Suffix Arrays

A suffix array is a sorted array of all suffixes of a given string.

Overview

Suffix arrays are a simple, space efficient alternative to suffix trees.

Definition

Let $S = S[1]S[2]...S[n]$ be a string and let $S[i, j]$ denote the substring of S ranging from i to j. The suffix array A of S is defined as an array of integers providing the starting positions of suffixes of S in *lexicographical* order. This means, an entry $A[i]$ contains the starting position of the i^{th} smallest suffix in S and thus for all $1 < i \leq n : S[A[i-1], n] < S[A[i], n]$.

Each suffix of S shows up in A exactly once.

Consider the text $S =$ banana$

i	1	2	3	4	5	6	7
$S[i]$	b	a	n	a	n	a	$

The text ends with the special sentinel letter $ that is unique and lexicographically smaller than any other character.

Suffix	i
banana$	1
anana$	2
nana$	3
ana$	4
na$	5
a$	6
$	7

Sorting

Suffix	i
$	7
a$	6
ana$	4
anana$	2
banana$	1
na$	5
nana$	3

The suffix array A contains the starting positions of these sorted suffixes:

i	1	2	3	4	5	6	7
$A[i]$	7	6	4	2	1	5	3

So for example, $A[3]$ contains the value 4, and therefore refers to the suffix starting at position 4 within S, which is the suffix `ana$` .

Algorithm

Suffix Arrays are closely related to *Suffix Trees*:

- *Suffix Arrays* can be constructed by performing a depth-first traversal of a suffix tree. If edges are visited in the lexicographical order of their first character than suffix array values corresponds to the leaf-labels given in the order in which these are visited.
- A *Suffix Tree* can be constructed in linear time by using a combination of suffix array and ***LCP array***.

  > The *Longest Common Prefix* array (LCP array) is an auxiliary data structure to the suffix array. It stores the lengths of the longest common prefixes (LCPs) between all pairs of consecutive suffixes in a sorted suffix array.

  For example, if $A = [aab, ab, abaab, b, baab]$ is a suffix array, the longest common prefix between $A[1] = aab$ and $A[2] = ab$ is a which has length 1, so $H[2] = 1$ in the LCP array H. Likewise, the LCP of $A[2] = ab$ and $A[3] = abaab$ is ab, so $H[3] = 2$.

Every suffix tree algorithm can be systematically replaced with an algorithm that uses a suffix array enhanced with additional information (such as the LCP array) and solves the same problem in the same time complexity.

Advantages of suffix arrays over suffix trees include improved space requirements, simpler linear time construction algorithms (e.g., compared to Ukkonen's algorithm) and improved cache locality.

Time and Space Complexity

A naive approach to construct a suffix array is to use a comparison-based sorting algorithm which require $O(n \log n)$ suffix comparisons, but a suffix comparison runs in $O(n)$ time, so the overall runtime of this approach is $O(n^2 log(n))$.

A suffix tree can be built in $O(n)$ and can be converted into a suffix array by traversing the tree depth-first also in $O(n)$, so there exist algorithms that can build a suffix array in $O(n)$ time.

Applications

- The suffix array of a string can be used as an index to quickly locate every occurrence of a substring pattern P within the string S.
- Longest repeated substring. An application of sorting to computational biology and plagiarism detection.

References

- Suffix Array Wikipedia
- Robert Sedgewick Notes