Example Homework Solution

Dan DeBlasio

1 Given a string S and an integer k, find the length and position of the first k (lexicographically) positions of this longest subsequence in S that occurs at least k times in $O(nk)$ time, where $|S| = n$.

Algorithm:

1. Construct the suffix array, a, and the longest common prefix array, lcp, for S.
2. Define two new variables max_ℓ and max_c which will hold the length and starting position in a of the current longest sequence (initialize these variables to 0).
3. For increasing starting positions $1 \leq i \leq n - k$ in a, let $\ell_i = min\{lcp[i], lcp[i+1], ..., lcp[i+k-1]\}$. if $max_\ell < \ell_i$ then set $max_\ell = \ell_i$ and $max_c = i$.

The maximum sequence length will be in max_ℓ, and $a[max_c], a[max_c + 1], ..., a[max_c + k]$ will be the locations in S of the lexicographically minimum substrings that contain this string.

The longest common prefix for a pair of suffixes in a is the minimum of the lcp values of those prefixes, since the suffixes are in lexicographic order that same prefix must be shared by the suffixes in between them. Therefore, by finding the set of $k-1$ locations in lcp that has the largest minimum value, we will have found the longest substring that repeated k or more times. Because the condition in Step 3 requires max_ℓ to be strictly less than ℓ_i, if there are strings of length max_ℓ that occur more than k times, or more than one string that occurs k times, max_c will be set to the minimum position in a where this limit is first encountered. a is in lexicographic order, therefore the returned location is the lexicographic minimum such set of prefixes.

Assume we have run our algorithm and we are at step 4. Let there exist some other substring of length $\ell' > max_\ell$ that appeared k times. The suffixes that begin with that string would be grouped together in a, call the starting position of these suffixes i'. Because all k of these suffixes share an i length prefix, the first pair satisfy the following: $S[a[i']...a[i'] + \ell'] = S[a[i'+1]...a[i'+1] + \ell']$ and $lcp[i'] \geq i$. Similarly $lcp[i'+1] \geq i, ...lcp[i'+k-1] \geq i$. Therefore the min operation in step 3, when $i = i'$ would result in the value $\ell' > max_\ell$ and would have triggered the resetting of the values max_ℓ and max_c. This contradicts the original assumption that the algorithm ran to completion, and therefore a longer substring does not exist.

The running time of step 1 for a string of length n is $O(n)$ as given by Manber and Myers. Steps 2 is an $O(1)$-time operation. Step 3 can be performed in $O(kn)$-time: at each position i in a we compare k values meaning the time for each position is $O(k)$, and we examine $n - k + 1$ starting positions i which is $O(n)$. Therefore the total running time is $O(kn)$.