Problem 4  Due: Monday, Mar. 12

Be sure to read the instructions on the assignments section of the class web page.

Pattern matching via suffix arrays. Suppose you are given a text $T$ of length $n$ and its suffix array $SA$. Given a query pattern $p$ of length $m$ you would like to know whether $p$ is a substring of $t$. This can clearly be done in $O(m \lg n)$ time by doing a binary search on $SA$. In this question we will see how this time can be reduced to $O(m + \lg n)$.

(a) Recall that the $i$th element $LCP[i]$ in the $LCP$ array is the length of the longest common prefix between the suffixes $SA[i]$ and $SA[i+1]$. We denote this value as $lcp(i, i+1)$. Assume you are given an oracle that, given $i$ and $j$, returns the minimum element in \{LCP[i], LCP[i+1], \ldots, LCP[j]\} in constant time. (In Lecture 16, we will build such an oracle in linear time.) Explain how we can use the oracle to compute $lcp(i, j)$, the length of the longest common prefix between the suffixes $SA[i]$ and $SA[j]$.

(b) Show how to use the oracle to speed up the binary search of a pattern $p$ in $SA$ to obtain $O(m + \lg n)$ query time.

(c) We know that storing $T$ in a suffix tree yields a query time of only $O(m)$. So why would we ever want to keep a suffix array instead?