Comp 555 Problem Set 4

Elliott Hauser

November 13, 2012

Question 1

a

While both Hash-table and Suffix trees would correctly solve this problem, they may be prefered over one another depending on the circumstances. Suffix trees have a run time of $O(m)$, making them preferable *ceteris paribus.* But there is computational and engineering overhead to generate the suffix tree. Hash tables are implemented natively in many programming languages (such as Python's dict structure), making them easier to implement in some situations.

That said, for our purposes computational and engineering overhead are negligible when compared to runtime savings, especially given that n is likely to be very long. So, the suffix tree structure is most appropriate for our purposes here.

b

Assuming the suffix table for m is already constructed as a graph, an algorithm to find the number of occurrences of m in n is

```
Preprocess $n$ into suffix tree $s_n$ in $O(n)$.

SuffixSearch($m$, $s_n$)
position = 0
for edge in childEdge($s_n$) == $m$[position]
    if descendent vertex has child edge == $m$[position +1]
        SuffixSearch($m$[position+1:len($m$)], descendants(edge)
    else
        matches = count(descendents)
        return matches
```

Question 2

Question 3

Question 4

Question 5

Question 6: Programming exercise