CS500: Fundamentals of Algorithm Design and Analysis Fall 2013

Lecture October 16

Prof. David Kirkpatrick *Scribe: NONE*

Unfortunately, there appears to have been no scribe for this class. The following is a brief outline of the topics discussed. Fortunately, they are well covered by material easily available on line.

In this lecture:

- We began by reviewing the material of the previous lecture concerning dictionaries over $\{0, 1, \ldots, m-1\}$, and in particular the limitation of (unaugmented) direct access tables in handling predecessor/successor queries. This included the description of x-fast tries (Willard 1982) that use an auxiliary tree and exploits heap-like addressing to enable binary search on the path from a leaf to the root, to reduce the cost of finding the lowest marked ancestor to $O(\lg \lg m)$. Unfortunately, this still leaves insert/delete operations costing $\Theta(\lg m)$, in the worst case, and only adds (by a constant factor) to the already excessive space requirements of direct access tables.
- To reduce the cost of insertions (and deletions, not discussed), we described the modification called a y-fast trie.
- To reduce the space requirements we discussed hashing; in particular *universal hashing*

1 y-fast tries

- The basic idea here is to cluster the elements of the set S being represented into groups $S_1, S_2, \ldots S_t$, where (i) each group has about $\lg m$ elements, (ii) all elements in S_i are smaller than all elements of S_j, if $i < j$, and (iii) each group is represented as a standard balanced binary search tree.
- We store just one *representative* element from each group in a x-fast trie
- A search now involves (i) finding the nearest representatives to each side of the query, and (ii) searching in the groups associated with those representatives. (Note that this takes $O(\lg \lg m)$ time fore each of these steps, even when the initial query belongs to S.)
- insertions may require some updating of $O(1)$ groups, at a cost of $O(\lg \lg m)$ to do re-balancing operations. Only every $\log n$ steps (at least) will we be required to change the number of groups (and hence the number of representatives). Thus insertions have an *amortized cost* of $O(\lg \lg m)$.

2 Hashing

- Common assumptions about the role of randomness in the design and implementation of hash functions were discussed.

- Went on to discuss the notion of *universal* families of hash functions; and the desire to have *compact* (efficiently encoded/decoded) such families. We mentioned that such families exist; a good treatment is provided in the text by Cormen+.
- We continued by proving/describing some properties of universal hash functions (essentially at the level of detail in Cormen) and mentioned their role in the design of *perfect* has functions.