Algorithmen und Datenstrukturen

Sortieren 2

Verwenden Sie als Basis für die nachfolgenden Aufgaben Ihre Lösung aus dem ersten Sortieren-Praktikum.

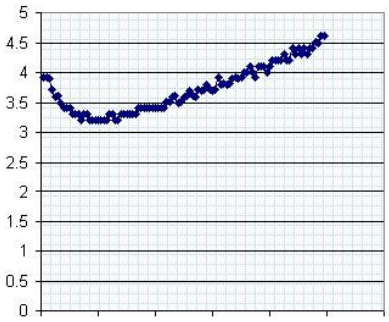
Aufgabe 1: Beschleunigung [3 Punkte]

Quicksort zählt zu den schnellsten bekannten Sortieralgorithmen. Dennoch lässt er sich noch beschleunigen, indem für kurze, zu sortierende Intervalle in der Rekursion ein einfaches Sortierverfahren verwendet wird, wie z.B. Insertion-Sort. Für diesen Zweck wird eine Schwelle (threshold) definiert, ab dem auf das einfache Verfahren umgeschaltet wird. Wir bezeichnen diesen neuen Sortieralgorithmus - bescheiden wie wir sind - als QuickerSort.

Schreiben Sie eine Klasse `QuickerSortServer`, die den CommandInterpreter implementiert und mittels welcher zufällige Werte mit dem QuickerSort-Verfahren sortiert werden können. Nehmen Sie als Schwelle für die Umschaltung 50 zu sortierende Werte.

Aufgabe 2: Bestimmung der optimalen Schwelle [2 Punkte]

Verwenden Sie Ihren QuickerSort-Algorithmus und bestimmen Sie empirisch (d.h. durch Messen) einen guten Schwellwert. Sie können die Werte in Excel auftragen und so das Minimum optisch bestimmen. Bei welchem Schwellwert hat Ihr Algorithmus das Optimum?



Figur 1: Beispiel - Laufzeit von QuickerSort in Abhängigkeit vom Schwellwert

Hinweis:

- Er sollte etwa im Bereich zwischen 1 und 100 liegen
- Vergleichen Sie die Zeiten Ihres Algorithmus mit Arrays.sort

Aufgabe 3: Beschleunigung durch Parallelisierung [5 Punkte]

Auf einem Rechner mit einem Mehrkernprozessor kann Quicksort noch erheblich beschleunigt werden, indem alle Rechenkerne ausgelastet werden. Implementieren Sie basierend auf Ihrem QuickerSort-Algorithmus einen solchen parallelen Sortieralgorithmus.

Hinweis

- Sie können Threadpools oder das Fork/Join Framework verwenden.