' be consecutive change-indices in $\{i_{n-\ell_e}, \ldots, i_n\}$ (not consecutive in time, but in value); then for any $q(a) \in B$, $q(a)_i, q(a)_{i+1}, \ldots, q(a)_{i'-1}$ can only take two forms, either $q(c)_i, q(c)_{i+1}, \ldots, q(c)_{i'-1}$ or $1 - q(c)_i, 1 - q(c)_{i+1}, \ldots, 1 - q(c)_{i'-1}$ — because $x_{i+1}, \ldots, x_{i'-1}$ are never changed. In the first form, the contribution to the hamming distance $\|q(a) - q(c)\|_1$ is 0, and in the second case it is at least $\frac{m}{2\ell_e}$.

Hence there can be at most $\frac{\ell_e}{5}$ consecutive i, i' which fall into the second case, and every $q(a) \in B$ equals $q(c)$ except for these positions, so there are at most[2]

$$\sum_{k=1}^{\ell_e/5} \binom{\ell_e}{k} \leq 2^{H_2(1/5)\ell_e} \leq 2^{0.6\ell_e}$$

7.6. Now we have that each $a_{\geq}$ for which $\|q(a) - q^e(a)\|_1 < \frac{m}{20}$ gives us a string $q^e(a)$ among $2^{0.1\ell_e}$-many, and a string $q(a)$ in the Hamming ball B_a of radius $\frac{m}{20}$ around $q^e(a)$. By the triangle inequality, B_a is contained in a hamming ball of radius $\frac{m}{10}$ around some string $q(c)$, hence each B_a has size at most $2^{0.6\ell_e}$. That implies that there are at most $2^{\cdot 7\ell_e}$ $a_{\geq}$ for which the condition holds, which is less than $2^{\ell_e}/2$. ■

[2] Above we use the estimate $\sum_{k=1}^{\alpha n} \binom{n}{k} \leq 2^{H(\alpha)n}$, where $H(\alpha) = \alpha \log \frac{1}{\alpha} + (1-\alpha) \log \frac{1}{1-\alpha}$, and $\alpha \in (0, 1/2)$. See Stasys Jukna, Extermal Cominatorics, 22.9.