Hochschule
Albstadt-Sigmaringen
Albstadt-Sigmaringen University

Technische Informatik
Prof. Dr. Tobias Häberlein

Algorithmik
SS 12

Aufgabenblatt 2

(Bearbeitungszeit: 3 Praktikumstermine)

Aufgabe 2.0. **5 Punkte**

In der Vorlesung wurde folgende rekursive Implementierung von InsertionSort vorgestellt:

def *insertionSortRek*(*l*):
  if *len*(*l*)≤1: **return** *l*
  else: **return** *ins*(*insertionSortRek*(*l*[1:]), *l*[0])

(a) Implementieren Sie – ebenfalls unter Verwendung von *ins* – eine iterative Variante von *insertionSortRek*.

(b) Verwenden Sie Pythons *timeit* und Pythons *random* Module um die Laufzeit ihrer iterativen Variante und die Laufzeit des rekursiven Algorithmus zu vergleichen; Verwenden Sie für den Vergleich Listen der Länge 500 mit zufälligen Einträgen zwischen 0 und 10^6.

Aufgabe 2.1. **10 Punkte**

(a) Implementieren Sie eine randomisierte Variante von Quicksort

quicksortRandomisiert(*lst*, *l*, *r*)

die eine Häufung ungünstiger Fälle dadurch vermeidet, dass das Pivot-Element der Partitionierung von *lst*[*l*:*r*+1] zufällig aus den Indizes zwischen (einschließlich) *l* und *r* gewählt wird.

(b) Implementieren Sie eine weitere randomisierte Variante von Quicksort

quicksortMedian(*lst*, *l*, *r*)

die das Pivotelement folgendermaßen wählt: Es werden zunächst drei zufällige Elemente aus der zu partitionierenden Liste (also aus *lst*[*l*:*r*+1]) gewählt. Als Pivot-Element wird der Median – also das mittlere der zufällig gewählten Elemente – ausgewählt.

Aufgabe 2.2. **10 Punkte**

Implementieren Sie eine rekursive nicht-destruktive Variante des Mergesort-Algorithmus. Bei Mergesort ist der *divide*-Schritt einfach: hier wird die Liste einfach in der Mitte geteilt. Der eigentliche Aufwand steckt hier im *combine*-Schritt, der die beiden rekursiv sortierten Listen zu einer großen sortierten Liste kombinieren muss. Dies geschieht im „Reißverschlussverfahren“: die beiden schon sortierten Listen müssen so ineinander verzahnt werden, dass daraus eine sortierte Liste entsteht. Dies wird in der englischsprachigen Literatur i. A. als *merging* bezeichnet.

Aufgabe 2.3. **10 Punkte**

Implementieren Sie ein Kombination von Insertionsort und Quicksort. Eine Liste der Länge größer

100 soll über Quicksort sortiert werden; statt des Rekursionsabbruch bei Listenlänge ≤ 0 soll die Rekursion jetzt bei ≤ 100 abbrechen und dann weiter über Insertionsort sortiert werden.

Aufgabe 2.4. **10 Punkte**

Verwenden Sie Pythons *timeit*-Modul um mit Listen der Länge 10^6 und zufälligen Einträgen aus $\{0, \ldots, 10^5\}$ die die schnellste Sortiervariente zu testen. Ein Cola für den Semester-schnellsten Sortieralgorithmus... (natürlich ohne Verwendung von Pythons internen Sortierroutinen).