Strings (2/2)

Prof. Rhadamés Carmona

Last revision: 06/02/2020

Content

- Suffix array

Suffix Array

- Given a text S of length n, the suffix array for S, is an array (suftab or suffix table) of integers specifying **the lexicographic ordering of the suffixes of the string S**.
- Before sorting the suffixes, the i-th suffix of S is S[i..n-1]. Something like &S[i]

Suffix Array

- Example: S = abaab

Suffixes:	**Sorted suffixes**:	**Suffix array**:
0 abbab	2 aab	2
1 baab	3 ab	3
2 aab	0 abbab	0
3 ab	4 b	4
4 b	1 baab	1

Suffix Array

- Data structure of choice for many, if not all, of the string processing problems to which suffix tree methodology is applicable
- Any problem whose solution can be computed using suffix trees is solvable with the same asymptotic complexity using suffix arrays
- Suffix arrays are easier to build and to use!

Creating the suffix Array

- Naïve approach: The integer table can be obtained in $O(n^2 \log n)$ by sorting, due to $O(n \log n)$ comparisons of $O(n)$ length.

```
char S[MAX_N];
int SA[MAX_N], n;
bool cmp(int a, int b)
{
        return strcmp(&S[a], &S[b])<0;
}
…
n = strlen(S);
for(int i=0; i<n; i++) SA[i] = i;
::sort(SA, SA+n, cmp);
```

Creating the suffix Array

- Radix-sort: move every suffix index of the array into a bucket of 26 array positions (or more than 26…) using the first character of each substring. Thus, in O(n), we put each suffix index in the corresponding entry in the array. Then, on each entry, we do the same using the second character, and so on.
- It is O(n log n) instead of O(n^2 log n).

Creating the suffix Array

GATAGACA$ (0)
ATAGACA$ (1)
TAGACA$ (2)
AGACA$ (3)
GACA$ (4)
ACA$ (5)
CA$ (6)
A$ (7)
$ (8)

First char only →

$ (8)
ATAGACA$ (1)
AGACA$ (3)
ACA$ (5)
A$ (7)
CA$ (6)
GATAGACA$ (0)
TAGACA$ (2)
GACA$ (4)

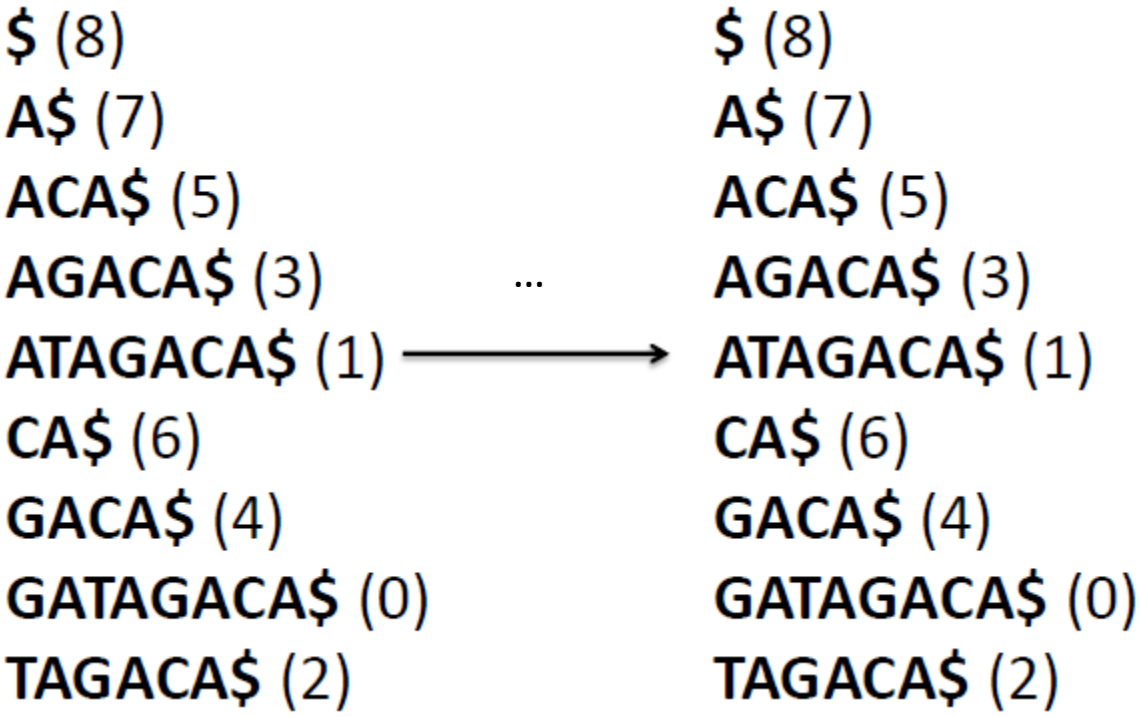
Creating the suffix Array

$ (8)
ATAGACA$ (1)
AGACA$ (3)
ACA$ (5)
A$ (7)
CA$ (6)
GATAGACA$ (0)
TAGACA$ (2)
GACA$ (4)

Second char on each bucket →

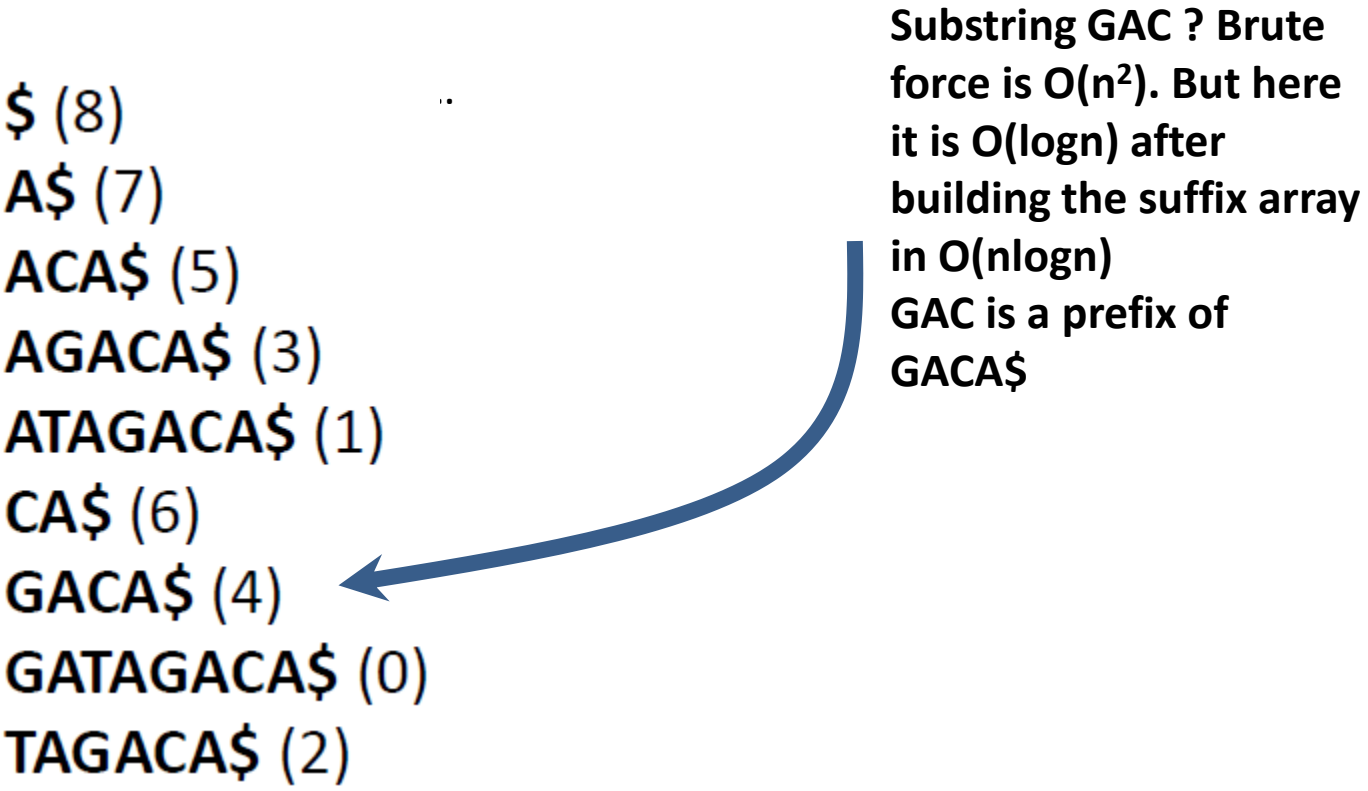
$ (8)
A$ (7)
ACA$ (5)
AGACA$ (3)
ATAGACA$ (1)
CA$ (6)
GACA$ (4)
GATAGACA$ (0)
TAGACA$ (2)

Creating the suffix Array



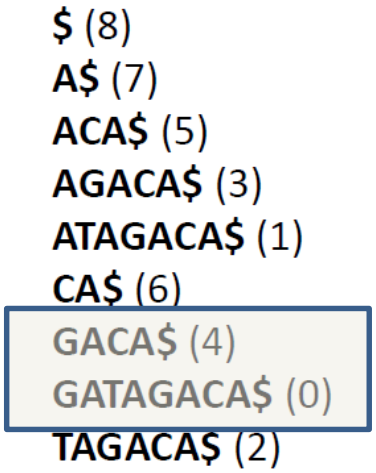
Some applications

- Seach for a substring: it is reduced to a binary search.



Some applications

- Locate every occurrence of a substring pattern P within the string S = finding every suffix that begins with P = 2 binary searches for P (one lower bound and one upper bound)



Substring GA in GATAGACA$?

Some applications

- Longest common prefix of two substrings (LCP), Kasai's algorithm
- Number of different substrings: using LCP
- Longest repeated substring problem
- …
- Recommended link:

  https://www.geeksforgeeks.org/suffix-array-set-1-introduction/

Exercises

https://www.spoj.com/problems/SUBST1/

https://www.spoj.com/problems/DISUBSTR/

https://www.spoj.com/problems/SARRAY/

https://www.spoj.com/problems/BEADS/

https://www.spoj.com/problems/SUBLEX/

https://www.spoj.com/problems/ADASTRNG/

https://www.spoj.com/problems/LCS2/

https://www.spoj.com/problems/ADAPHOTO/