Übung 2 - Musterlösung Abgabe 8.05.2019

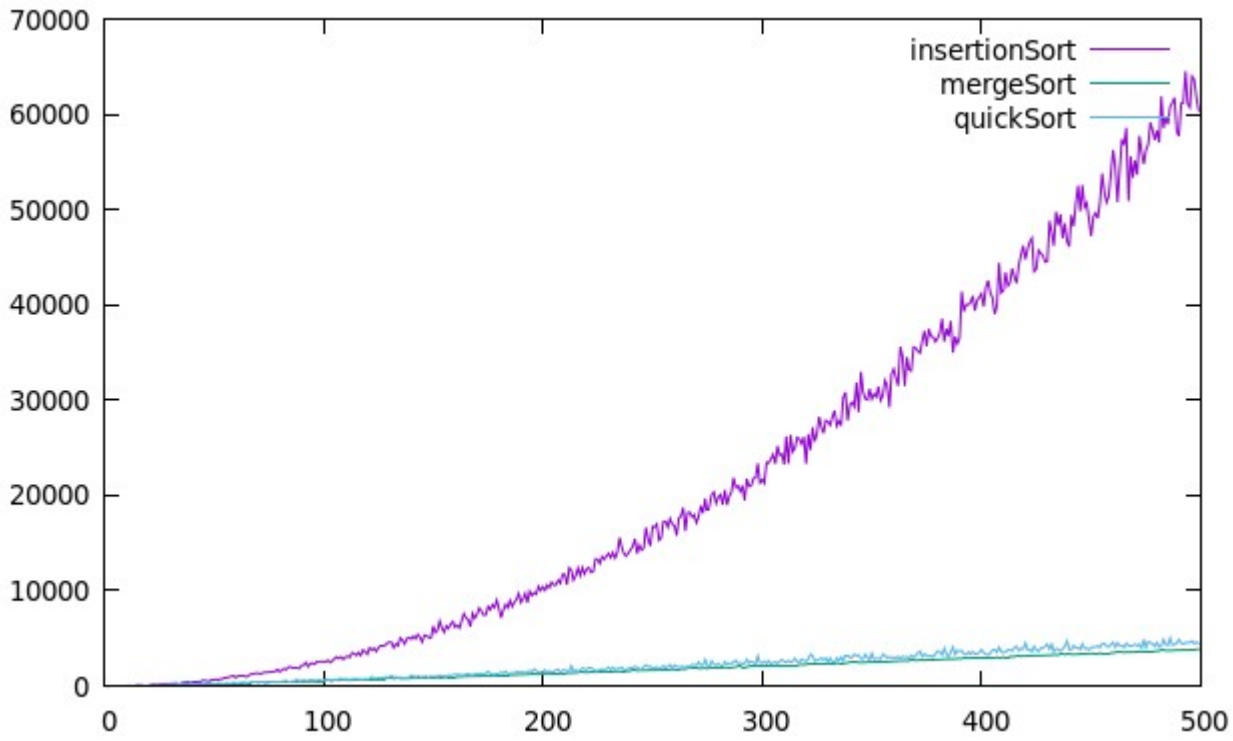
Aufgabe 3 – Vergleichen und Testen von Sortierverfahren 20 Punkte

Anmerkung: Die Lösung beruht auf einer älteren Version der Aufgabe, bei der statt insertionSort() mit selectionSort() gearbeitet wurde. Die Resultate gelten aber entsprechend.

a) Eine Möglichkeit der Implementation der Algorithmen befindet sich im Anhang. Das Plotten kann durch Gnuplot nach Ausführen von `sort.py` folgendermaßen geschehen:

```
gnuplot> set datafile separator ","
gnuplot> plot [0:500][0:70000] 'varyN.txt' using 1:2 with lines title 'insertionSort',\
>'varyN.txt' using 1:3 with lines title 'mergeSort',\
>'varyN.txt' using 1:4 with lines title 'quickSort'
gnuplot> set terminal postscript eps color solid
gnuplot> set output 'nVary.ps'
gnuplot> replot
```

Damit erhält man dann folgende Grafik:



Wie man sieht kommt man für `insertionSort()` gut an die quadratische Näherung. Im Vergleich zu `mergeSort()` sind `quickSort()` und `insertionSort()` sind etwas unregelmäßig, da deren Effizienz sehr stark von der Beschaffenheit des Anfangsarrays abhängt. Das Fitting kann man so durchführen:

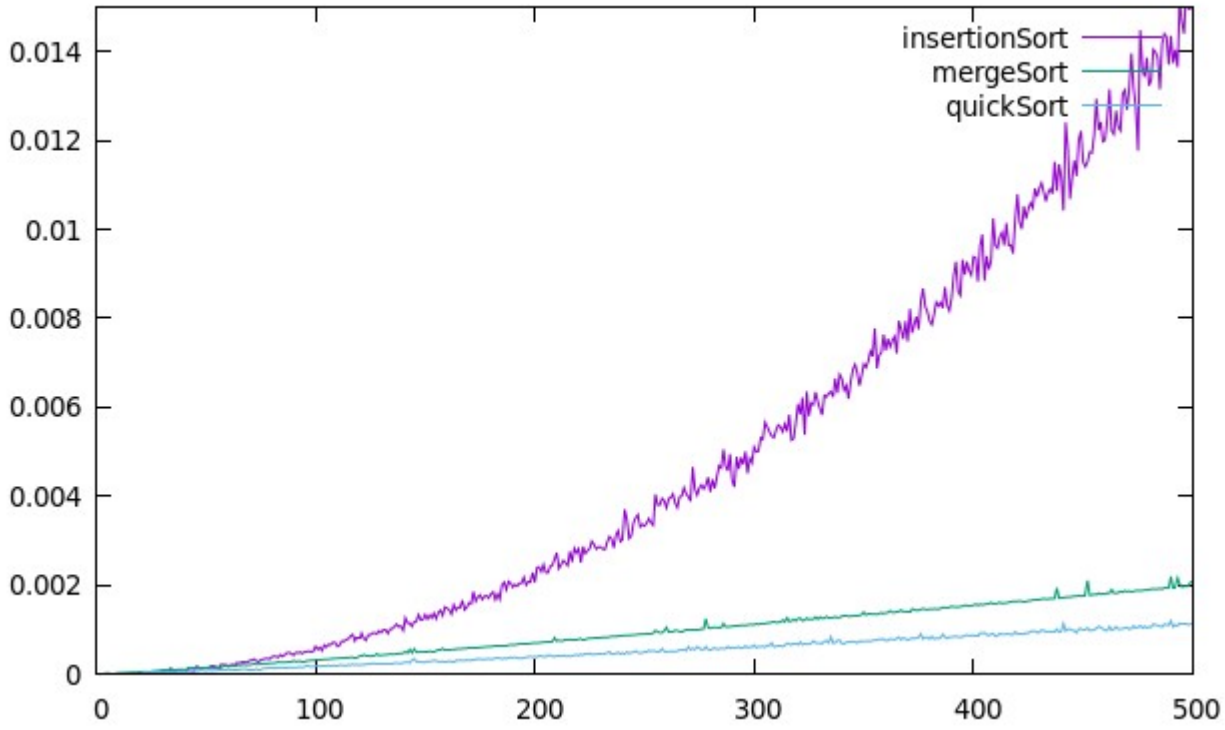
```
gnuplot> set datafile separator ","
gnuplot> f(x) = a*x**2 + b*x + c
gnuplot> g(x) = d*x*log(x)/log(2) + e*x + f
gnuplot> h(x) = g*x*log(x)/log(2) + h*x + i
gnuplot> fit f(x) 'varyN.txt' using 1:2 via a,b,c
gnuplot> fit g(x) 'varyN.txt' using 1:3 via d,e,f
gnuplot> fit h(x) 'varyN.txt' using 1:4 via g,h,i
```

Damit kommt man dann auf folgende Variablen:
a,b,c = 0.251171, 0.318301, 23.7957
d,e,f = 1.00887, -1.32371, 3.11068
g,h,i = 1.33049, -2.56707, 2.73502

b)

```
gnuplot> set datafile separator ","
gnuplot> plot [0:500][0:0.015] 'varyT.txt' using 1:2 with lines title 'insertionSort',\
>'varyT.txt' using 1:3 with lines title 'mergeSort',\
>'varyT.txt' using 1:4 with lines title 'quickSort'
gnuplot> set terminal postscript eps color solid
gnuplot> set output 'tVary.ps'
gnuplot> replot
```

Das ist der Fall: Die funktionelle Form bleibt erhalten, es ändern sich lediglich die Konstanten, wie man an folgendem Plot sieht:



1. Beachte, dass bezüglich der Anzahl der Vergleiche `mergeSort()` schneller als `quickSort()` ist. Beim Zeitvergleich sieht es umgekehrt aus. Das liegt daran, dass das Kopieren von Arrayelementen auch Zeit braucht und diese Zeit bei `quickSort()` kleiner als bei `mergeSort()` ist.