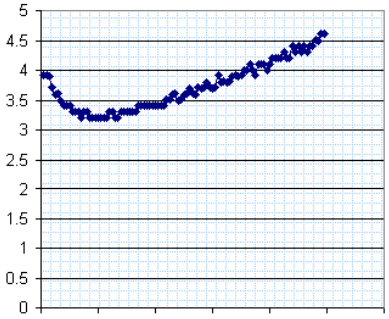
Algorithmen und Datenstrukturen

Sortieren 2

Aufgabe 1 Beschleunigung [2 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 (insertion_threshold) definiert, ab dem auf das einfache Verfahren umgeschaltet wird (lediglich für das Intervall!). Wir bezeichnen diesen neuen Sortieralgorithmus – bescheiden wie wir sind - als Quicker-Sort.



Figur 1 Laufzeit von Quicker-Sort in Abhängigkeit vom Schwellwert

Schreiben Sie eine Klasse `QuickerSortServer`, die den CommandExecutor implementiert und mittels der zufällige int-Werte (vergleiche letztes Praktikum) mit dem (standard) QuickSort-Verfahren sortiert werden können. Nehmen Sie als Schwelle für die Umschaltung 50 und messen Sie die Zeit.

Bestimmung der optimalen Schwelle [3 Punkte]

Implementieren Sie Quicker-Sort und bestimmen Sie empirisch (d.h. durch Messen) einen guten Schwellwert. Sie können die Werte in Excel auftragen und so das Minimum optisch bestimmen.

Hinweis:

- Er liegt im Bereich zwischen 1 und 100
- Vergleichen Sie die Zeiten mit dem Stream Sort; sie sollten diesen mit einer guten Wahl des Threasholds schlagen können.

Abgabe

Praktikum: ADS12.1

Filename: QuickerSortServer.java

Aufgabe 2 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 Quicksort einen solchen parallelen Sortieralgorithmus.

Hinweis

- Das Fork/Join Framework verwenden.

Abgabe

Praktikum: ADS12.2

Filename: QuickerSortServer.java