DAP2 Praktikum für ETIT und IKT, SoSe 2018, Kurzaufgabe K4

Fällig am 04.06. um 17:30

Die Regeln für das Programmieren gelten in gewohnter Härte. Ab jetzt wird auch hässlicher Code bestraft. Assertions sind heute nicht notwendig.

In den bisherigen Übungen haben wir für Felder u.Ä. die STL-Container benutzt. Heute wollen wir ausnahmsweise noch mal den Umgang mit Zeigern üben.

Kurzaufgabe K4 (2 Punkte)

Die längste gemeinsame Teilfolge (Longest Common Subsequence) zweier Folgen *a*, *b* ist eine Folge *c* = lcs(*a*,*b*) für die gilt:

1. *c* ist sowohl eine Teilfolge von *a* als auch eine Teilfolge von *b* und
2. es gibt keine andere Teilfolge von *a* und *b*, welche länger als *c* ist.

Bitte beachten Sie, dass die LCS nicht aus direkt aufeinanderfolgenden Zeichen der beiden Sequenzen bestehen muss. In der Vorlesung wurde ein Algorithmus vorgestellt, der die längste gemeinsame Teilfolge zweier Zeichenfolgen unter Verwendung der Technik der dynamischen Programmierung bestimmt. Implementieren Sie diesen Algorithmus und bauen ihn als Funktion `string LCS(const string a, const string b)` in das bereitgestellte Programmgerüst ein. Danach stellen Sie die Entwicklung der Laufzeit Ihrer Implementierung in Abhängigkeit von der Länge der Zufallsfolgen mit Matlab grafisch dar. Beide Folgen sollen hier die gleiche Länge aufweisen. Bitte beachten Sie:

- Ihre Funktion darf nicht davon ausgehen, dass beide Strings gleich lang sind.
- Der zu implementierende Algorithmus benötigt ein zweidimensionales Array. Implementieren Sie das Array als `int **Field;` , so dass Sie mit z.B. `Field[15][26]` darauf zugreifen können. Der komplette Speicher für dieses Array ist dynamisch zur Laufzeit mit `new` zu allozieren. Da dies von Compilern nur teilweise unterstützt und nicht dem Standard entspricht, ist eine direkte Allokation der Form `int **Field = new int[n][m]` nicht erlaubt.
- Die LCS ist nicht immer eindeutig, nur Ihre Länge ist eindeutig.

- Beispiele für `lcs` sind:

```
>lcs
Usage:
lcs [ -s Scatter ] [ -r Seed ] [ -l m ] [ -o ] [ -t ]  n

Options:
-s Scatter : Integer scatter>= 0, gives speed of variation in random
strings
-r Seed    : Integer seed>0 for random generator, default: 1,
overrides random
-l m       : Optional length of second string, m>0
-o         : Activates output of random strings
-t         : Measure time for LCS
n          : n>0, Length of random strings
```

Options may occur in any order.

```
>lcs 10
LCS    : baaZaaZ

>lcs 10 –o
First  : abcbcbccbb
Second : babcbcdded
LCS    : abcbc

>lcs 1000 -s 400
LCS    : EtPvzjSyBdRjOfCGPHxzufIXuxJINanVUxKXFxjGVtVYDEUrFLVoIvV
nEWWxMsOjmahOVMVJgWglsplgWNwQDvsSQCEtjwcOROWkgBGxKhqpfyzoYpqJjhP
NJOAAFndzGgnYAGeNmEYZoOPKNQaVeijchcYMTPPTLzIuBRGTvxjBpvdHTYeeCHH
RyPQQFARIFnwRvyCFrhPUXvFUgteWZ

>REM Your Software should produce exactly the same output
>REM if you enter the following line:
>REM
>lcs -s 1 -r 1 -l 15 -t -o 20
First  : abcccddcbbcdeeddeefe
Second : baZYZYYYXYZabab
LCS    : abb
It took 0 Seconds to compute.
```