Datenstrukturen und Algorithmen, SS 2024
Prof. Dr. P. Rossmanith
D. Cloerkes, M. Gehnen, D. Mock, N. Speetzen

Datum: Abgabe: 19.04.2024

Übungsblatt 01

Aufgabe T1

Ordnen Sie die folgenden Funktionen in der Reihenfolge Ihres asymptotischen Wachstums (ohne lange Nachzudenken). Mit $\log n$ bezeichnen wir den Logarithmus zur Basis 2.

- $\log(n)$
- $\log(n^5)/\log\log(n)$
- $n^{\log\log n}$
- $\log(n)^{\log n}$
- n^2
- $n \log n$
- 2^n
- $\sqrt{n}$

Aufgabe T2

Beweisen oder widerlegen Sie, dass $\frac{1}{2}n^2 + 7n + 14 = O(n^2)$ gilt.

Aufgabe T3

Sie haben zwei *einfach* verkettete Listen gleicher Länge. Wie können den *zip* dieser Listen berechenen? Der *zip* zweier Folgen $a_1, \ldots, a_n$ und $b_1, \ldots, b_n$ ist einfach

$$a_1, b_1, a_2, b_2, \ldots, a_n, b_n.$$

Natürlich kann man einfach beide Listen durchgehen und eine neue List erzeugen. Falls man die alten Listen aber nicht mehr braucht, gibt es eine bessere Methode.

Aufgabe H1 (10 Punkte)

Beweisen Sie formal mithilfe der Definition der O-Notation, dass

$$\sqrt{2n(n+2)(n-1)} = \Theta(n^{3/2}).$$

Aufgabe H2 (10 Punkte)

Tragen Sie für jede Kombination in der Tabelle ein, wie sie sich im Sinne der O-Notation zueinander verhalten.

Wenn f die Funktion einer Zeile und g die Funktion einer Spalte ist, dann finden Sie heraus ob $f = O(g)$, $f = \Omega(g)$ oder gar $f = \Theta(g)$ git. Tragen Sie dann O, Ω oder Θ in das entsprechende Feld ein. (Wenn mehrere Symbole richtig sind, wählen Sie das aussagekräftigste.)

	$n!$	$\log(n!)$	$\log(n^n)$	n^n
$n!$				
$\log(n!)$				
$\log(n^n)$				
n^n				

Aufgabe H3 (10 Punkte)
Neulich hat Matthias bemerkt, dass Autoren auch faule Menschen sind und Wörter mehrfach benutzen. Um sich ein Bild davon zu machen, wie verbreitet dieses Phänomen ist, beschließt er, die Häufigkeit jedes Wortes zählen zu lassen, und stellt euch daher folgende Aufgabe:
In einer doppelt verketteten Liste soll jedes Wort als Schlüssel gespeichert werden, das dazugehörige Datum die Anzahl des Wortes sein. Folgende Strategien sind denkbar:

- Naiv: Wenn ein Wort in der Liste gefunden wird, wird dort der Zähler um eins erhöht. Wird das Wort nicht gefunden, wird es an die hinten Liste angehangen.
- Move to Front: Wenn ein Wort in der Liste gefunden wird, wird der Zähler um eins erhöht und das Wort an den Anfang der Liste verschoben. Wird das Wort nicht gefunden, wird es vorne in der Liste eingefügt.

Schreiben Sie dazu je ein Programm, dass die benötigten Listenoperationen zählt. Der Einfachheit nehmen wir an, dass jedes Löschen, Einfügen, sowie ein Schritt auf das nächste Element je eine Operation ist. Was beobachten Sie bei der Anwendung auf die drei beigefügten Texte?
Konkret sind die Aufgaben:

- Schreiben Sie ein gut lesbares, kommentiertes Programm für beide Verfahren in einer Standardsprache (Java, C++, Python, ...), und hängen Sie es ihrer Lösung an.
- Wenden Sie beide Programme es auf die drei Texte an: Wieviele Operationen benötigen sie jeweils?
- Begründen Sie das erwartete Verhalten.

Bemerkung: Jeder der drei Texte enthält in der ersten Zeile die Anzahl der Wörter n, gefolgt von n Wörtern in je einer eigenen Zeile. Die Texte sind um Sonderzeichen und Groß/Kleinschreibung bereinigt. Ein Lösungsvorschlag wird in Java veröffentlicht.

Aufgabe H4 (Zusatzaufgabe)
Finden Sie möglichst einfache Funktionen f.

a) $\displaystyle\sum_{k=1}^{n} k^{7/2} = f(n) + O(n^4)$

b) $(n+1)^n = en^n\big(1 + f(n) + O(n^{-2})\big)$

c) $\displaystyle\prod_{k=n}^{2n}\Big(1 + \frac{1}{k^2}\Big) = f(n) + O(1/n^2)$