Algorithmen und Datenstrukturen

Dr. M. Lüthi, Dr. G. Röger
Frühjahrssemester 2018

Universität Basel
Fachbereich Informatik

Übungsblatt 11

Abgabe: 25. Mai 2018

In diesem Übungsblatt erarbeiten und implementieren sie das LSD (Least Significant Digit) Sortierverfahren und experimentieren mit diesem. Sie finden einen Scan des relevanten Buchkapitels aus dem Buch *Algorithmen*, von Robert Sedgewick und Kevin Wayne, auf dem Adam-workspace (`string-sortierung.pdf` `https://adam.unibas.ch/goto_adam_file_677775_download.html`).

Als Grundlage für die praktischen Übungen verwenden Sie das Projekt im Zip-File "blatt11.zip", welches Sie auf dem Adam-Workspace finden. Die Lösungen zu den Theoriefragen geben Sie als Dokument im PDF Format, mit dem Namen `loesung-11.pdf` ab. Bitte fügen sie diese PDF Datei zur Projektstruktur hinzu. Laden Sie Ihre Lösungen als zip Datei auf Courses hoch.
Das kompilieren und ausführen der Testfälle erfolgt wie in den vorhergehenden Übungen durch `gradlew build` respektive `gradlew test`.
Falls sie keine IDE verwenden, können Sie ihre Programme von der Kommandozeile mittels

```
.\gradlew sortExperiments
```

starten.

Aufgabe 11.1 (Erarbeiten des LSD-Sortierverfahrens, 3 Punkte)

Lesen Sie die Abschnitte 5.1.1 und 5.1.2 im Kapitel *LSD-Sortierverfahren* und beantworten Sie dazu folgende Fragen:

- Erklären sie wozu die Arrays `count` und `aux` im Algorithmus für schlüsselindiziertes Zählen und dem LSD-Sortierverfahren benutzt werden.
- Weshalb ist die Stabilität vom schlüsselindiziertem Zählen wichtig um zu garantieren, dass das LSD-Sortierverfahren korrekt funktioniert?
- Was ist die Laufzeitkomplexität vom LSD-Sortierverfahren und wie stark beeinflusst die Grösse vom Alphabet (im Buch als R bezeichnet) die Laufzeit?

Aufgabe 11.2 (Ablaufprotokol für LSD-Sortierverfahren, 1 Punkt)

Geben Sie die Reihenfolge der folgenden Schlüssel nach jedem Zwischenschritt (d.h. nach dem Sortieren nach dem zweiten Buchstabe (d=1) und nach dem Sortieren des ersten Buchstabens (d=0)) an.
no is th ti fo al go pe to co to th ai of th pa

Aufgabe 11.3 (LSD-Sortierverfahren für Strings variabler Länge, 2 Punkte)

Erweitern sie die Implementation des LSD-Sortierverfahrens (`exercise11.LSDSort.sort`), so dass sich damit Strings variabler Länge sortieren lassen. Testen Sie ihre Implementation anhand der entsprechenden Unit tests.

Aufgabe 11.4 (3-Weg Quicksort für Strings, 1 Punkt)

Implementieren Sie das 3-Weg Quicksort für Strings indem sie die Methode sort in `exercise11.StringQuicksort.java` implementieren. (Sie können den Python Code im Jupiter

Notebook `Strings.ipynb` als Vorlage nehmen). Testen Sie ihre Implementation anhand der entsprechenden Unit tests.

Aufgabe 11.5 (Vergleich Quicksort vs. LSD Sortierung, 3 Punke)
Ergänzen sie die Klasse `exercise11.SortExperiments` um

- einen Generator für zufällige Wörter fester Länge: Implementieren Sie dazu die statische Methode `randomFixedLengthWords` , die die int-Werte N und W als Argumente übernimmt und ein Array von N Stringwerten zurückliefert , die jeweils Strings von W Zeichen aus dem Alphabet (a-zA-Z) sind.
- einen Generator für zufällige Wörter fester Länge mit gemeinsamen Präfix: Implementieren Sie dazu die statische Methode `randomItemsWithCommonPrefix` , die die int Werte N, P, W als Argument übernimmt und ein Array von N Stringwerte der grösse W zurückliefert, die jeweils ein gemeinsamens Präfix der Länge P haben.
- Vergleichen Sie die Laufzeit vom LSD-Sortierverfahren mit dem 3-Wege-Quicksort für Strings unter Verwendung der zwei verschiedenen Schlüsselgeneratoren. Nutzen sie dazu die Klasse `exercise11.utils.Stopwatch`. Wie ist die Laufzeit der zwei Verfahren bei zufälligen Strings? Wie verändert sich die Laufzeit, wenn das gemeinsame Präfix immer länger wird? Können Sie das Verhalten erklären?

Die Übungsblätter dürfen in Gruppen von zwei Studierenden bearbeitet werden. Bitte schreiben Sie beide Namen auf Ihre Lösung.