6.851 Advanced Data Structures (Spring'07)

Prof. Erik Demaine TA: Oren Weimann

Problem 6 – Solution

On Weak d-universal Hash Families. Since $x \neq y$, without loss of generality we assume their first bit is different. Notice that $\Pr[h_M(x) = h_M(y)] = \Pr[M(x-y) = 0]$. The key observation is that for any M such that $M(x-y) = 0$, the first column is determined uniquely once we fix the other $\ell - 1$ columns randomly. Therefore, the probability of a collision is exactly

$$\frac{2^{k(\ell-1)}}{2^{k\ell}} = \frac{1}{2^k} = \frac{1}{m}$$

Deterministic y-fast tries. In the indirection step, we group elements into groups (BSTs) of size $\lg^6 u$ instead of $\lg u$. First notice that queries require $O(\lg\lg u)$ time as our hash function can be evaluated in $O(1)$ time. So we only need to verify that updates cost $O(\lg\lg u)$ as well. An "expensive" insertion (i.e. when splitting or merging BSTs) consists of $\lg u$ hash insertions each taking $\lg^5 u$ time. So every $\Theta(\lg^6 u)$ insertions we have an expensive insertion that costs $\lg^6 u$ time ($O(1)$ amortized). The "cheap" insertions (i.e. when inserting into a BST) cost $O(\lg\lg^6 u) = O(\lg\lg u)$.

Range Existence Queries.

We store the binary prefixes of all elements in S in a trie similar to that of the y-fast trie, but without indirection (hence the $O(n \lg u)$ space). In addition, every trie node stores the minimum and maximum element in its subtree. For $req(a, b)$, we first compute $x = \mathrm{LCA}(a, b)$. This can be done in $O(1)$ time by finding the most significant bit of $a \oplus b$.

- if x is not in the hash table return false.
- otherwise, return true iff the maximum of $x.left\_child$ or the minimum of $x.right\_child$ are in the range $[a, b]$.