Weekplan: Suffix Sorting

Philip Bille and Inge Li Gørtz

References and Reading

[1] Linear work suffix array construction, J. Kärkkäinen, P. Sanders, S. Burkhardt, J. ACM, 2006.

[2] Scribe notes from MIT.

[3] Algorithms on Strings, Trees, and Sequences, Chap. 5-9, D. Gusfield

[4] On the sorting-complexity of suffix tree construction, M. Farach-Colton, P. Ferragina, S. Muthukrishnan, J. ACM, 2000

We recommend reading [1] and [2] in detail. [3] provides an extensive list of applications of suffix trees and [4] is the first suffix-tree construction algorithm matching the sorting time bound.

Exercises

1 LSD and MSD Radix Sort Radix sort that process digits in right-to-left order is called *LSD radix sort*. If we instead process digits in left-to-right order we call the algorithm *MSD radix sort*. Solve the following exercises.

1.1 Show that LSD radix sort correctly sorts any input.

1.2 Explain why each step in LSD radix sort must use a stable sorting algorithm.

1.3 Show that that are input for which MSD radix sort does not correctly sort any input.

1.4 [$*$] Explain how to modify MSD radix sort to sort correctly.

2 [w] **Prefix Doubling** Suffix sort `cocoa` using prefix doubling.

3 Odd-Even Sampling Suppose we modify the sampling of suffixes in the DC3 algorithm such that the sampled and non-sampled suffixes are those starting at even and odd positions, respectively. Determine if the algorithm still works, i.e., show that it still works or explain where it fails.

4 [w] **Suffix arrays** Write the suffix array for the string `mississippi$`.

5 Searching in Suffix Arrays Let S be a string of length n and let SA be the suffix array of S. Given the SA and S show how to support search(P) for a string P of length m in time $O(m \log n + \text{occ})$.

6 LCP array Let S be a string of length n, let ST be the suffix tree of S and let SA be the suffix array of S.

The LCP array LCP(S) is an array of length n, where $\text{LCP}[0] = -1$ and $\text{LCP}[i]$ is the length of the longest common prefix of the suffix $SA[i-1]$ and $SA[i]$ for $i \leq 2 \leq n$.

6.1 [w] Write the LCP array for the string `mississippi$`.

6.2 Show how to obtain the LCP array from the suffix tree. *Hint:* Consider the stringdepth of the internal nodes.

6.3 Given two indicies i and j show how to efficiently compute the length of the longest common prefix of two suffixes $SA[i]$ and $SA[j]$ using the LCP array.

7 Approximate String Matching with Hamming Distance The *Hamming distance* between two equal length strings S_1 and S_2 is the number of positions i such that $S_1[i] \neq S_2[i]$. Let P and S be strings over alphabet Σ of lengths m and n, respectively. Given a parameter k, show how to compute all ending positions of substrings in S whose Hamming distance to P is at most k. *Hint:* Suffix trees and longest common extensions.

8 Faster Search in Suffix Arrays Let S be a string of length n and let SA be the suffix array of S. In this exercise we will improve the time for search(P) for a string of length m from $O(m \log n + \text{occ})$ to $O(m + \log n + \text{occ})$. A comparison of character in P is *redundant* if the character has been compared before. The idea in the speed up is to reduce the number of redundant comparisons to at most one in each iteration.

Let L be the left boundary in our binary search and let R be our right boundary (initially $L = 0$ and $R = n-1$). In each iteration we query the position $M = \lceil (R+L)/2 \rceil$ and update L and R accordingly.

Let $l = \text{lcp}(L, P)$ and $r = \text{lcp}(R, P)$. In the beginning of the search we explicitly compare P to the suffixes $SA[1]$ and $SA[n]$ to find l and r.

8.1 Let $min = \min(l, r)$. Argue that all suffixes in $SA[L, R]$ has a longest common prefix with P of length at least min. Explain how to use this to speed up the binary search. This does not change the worst case bounds – why?

8.2 We can use $\text{lcp}(L, M)$ and $\text{lcp}(R, M)$ to obtain better worst case bounds. Assume $l > r$. Explain what the algorithm should do in each of the following cases:

- $\text{lcp}(L, M) < l$.
- $\text{lcp}(L, M) > l$.
- $\text{lcp}(L, M) = l$.

8.3 Given the S, SA and LCP show how to support search(P) for a string P of length m in time $O(m + \log n + \text{occ})$.

9 Suffix Tree Construction Bounds Solve the following exercises.

9.1 [$*$] Show that any algorithm for suffix tree construction of a string of length n over an alphabet Σ must use $\Omega(\text{sort}(n, |\Sigma|))$ worst-case time. *Hint:* Show that an algorithm using $o(\text{sort}(n, |\Sigma|)$ time would lead to a contradiction.

9.2 [$*$] Suppose that we drop the requirement that sibling edges are sorted from left-to-right. Show how construct such a suffix tree in $O(n)$ expected time. *Hint:* hash.