Technische Universität München
Fakultät für Informatik
Lehrstuhl für Effiziente Algorithmen
Prof. Dr. Ernst W. Mayr
Chintan Shah, Chris Pinkau, Jeremias Weihmann

Wintersemester 2012
Aufgabenblatt 10
17.12.2012

Praktikum Algorithmen-Entwurf

Due Date: Monday, 7th January 2013, 12:00

Aufgabe 1 (Suffix arrays)

Write a program for the construction of a suffix array for a given input text X of length n in time $O(n \log n)$. Use a little memory space as possible. Moreover, write the algorithm for searching the text X for a search pattern Y of length m to run in time $O(m \log n)$. If the pattern occurs in X, your program should display the number of occurrences as well as the positions of the first and last occurrences in X. These are the positions with the smallest and largest suffix, with respect to the lexicographic ordering. These correspond to the entries you find with binary search in the **Pos** array.

Your program must not be implemented with LEDA. The running time for the construction of the suffix array is to be displayed, as well as the running times of the individual searches for the patterns. Check your implementation with these values and determine whether it really grows linearly in the length of the text, or the length of the search pattern, respectively.

The input text consists of arbitrary ASCII characters (of type `char`), hence the size of the alphabet is 256. The character `$` is the end-of-file symbol and occurs only at the end of texts. As example inputs you may use the text files `text1.txt` to `text6.txt` and the pattern files `text1.pat` to `text6.pat` (as well as the corresponding solution files `text1.sol` to `text6.sol`). Every text file contains a text, followed by `$`. The pattern files contain a bunch of patterns, all followed by a `$` and a line break. Your program receives the names of the input file and the pattern file as command line parameters; the output is displayed on the screen.

Hints

Consider the example of a correct output for the algorithm on the next site.

Example for input and output of the algorithm

Input:

```
Heute programmieren wir
einen Algorithmus zur
Suche in Texten.$
te$
wir
ein$
xx$
```

Output:

```
Suffix-Array: Zeit 0.000238 Sek
Muster 1: 2 Vorkommen
          lex. kleinste Pos. 4, lex. groesste Pos. 59
          Zeit: 0.000012 Sek
Muster 2: 1 Vorkommen
          lex. kleinste Pos. 21, lex. groesste Pos. 21
          Zeit: 0.000006 Sek
Muster 3: 0 Vorkommen
          Zeit: 0.000002 Sek
```