

**BCB 567/CprE 548**  
**Fall 2007**  
**Exam 2**  
**Solutions**

1. (a) This is true. Consider that  $LCP[i] = k$ . This means that suffix  $SA[i]$  and suffix  $SA[i + 1]$  have a longest common prefix of length  $k$ . This implies that suffixes  $SA[i] + 1$  and  $SA[i + 1] + 1$  have a longest common prefix of  $k - 1$ . This means that the range minimum in the LCP array between indexes  $INV[SA[i] + 1]$  and  $INV[SA[i + 1] + 1]$  is  $k - 1$ . However, for this to be true, there must be some entry in the LCP array equal to  $k - 1$ .

- (b) The maximum zero entries in the LCP is equal to the alphabet size (if you consider the last entry of the LCP array) or alphabet size - 1 if you do not. Think about a dictionary and how many times the first letter changes.

**Grading Notes:** Both  $\Sigma$  and  $\Sigma - 1$  received full credit. Because the alphabet size can be at most  $N$ , the answers  $N$  or  $N - 1$  received partial credit.

- (c) The minimum number of occurrences is 6. For each incoming suffix link, there exists some internal node with path label  $c\alpha$ . Therefore,  $c\alpha$  occurs at least twice. Because there are 3 suffix links, we have at least  $3 \times 2 = 6$  occurrences.

- (d) i. No, counter example: ABC and ABCAB  
ii. No, counter example: ABC and ABCAB  
iii. Yes  
iv. No, counter example: ABC and ABCAB

2. (a) –

- (b) –

- (c) i. To find length 3 repeats in the look-up table, search for table entries with more than one index.  
ii. To find length 3 repeats in the suffix tree, perform the following partial depth first traversal of the tree: For each internal node  $u$  encountered with  $str\text{-}depth(u) \geq 3$ , report the length 3 prefix of  $path\text{-}label(u)$  as an answer do not continue traversing the subtree rooted at  $u$ .

**Grading Notes:** Any reasonable description involving looking for length three paths should receive full credit. If you did not correctly make sure that the path was a repeat, this would have lost points.

The repeats are ACA and CAC

3. (a) To enumerate all right minimal repeats do the following. For each internal node  $u$  (not including the root), look at the parent of  $u$ , which we will call  $v$ . Report a right minimal repeat as  $path\text{-}label(v)$  concatenated with the first character of the edge label from  $v$  to  $u$ .

- (b) The easiest way to enumerate left minimal repeats is to reverse the string and apply the solution to part *a*. The direct solution to the problem follows. Use an bottom-up tree operation to create a variable *count* that stores the number of leaves in the subtree for each internal node. Now, for each suffix link  $u \rightarrow v$ , if  $count(v) > count(u)$  report  $path-label(u)$  as a left minimal repeat.

**Grading Notes:** If you identified the equivalence of the two problems when you reverse the string, you got half credit, even if the rest of the solution was incorrect.

4. (a) First you must find  $j$  where  $SA[j] = i$ . This can be done either by searching the suffix array in  $O(n)$  or using the inverse suffix array ( $INV[i] = j$ ) in  $O(1)$  time. The length of the longest right maximal repeat is the maximum of  $LCP[j]$  and  $LCP[j - 1]$ .

**Grading Notes:** Any answer that considered only one of  $LCP[j]$  and  $LCP[j - 1]$  received half credit.

- (b) This algorithm over counts the number of right maximal repeats. This is because, just as in the dictionary, different pairs of consecutive words might have the same longest common prefix.

**Grading Notes:** Any answer that gave a valid reason why the algorithm was over counting should have received credit.

- (c) To prevent over-counting, keep a stack of values:  $S$ . The algorithm: Set *count* to zero. Scan  $LCP$ , and for each  $i$  do the following: While  $S$  is not empty and  $LCP[i] < S.top$ , pop values off of  $S$ . If  $LCP[i] > S.top$  or  $S$  is empty and  $LCP[i] > 0$ , increment *count* and push  $LCP[i]$  onto  $S$ . (If  $LCP[i] = LCP[S.top]$ , then do nothing as we have already counted this right maximal repeat previously.)

**Grading Notes:** Half credit was given if you identified an additional condition:  $LCP[i] \neq LCP[i - 1]$ , even though this still is not the correct solution.

5. First construct a generalized suffix tree using only the first  $2p_i$  characters of each string  $s_i$ . This takes  $O(\sum p_i)$  time and space. Find the length of the shortest period of  $P$ , which we will call  $p_P$ . This takes  $O(|P|)$  time and space using a suffix tree of  $P$  as described in the solution to homework 4 problem 1.

Search for the pattern  $P$  in the GST constructed in the first step. Modify the normal search algorithm in the following way. If all of the pattern is matched, report that the pattern is found as usual. However, as long as the first  $p_P$  characters of the pattern have been matched, any partial match that ends in at a leaf could also be a match. Any leaf labeled  $(i, j)$  such that  $|S_i| - j + 1 \geq |P|$  and  $p_i = p_P$  will be a valid solution. Therefore after we have matched the first  $p_P$  characters of the pattern, we will consider the \$ symbol to be a match, and we will look at all leaf labels we encounter to check for an occurrence of the pattern.

**Grading Notes:** No credit was awarded if you tried to construct the GST for the full strings  $S_1, S_2, \dots$ . At least half credit was awarded if you identified the need to construct a GST using only a prefix of each string based on the period. Additional credit was awarded if you recognized the usefulness of finding the shortest period of  $P$ . A solution involving repeatedly restarting the pattern match at the root was given partial credit. However, points were lost if you did not account for making sure that each new matching region occurs at consecutive substrings of some string.