6.897: Advanced Data Structures Spring 2005

Lecture 12 — March 15, 2005

Prof. Erik Demaine *Scribe: Ilya Baran*

1 Overview

In the last lecture we used round elimination to prove lower bounds for the static predecessor problem in the cell probe model. We showed a lower bound of $\Omega(\min\{\lg_a w, \lg_b n\})$ on the number of probes required to solve the problem, where $a = O(\lg \text{space}(n))$, the number of bits to index the data structure, and $b = w$, the number of bits returned by a single cell probe. For a polynomial size data structure, this implies that when $\lg n \lg\lg n = \lg^2 w$, some problem instances require $\Omega(\frac{\lg w}{\lg\lg w}) = \Omega(\sqrt{\frac{\lg n}{\lg\lg n}})$ probes.

The bound $\lg_w n$ is matched by fusion trees, but van Emde Boas achieves $\lg w$ per query, which does not match $\lg_a w$. In this lecture we show upper and lower bounds of $\Theta(\min\{\lg_w n, \frac{\lg w}{\lg\lg w}, \frac{\lg w}{\lg \frac{a}{\lg n}}\})$.

2 An $O(\frac{\lg w}{\lg \frac{a}{\lg n}})$ Data Structure

We show how to use superpolynomial space ($a = \omega(\lg n)$) to build a data structure. As for the lower bound, we think of it as an interaction protocol between Alice, the processor, and Bob, the memory. We break a query into $\Theta(a/\lg n)$ equal size chunks, each of size $\Theta((w \lg n)/a)$ bits. This corresponds to a tree with height $\Theta(a/\lg n)$ and branching factor $2^{\Theta((w \lg n)/a)}$. The n nodes in the set correspond to n root-to-leaf paths in this tree. Using a single query from to Bob, we would like to determine the node at which the path from the root to the query node diverges from all paths to the leaves in the set. If we could do that, we reduce the problem to the predecessor problem on that node, which corresponds to a chunk of length $w' = w/\Theta(a/\lg n)$. Doing this recursively gives the desired bound.

To find the first node on the path with a mismatch, we hash each chunk into $\Theta(\lg n)$ bits and send the hash values for each chunk packed together to Bob. Because there are n elements, we can choose the constant so that there are no false positives with high probability. Bob responds with the index of the first chunk with a mismatch and Alice recurses on this chunk. Because Bob represents memory, he must be a lookup table with a precomputed response to every query Alice may send. Notice that we only need to recurse if there is more than one element in the chunk where the query diverges from the elements. In other words, if we consider the tree formed by the n paths, recursion occurs only if the query path diverges at a node with at least 2 children (if the divergence occurs at a node with only one child, the maximum of its subtree is returned or the predecessor of the minimum of its subtree). The size of the recursion tree is therefore $O(n)$ and so the total size of the lookup tables can be to be $2^{\Theta(a)}n = 2^{\Theta(a)}$.

We note that if a is $\Omega(lg^{1+\varepsilon} n)$ then the time bound achieved by this data structure is $O(\lg_a w)$, which we already know is optimal. We will later show that the time is optimal even for slightly smaller a.

3 An $O(\frac{\lg w}{\lg \lg w})$ Data Structure

We use the same idea as for the above data structure, except in this case, we break a query up into k chunks, where k will be determined later. As before, we send $\Theta(a/k)$ bits for each chunk. However, there may be more than $2^{\Theta(a/k)}$ elements in a chunk. To deal with this, let us define the weight of an element in a chunk to be the number of its descendants (elements who share the prefix up to that chunk) and let us store just the $2^{\Theta(a/k)}$ heaviest edges in the hashtable. Now the first mismatch in a hashtable can occur either at the place where the query path diverges from the paths of the n elements or at the place where the query path proceeds along a light edge. We can determine which case we are in with an additional probe. If the query path actually diverges, we recurse on the chunk as in the first data structure, having reduced w to w/k. If the query path proceeds along a light edge, that edge must have at most $n/2^{\Theta(a/k)}$ descendants and we recurse on its subtree, having reduced n to $n/2^{\Theta(a/k)}$.

At every probe we wither reduce n by a factor of $2^{\Theta(a/k)}$ or we reduce w by a factor of k. Clearly, then, there are at most $O(\max\{\lg_k w, \lg_{2^{\Theta(a/k)}} n\})$ probes per query. We choose k to balance these out: $\frac{\lg w}{\lg k} = \frac{\lg n}{\Theta(a/k)}$. Multiplying through, we find that $k \lg k = \Theta(a \lg w / \lg n)$ and taking the logarithm of both sides, we find that $\lg k = \Theta(\lg(a/\lg n) + \lg\lg w) = \Theta(\max\{\lg(a/\lg n), \lg\lg w\})$. Therefore, the running time, $\lg_k w$, is $\Theta(\min\{\frac{\lg w}{\lg(a/\lg n)}, \frac{\lg w}{\lg\lg w}\})$, as desired.

Using arguments similar to the one for the simpler above data structure, we can show that if we convert this protocol into a static data structure using lookup tables, space usage will be $O(n^{1+\varepsilon})$.

4 A Tight Lower Bound

We now show how to modify the lower bound proved in last lecture to match the upper bounds given by the data structures above. The goal is to use round elimination to reduce n to at least $n/2^{\Theta(a/k)}$ and w to at least w/k at every round, as in the upper bound. The lower bound from last lecture reduces n to n/bt^2 (where t is the number of rounds) and w to w/at^2 each round. We cannot afford the reduction from w to w/at^2, so we tighten it using a lower-bound framework by Chakrabarti and Regev [2]. This framework is based on two important ideas, and it constitutes a refinement of the round elimination lemma.

The first idea is message compression: if Alice speaks first in a protocol for $f^{(k)}$, we can get a protocol for f but with Alice's first message decreased to length $O(a/k)/\delta^2$ and the error probability increased from p to $p+\delta$. If we set $\delta = \Theta(1/t)$, we do not increase the error probability too much, but we reduce the first message to have length $a_1 = O(a/k)t^2$.

The above procedure reduces the first message length and because we obtain a protocol for f from one for $f^{(k)}$, we reduce w to w/k. However, we have not eliminated a round. To do this, we use the second idea from [2], message switching. The idea is that if Alice's first message has length l, then it can be eliminated and instead Bob can send a first message consisting of all 2^l possible replies to Alice's message. The next message from Alice must be increased by an additive l, because Alice must actually tell Bob her original message. However, this can be ignored since it's just a constant factor, and we'll soon eliminate this message anyway. Using message switching, we can therefore eliminate Alice's first message and increase Bob's first message length to $w2^{\Theta(a/k)t^2}$.

We can now eliminate the message from Bob to Alice, using the round elimination lemma (as in the previous lecture), reducing n to $\frac{n}{w2^{\Theta(a/k)t^2}t^2} = \frac{n}{w2^{\Theta(a/k)t^2}}$. Because whenever we eliminate two rounds, w reduces to w/k, we can eliminate $\Omega(\min\{\lg_k w, \lg_{w2^{\Theta(a/k)t^2}} n\})$ rounds, which is our lower bound. To obtain the strongest possible lower bound, we balance: $\frac{\lg w}{\lg k} = \frac{\lg n}{\lg w + \Theta(a/k)t^2}$. The right hand side of this equation can be written as $\Theta(\min\{\lg_w n, \frac{\lg n}{at^2/k}\})$. We now balance $\frac{\lg w}{\lg k} = \frac{\lg n}{at^2/k}$ or $k \lg k = \frac{at^2 \lg w}{\lg n}$. When we take the logarithm of both sides, t^2 becomes negligible because $t = O(\lg w)$, and the calculation proceeds as for the upper bound.

5 Related Problems

We discussed several related problems for integer search.

5.1 Dynamic Predecessor

The problem is to maintain a predecessor data structure subject to insertions and deletions. It is known how to solve this problem in $O(\textit{static} \cdot \lg \lg n)$ time where *static* is the static time bound proved above. Also Raman [5] showed how to solve this problem in $O(\lg_w n)$ time. The van Emde Boas queue solves this in $O(\lg w)$ time. It is open whether one can achieve $O(\lg_w n)$ or $O(\lg w)$ deterministically, since both data structures mentioned above are randomized.

It is not know how to obtain better bound in terms of w dynamically. This has a close relationship to static data structures of near-linear size (note that all improvements to van Emde Boas discussed about use $O(n^{1+\varepsilon})$ space). There has been convincing progress by Pătraşcu and Thorup in showing an $\Omega(\lg w)$ lower bound for small-space data structures, which would imply the same lower bound for the dynamic case.

5.2 Dictionary

Hash tables solve the dictionary problem in $O(1)$ time, but require randomization. A deterministic data structure was developed by Pagh [4] that supports updates in $\log^{O(1)} n$ time and queries in $(\log \log n)^{O(1)}$ time. It is not even known how to match the predecessor bounds, since van Emde Boas and Raman's dynamic fusion trees are randomized.

5.3 Range Reporting

In the range reporting problem, a set S of n integers is given. For a query of the form $[l, r]$, the goal is to report an element of $S \cap [l, r]$ if any exists. An optimal static range reporting data structure that uses $O(n)$ space and answers queries in $O(1)$ time was developed by Alstrup, Brodal, and Rauhe [1].

A dynamic range reporting data structure that supports $O(\lg w)$ time updates and $O(\lg \lg w)$ time queries (an exponential improvement over van Emde Boas) was developed by Mortensen, Pagh, and Pătraşcu [3].

References

[1] S. Alstrup, G.S. Brodal, T. Rauhe, *Optimal Static Range Reporting in One Dimension*, Proc. 33rd Annual ACM Symposium on Theory of Computing (STOC), 476-482, 2001.

[2] A. Chakrabarti, O. Regev, *An Optimal Randomised Cell Probe Lower Bound for Approximate Nearest Neighbour Searching*, Proc. 45th Annual IEEE Symp. on Foundations of Computer Science (FOCS), 473–482, 2004.

[3] C.W. Mortensen, R. Pagh, M. Pătraşcu, *On Dynamic Range Reporting in One Dimension*, Proc. 37th ACM Symposium on Theory of Computing (STOC), 2005, to appear.

[4] R. Pagh, *A Trade-Off for Worst-Case Efficient Dictionaries*, Nordic Journal of Computing, 7(3):151–163, Fall 2000.

[5] R. Raman, *Improved data structures for predecessor queries in integer sets*, manuscript, 1995, `http://citeseer.ist.psu.edu/raman96improved.html`