Weekplan: Suffix Trees
Philip Bille

References and Reading

[2] Scribe notes from MIT.


Exercises

1 Suffix Trees
Solve the following exercises.

1.1 [w] Draw the suffix tree $T$ for the string cocoa$$. Write edge labels (substrings) and leaf labels (suffix number).

1.2 [w] Add string depth for each node in $T$. Verify that the length of the longest common prefix of suffixes cocoa and coa is the string depth of the NCA/LCA of the corresponding leaves in $T$.

2 [w] Substring Counting Let $S = s_0s_2\cdots s_{n-1}$ be a string of length $n$ over an alphabet $\Sigma$. We are interested in a data structure for $S$ that supports the following query.

- $\text{count}(P)$: return the number of occurrences of $P$ in $S$.

Give a data structure that supports $\text{count}(P)$ queries efficiently.

3 Common Substrings and Repeats
Solve the following exercises. Assume you have an efficient black-box algorithm for computing the suffix tree of a string.

3.1 A repeat in a string $S$ is a substring $R$ that occurs at least twice in $S$. Show how to efficiently compute the length of a longest substring of $S$ that is a repeat.

3.2 Given strings $S_1$ and $S_2$ a longest common substring is a substring of both $S_1$ and $S_2$ of maximal length. Show how to efficiently compute the length of a longest common substring of $S_1$ and $S_2$.

4 Suffix Trees for Multiple Strings
The suffix tree for a set of strings $S_1, \ldots, S_k$ of total length $n$ over alphabet $\Sigma$ is the compact trie of all suffixes of the strings $S_1s_1, S_2s_2, \ldots, S_k s_k$. Each $s_i$ is a special character not in $\Sigma$. The label of a leaf is a pair $(i, j)$ such that the string to $(i, j)$ is suffix $j$ of string $S_i$. Suppose you have an efficient black-box algorithm for computing the suffix tree of a single string. Show how to use this algorithm to construct the suffix tree for $S_1, \ldots, S_k$ efficiently.

5 Restricted Suffix Search
Let $S$ be a string of length $n$ over alphabet $\Sigma$. Give an efficient data structure for $S$ that supports the following query:

- $\text{rsearch}(P, i, j)$: report the starting positions of occurrences of string $P$ in $S[i, j]$.
6 [w] **Prefix Doubling**  Suffix sort cocoa using prefix doubling.

7 **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.

8 **Suffix Arrays**  Let $S$ be a string of length $n$. The *suffix array* $SA$ of length $n + 1$ containing the left-to-right sequence of labels of leaves in the suffix tree. Given the $SA$ and $S$ show how to support search($P$) for a string $P$ of length $m$ in time $O(m \log n + \text{occ})$.

9 **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 $\Sigma$ 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:* Longest common extensions.

10 **Suffix Tree Construction Bounds**  Solve the following exercises.

10.1 [*] Show that any algorithm for suffix tree construction of a string of length $n$ over an alphabet $\Sigma$ 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.

10.2 [*] 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.