Technische Universität München
Institut für Informatik
Prof. Dr. Ernst W. Mayr

SS 1998
Aufgabenblatt 9
13. Juli 1998

Praktikum Algorithmen-Entwurf

(Abgabetermin: Montag, den 20. Juli 1998, 24.^{00}Uhr)

Suchen in Texten

Aufgabe 1 Suffix-Bäume

Schreiben Sie ein Programm, das für einen Eingabetext X der Länge n mit McCreight's Algorithmus in linearer Zeit $O(n)$ einen Suffix-Baum konstruiert und damit für Muster Y der Länge m in Zeit $O(m)$ prüft, ob das Muster im Text vorkommt, und gegebenenfalls die Anzahl der Vorkommen sowie die Position des ersten und des letzten Vorkommens ausgibt.

Ihr Programm soll nicht animiert werden; stattdessen soll die Laufzeit für die Konstruktion des Suffix-Baums sowie die Laufzeit für die Bearbeitung jeden einzelnen Musters ausgegeben werden. Prüfen Sie anhand der ausgegebenen Zeiten, daß die Laufzeit Ihrer Implementierung tatsächlich linear mit der Textlänge bzw. mit der Musterlänge wächst. Zur Zeitmessung können sie die Funktion `clock()` aus `time.h` verwenden, die die verbrauchte CPU-Zeit seit dem ersten Aufruf der Funktion zurückliefert (Einheit: `CLOCKS_PER_SEC`).

Der Eingabetext besteht aus beliebigen ASCII-Zeichen (Typ `char`), die Alphabet-Größe ist also 256. Das Zeichen `$` ist das Schlußzeichen und kommt ausschließlich am Ende des Textes vor. Die Beispieleingaben stehen in den Textdateien `text1` bis `text6` im Verzeichnis `/usr/local/lea/praktikum/suffix` zur Verfügung. Jede solche Datei enthält zuerst einen Text, abgeschlossen mit `$`. Darauf folgt ein Zeilenvorschub und dann eine Reihe von Mustern, die jeweils von einem `$` und einem Zeilenvorschub gefolgt sind. Korrekte Beispielausgaben sind in den gleichnamigen Dateien mit Endung `.sol` zu finden.

Der Algorithmus soll in C++ *ohne* Verwendung von LEDA implementiert werden. Die ausgehenden Kanten von einem Knoten im Baum können zum Beispiel in einem Feld mit Größe gleich der Alphabetgröße oder in einer verketteten Liste gespeichert werden. Die erste Möglichkeit gibt konstante Zugriffszeit für jede Kante aus einem Knoten, ist aber (für größere Alphabete) aus Platzgründen nicht zu empfehlen. Suchen in der verkettete Liste kann mittels linearem Suchen implementiert werden, da die Alphabetgröße als konstant angenommen werden kann und die Gesamtlaufzeit daher linear bleibt. Es ist aber natürlich erlaubt, bessere Verfahren zu implementieren!

Vergessen Sie nicht, Ihr Programm auch so zu kommentieren, daß die gewählte Repräsentation der Knoten des Suffix-Baums leicht nachzuvollziehen ist. Prüfen Sie die Korrektheit Ihrer Implementation nach, indem Sie Ihr Programm testen lassen, daß Y wirklich in den vom Algorithmus berechneten ersten und letzten Positionen vorkommt; die Korrektheit der ausgegebenen Anzahl von Vorkommen braucht aber nicht überprüft werden.

Beachten Sie das Beispiel für eine korrekte Ausgabe des Algorithmus auf der Rückseite dieses Aufgabenblattes.

Beispiel für Eingabe und Ausgabe des Algorithmus

Eingabe:

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

Ausgabe:

```
Suffix-Baum: Zeit 0.5 Sek
Muster 1: 2 Vorkommen, erste Pos. 4, letzte Pos. 59, Zeit: 0.1 Sek
Muster 2: 1 Vorkommen, Pos. 21, Zeit: 0.2 Sek
Muster 3: 0 Vorkommen, Zeit: 0.1 Sek
```