Algorithmen und Datenstrukturen

Sommersemester 2022

Prof. Dr. Philipp Jenke

HAW HAMBURG | Department Informatik

Aufgabe 2

Dieses Aufgabenblatt baut auf den Vorlesungen auf:

- Sortieren,
- Rekursives Sortieren.

Deliverables:

- PDF mit Lösungen zur Theorie-Aufgabe
- Code für Praxisaufgabe (Code-Konventionen, Tests)

1 Theorie

1.1 Vollständige Induktion

Beweisen Sie mit vollständiger Induktion, dass $5^{2n} + 24n - 1$ teilbar durch 24 für alle $n \in \mathbb{N}_0$ ist.

2 Praxis

2.1 Mergesort

Implementieren Sie den Mergesort-Algorithmus. Der Algorithmus sortiert eine `java.util.List` von beliebigen Elementen. Die Klasse muss daher generisch sein. Zum Sortieren erhält sie einen passenden `Comparator`.

```
var data = Arrays.asList(23, 42, 11, 1, 12);
var mergeSort = new MergeSort<Integer>();
mergeSort.setup(data, (i1, i2) -> i1 - i2);
```

Es werden allerdings nicht die Element-Positionen in der Eingabe-Liste verändert. Stattdessen wird die Sortierung durch ein Permutations-Array τ angegeben. Das Array hat so viele `int`-Elemente wie die Eingabedaten-Liste Elemente hat. Jedes τ-Element gibt den Index des entsprechenden Eingabe-Elements nach der Sortierung an. Auch intern verwenden Sie nur Permutations-Arrays und keine weiteren Listen der Eingabe-Elemente. Für ein Permutationsarray `tau` greifen Sie also auf das i'te Element so zu: `data.get(tau[i])`.

```
int[] tau = mergeSort.sort();
```

Für das obige Beispiel sollte `tau` dann so aussehen: $[3, 2, 4, 0, 1]$. Das bedeutet, dass das erste Element nach der Sortierung das mit dem Index 3 ist (also die 1), das zweite Element ist das mit dem Index 2 (also die 11) und das letzte in den sortierten Daten ist das mit dem Index 1 (also die 42).

2.2 Ergänzung: In-Situ-Verfahren

Modifizieren Sie Ihre Implementierung so, dass beim rekursiven Aufruf immer ein in-situ-Verfahren verwendet wird, wenn die Anzahl der zu sortierenden Elemente einen Schwellwert `d_insitu` unterschreitet. Arbeiten Sie mit `d_insitu = 100` und setzen Sie als in-situ-Verfahren das Insertion-Sort um. Verwenden Sie nur ein einziges Daten-Array für die Insertion-Sort-Implementierung, sodass Sie beim Ausführen von Insertion-Sort kein neues Daten-Array erstellen müssen.

Vergleichen Sie die Laufzeit mit und ohne diese Anpassung für große Datensätze (mit beispielsweise 100.000 und mehr Elementen).

Hinweis: Tatsächlich ist der Performance-Gewinn durch dieser Änderung bei dem Mergesort-Verfahren nicht sehr deutlich. Die Anpassung ist insbesondere beim Quicksort sinnvoll.