EIGENE MESSUNGEN ZU AUFGABE 2

TEAM 10: ANTON, IGOR & MESUT

EINLEITUNG

Die Aufgabe 2 ,,naives vs komplexes Sortieren“ bei der Veranstaltung ,,Algorithmen und Datenstrukturen“ (HAW Hamburg) erforderte, dass wir eine endliche Folge von positiven Zahlen in einer erzeugten Datei ,,zahlen.dat“ *(oder vorgegebene ,,zahlen.dat“)* nach bestimmten Sortieralgorithmen implementieren – basierend auf unsere vorherige Implementation von ADTArray. Zu den Sortieralgorithmen haben wir ebenfalls Laufzeit und Zugriffsmessungen erstellt, sodass wir beurteilen können, welches Sortierverfahren wann bestens geeignet ist.

MESSUNGEN

Um möglichst aussagekräftige Messungen durchzuführen, haben wir bei der Implementation darauf geachtet, dass die erstellte .csv-Datei *(interne Vorgabe)* auch Übersichtlich gestaltet ist, sodass dementsprechend ein Graph bei Excel nach Vorgabe erstellt werden kann. Die unten gezeigten Messungen in der Tabelle sind nur herausgenommene Beispieldaten aus der ,,benchmark.csv“ -Datei.

Bei den Resultaten auf folgendes aufpassen:

- Angenommen: die Laufzeit bei Quicksort ,,**200 Zahlen,63,16,79**“ *(die Messung befindet sich bei der Tabelle an der vierten Stelle)*. Die 63 ist dabei die Zeit, die Quicksort selbst braucht, die 16 für InsertionSort und die 79 ist die gesamte Zeit für den Aufwand.

Versuchsaufbau	Algorithmus	Resultate	Kommentar
Laufzeit	Insertion Sort (random number)	12 Zahlen,16 100 Zahlen,235 200 Zahlen,625 500 Zahlen,3750 1000 Zahlen,12000	Insertion Sort braucht für zufällige Zahlen bedauerlicherweise sehr viel.
Laufzeit	Insertion Sort (left sorted)	12 Zahlen,0 100 Zahlen,15 200 Zahlen,31 500 Zahlen,32 1000 Zahlen,31	Für linkssortierte Zahlen braucht Insertion Sort sich nicht mal anstrengen.
Laufzeit	Insertion Sort (right sorted)	12 Zahlen,16 100 Zahlen,219 200 Zahlen,1032 500 Zahlen,5687 1000 Zahlen,23782	Bei rechtssortierten Zahlen haben wir noch mal bei größeren Zahlen einen aufwändigeren Insertion Sort.
Laufzeit	Quicksort (pivot links – random numbers)	12 Zahlen,0,0,0 100 Zahlen,32,0,32 200 Zahlen,63,16,79 500 Zahlen,157,0,157 1000 Zahlen,234,16,250 2000 Zahlen,484,15,499 5000 Zahlen,1219,0,1219	Quicksort braucht bei einem pivot links mit zufälligen Zahlen bei beispielsweise 200 Zahlen nur 63 ms.
Zugriffe	Insertion Sort *(random number)*	12 Zahlen,88 100 Zahlen,5394 200 Zahlen,20496 500 Zahlen,123288 1000 Zahlen,492820	Zugriffe bei Insertion Sort sind enorm viel, gerade bei zufälligen Zahlen ist das allerdings vorhersehbar.

Zugriffe	Quick Sort (pivot links)	12 Zahlen,1,83,84 100 Zahlen,2806,2757,5563 200 Zahlen,5934,6231,12165 500 Zahlen,38609,39234,77843 1000 Zahlen,128719,130222,258941 2000 Zahlen,895928,897403,1793331 5000 Zahlen,3346878,3354079,6700957	Quick Sort mit einem pivot links braucht großen Aufwand (bei 200 Zahlen 5934ms) ohne Insertion Sort.
Zugriffe	Quick Sort (pivot random)	12 Zahlen,1,57,58 100 Zahlen,2428,2435,4863 200 Zahlen,7562,7797,15359 500 Zahlen,33430,34137,67567 1000 Zahlen,237929,238560,476489 2000 Zahlen,845646,847339,1692985 5000 Zahlen,5620078,5623611,11243689	Quick Sort mit einem zufälligen Pivot braucht allerdings auch einen großen Aufwand.