6.851 Advanced Data Structures (Spring'12)

Prof. Erik Demaine

$\boxed{\text{Problem 8}}$ *Sample solution*

Signature compression. We will first multiply the input word x by an integer m which will arrange $h_1, h_2, \cdots, h_k$ into a $\lg^2 n$ bit segment, then shift this segment to the least significant bits of the word, and finally mask out all of the other bits. This consists of 3 word-RAM operations, thus it clearly runs in $O(1)$ time.

Let $m = 0^{w/k-\lg n-1}10^{w/k-\lg n-1}1 \cdots 0^{w/k-\lg n-1}1$. $m * x$ consists of each h_i shifted left by $(w/k - \lg n)j$ and added together. In x, h_i appears at position $w/k(i-1)$ so the version of it which is shifted left by $j = k - i$ appears at position $w - w/k - (k+i)\lg n$. And since each block h_i has size $\lg n$, $h_1, \cdots, h_k$ lie adjacent in $m * x$ at positions $w - w/k - (k+1)\lg n$ through $w - w/k - (2k)\lg n$. We can then shift right by $w - w/k - (2k)\lg n + \lg n$ and mask out the result to get our desired $0^{w-k\lg n}h_1h_2\cdots h_k$.

Level ancestors construction. Create the lookup table for all possible subtrees of size $k = 1/4\lg n$ by iterating through the binary string representations of them (of which there are 4^k). For each tree, iterate through each level ancestor query that falls within it (of which there are k^2) and determine the answer by looping up the tree (in $O(k)$ time). Thus the lookup table is constructed in $O(4^k k^3) = o(n)$ time. Prune off the small subtrees, by recursively computing the size of the subtrees rooted at each node.

In order to compute the ladder decomposition, first compute the long-path decomposition, and then iterate through each path and extend it into a ladder in $O(n)$ time. To compute the long-path decomposition, first traverse the tree and compute the maximum depth below each node, as well as a pointer to the maximum depth child of each node. Next build the paths by following these pointers and recursively building the paths from the other children.

To compute the jump pointers from each leaf, we will use the ladder decomposition. All $\lg n$ jump pointers fall within $O(\lg n)$ ladders from the leaf to the root, so simply follow these ladders and extract all of the jump pointers in $O(\lg n)$ time per leaf.