Algorithmen & Datenstrukturen

Praktikum 6: Schnelles Sortieren

Nils Eggebrecht
Florian Heuer

Dienstag, 9. Mai 2017

Inhaltsverzeichnis

1. Einleitung

Die Aufgabe ist es einen schnelleren Sortieralgorithmus zu entwickeln als Quicksort. Dazu wurden Überlegungen und Recherchen angestellt, dessen Ergebnisse in der folgenden Dokumentation festgehalten sind.

2. Verfahren (Insertionquicksort)

Um einen schnelleren Algorithmus als Quicksort zu entwerfen, ist die Idee den bestehenden Algorithmus zu optimieren und die bisherigen Schwächen von Quicksort zu mindern. Die Idee dazu lieferte eine [1]Seite der Internetpräsenz „linux-related.de“. Das Verfahren ist im folgenden beschrieben.

2.1 Beschreibung

Quicksort funktioniert nach dem Prinzip „Teile und Herrsche“ und arbeitet mit einer hohen Anzahl rekursiver Aufrufe. Ziel des neuen Verfahrens ist es die Anzahl dieser Aufrufe zu mindern und so den Algorithmus zu beschleunigen. Dazu wird ab einer bestimmtem Größe (Schwellenwert) der entstehenden Teillisten nicht mehr der Quicksortalgorithmus selbst rekursiv aufgerufen, sondern einer weiterer Algorithmus der für teilsortierte und kleine Listen effizienter arbeitet als Quicksort. Dieser Algorithmus heisst Insertionsort. Der verwendete Schwellenwert ist auf 50 festgelegt.

2.2 Insertionsort

Der Insertionsort-Algorithmus wird auf der folgenden [2]Internetseite beschrieben, sodass an dieser Stelle von einer gesonderten Beschreibung abgesehen wird.

3. Messung & Analyse im Vergleich zu Quicksort

Für die Messung des optimierten Algorithmus werden die Werte (Rekursionen, Vergleiche, Zuweisungen & Laufzeit) für verschiedene Problemgrößen (N=10^K, K=1…6) aufgenommen. Der Algorithmus ist in Java implementiert. Der Code wird in der IDE „Eclipse“ ausgeführt.

3.1 Ergebnisse

3.1.1 Insertionquicksort

N	Rekursionen	Vergleiche	Zuweisungen	Laufzeit
10^1	0	30	83	0 ms
10^2	4	1239	2460	0 ms
10^3	74	16340	31095	0 ms
10^4	753	211025	379611	2 ms
10^5	7578	2555257	4459841	13 ms
10^6	75777	29920260	51073738	109 ms

[1] http://www.linux-related.de/index.html?/coding/alg_quicksort.htm

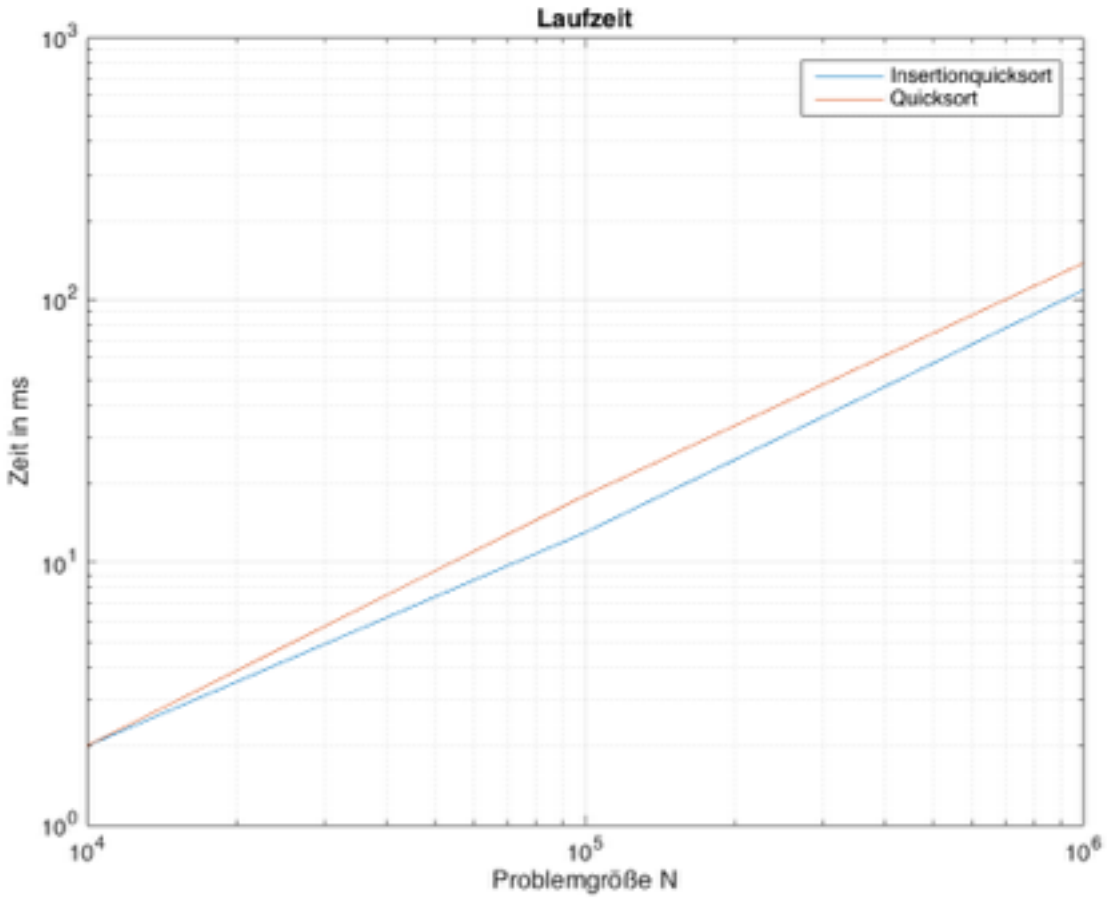
[2] https://de.wikipedia.org/wiki/Insertionsort

3.1.2 Quicksort

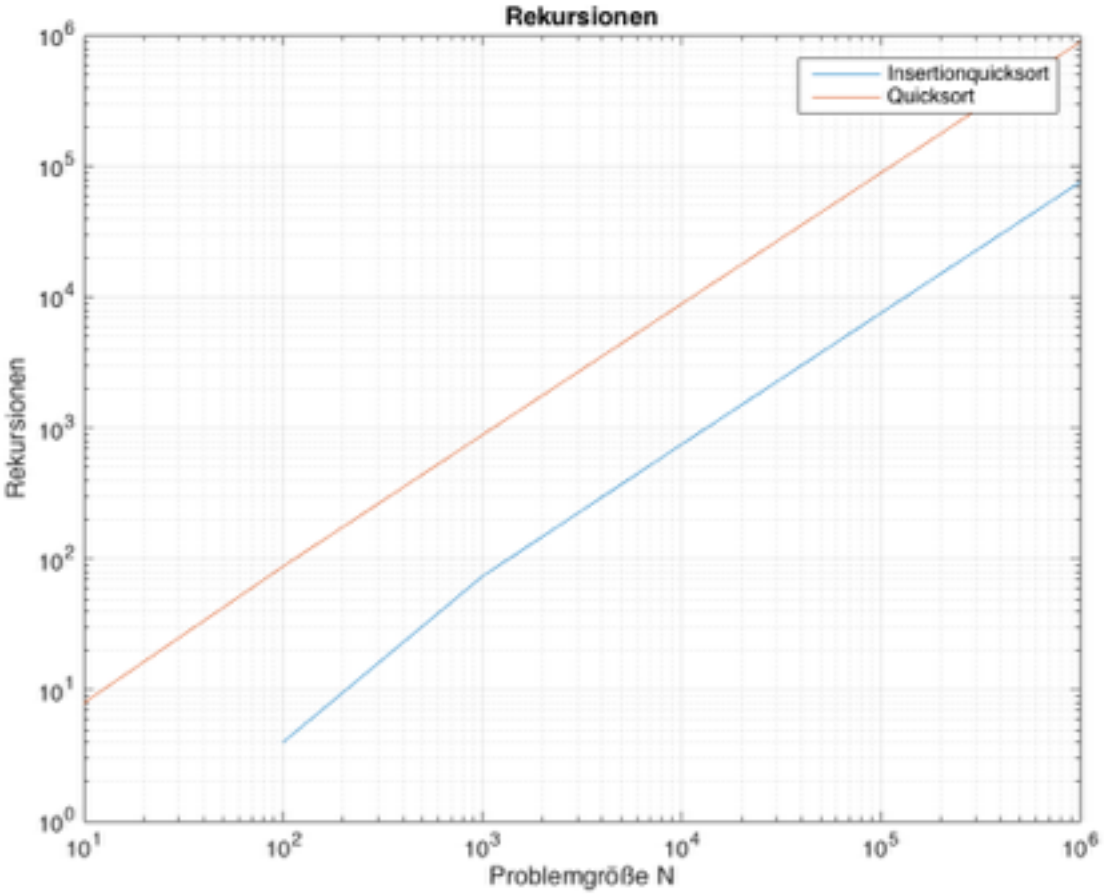
N	Rekursionen	Vergleiche	Zuweisungen	Laufzeit
10^1	6	42	90	0 ms
10^2	88	819	1607	0 ms
10^3	889	13431	23172	0 ms
10^4	8910	177606	303634	2 ms
10^5	89047	2252884	3678801	18 ms
10^6	890075	26867397	43849637	138 ms

3.2 Darstellung

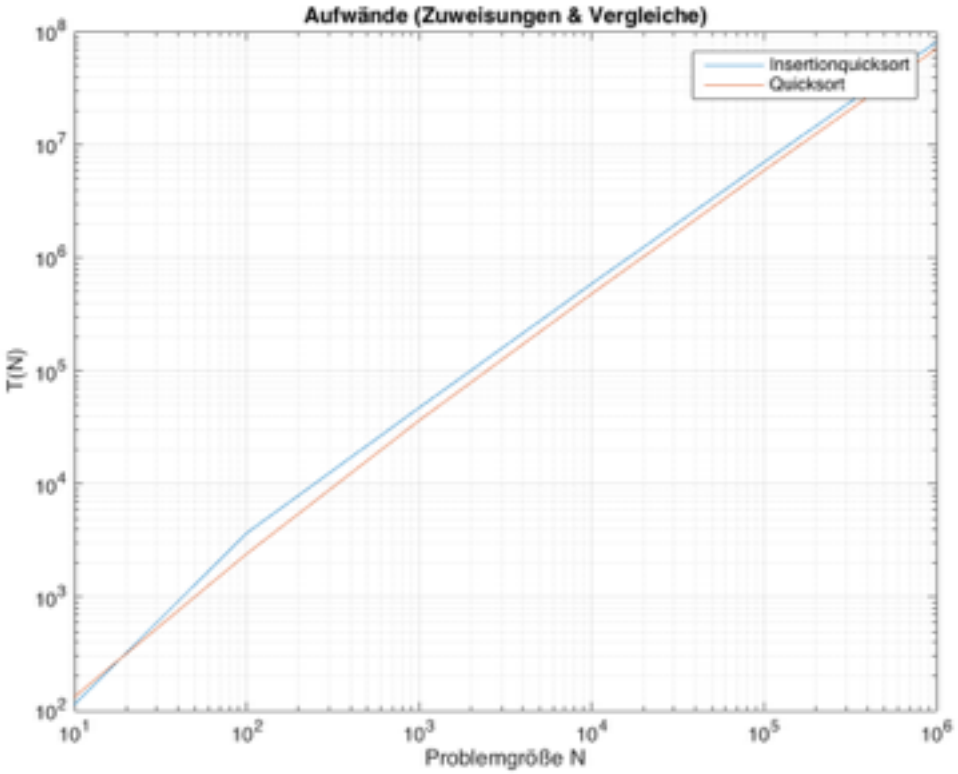
3.2.1 Laufzeit



3.2.2 Rekursive Aufrufe



3.2.3 Zuweisungen & Vergleiche



3.3 Auswertung

Wie man aus den obigen Abbildungen und Tabellen schließen kann, ist die Laufzeit des neuen Algorithmus erst mit steigender Größe N messbar (Bereich ab 10^5 & 10^6). So ist die Laufzeit in diesem Bereich um ca. 24% schneller als bei Quicksort.

Die Rekursionen wurden deutlich verringert, ab dem Bereich ab 10^3 hat der Insertionquiksort fast konstant 1/12 der rekursiven Aufrufe als Quicksort.
Vergleiche und Zuweisungen sind bei Insertionquicksort gestiegen.

Abschliessend zu erwähnen ist, dass bei den Versuchen eine Abhängigkeit vom Schwellenwert festgestellt wurde. Hat man den Wert geändert, hatte dies große Auswirkungen auf die gemesssenen Größen. Durch probieren hat der Wert 50 die besten Ergebnisse geliefert.