Algorithmen und Datenstrukturen
WS 2025
von Luxburg/Bordt

Übungsblatt 9

Abgabe am 12.1.2026

Aufgabe 1: Stabilität (2 Punkte)

Ein Sortieralgorithmus heißt **stabil**, wenn er die relative Reihenfolge von Elementen mit gleichem Schlüsselwert beibehält. Das bedeutet: Wenn zwei Elemente a und b denselben Sortierschlüssel haben und a in der Eingabe vor b steht, dann steht a auch in der sortierten Ausgabe vor b.

Beispiel: Gegeben sei eine Liste von Studierenden, die bereits nach Namen sortiert ist. Sortieren wir diese Liste nun stabil nach Matrikelnummer, so bleiben Studierende mit gleicher Matrikelnummer (falls vorhanden) weiterhin alphabetisch nach Namen geordnet. Stabilität ist besonders wichtig, wenn nach mehreren Kriterien sortiert werden soll.

a) Entwickeln Sie eine Strategie, mit der Sie mit $O(n)$ zusätzlichen Speicher *jedes* Sortierverfahren stabilisieren können.

Aufgabe 2: Quicksort (2 Punkte)

Es werde Quicksort auf einem Array aus n Zahlen ausgeführt.

a) Betrachten Sie eine Quicksort-Variante, welche als Pivotelement dasjenige Element der Eingabesequenz A verwendet, welches am nächsten am arithmetischen Mittelwert der Zahlen aus A liegt. Kann auf diese Weise die asymptotische worst-case Laufzeit von $\mathcal{O}(n^2)$ unterschritten werden? Beweisen oder widerlegen Sie.

Aufgabe 3: Doppelte Pointer (2 Punkte)

Gegeben sei eine doppelt verkettete Liste mit N Pointern auf verschiedene Objekte. Entwickeln Sie einen effizienten Algorithmus, der Duplikate identifiziert und aus der Liste entfert.

Aufgabe 4: Kleinster Abstand (2 Punkte)

Gegeben sei ein Array mit N verschiedenen Zahlen. Entwickeln Sie einen effizienten Algorithmus, der die beiden Zahlen mit dem kleinsten Abstand findet. Geben sie eine untere- und eine obere Schranke and die Laufzeit an.

Aufgabe 5: Fast Sortiert (4 Punkte)

Gegeben sei ein Array mit N Elementen, von denen jedes Element maximal k Positionen von der Stelle entfernt ist, an der es sich im sortierten Array befinden würde. Entwickels Sie einen Algorithmus mit Laufzeit $O(N \log k)$, der das Array sortiert.

Aufgabe 6: Dreiersumme (4 Punkte)

Gegeben sei ein Array mit N verschiendenen ganzen Zahlen (sowohl positive als auch negative). Wir suchen die drei Zahlen in dem Array, deren Summe so nah wie möglich an 0 ist. Entwickeln Sie einen Algorithmus mit Laufzeit $O(N^2)$ der dieses drei Zahlen findet.

Aufgabe 7: Laufzeit auf meinem Rechner (4+2+1+2+1+1+2+4+2+1 Punkte)

In dieser Aufgabe wollen wir die Laufzeit von zwei verschiedenen Sortieralgorithmen vergleichen: Selection Sort und Quick Sort. Wir interessieren uns ausnahmsweise mal nicht für die asymptotische Laufzeit, sondern für die Laufzeit auf unserem eigenen Rechnern. Dabei interessieren wir uns auch für den Unterschied, den die Wahl der Programmiersprache macht. Los geht's!

1. Implementieren Sie Selection Sort und Quicksort in Python. Dazu können Sie das auf der Website bereitgestellte Notebook `sorting.ipynb` verwenden.

2. Bei Quicksort gibt es verschiedene Möglichkeiten das Pivot-Element zu Wählen. Erläutern Sie kurz die Rolle des Pivot-Elements. Wie wird das Pivot-Element in Ihrer Implementierung gewählt?

3. Gibt es für Ihre Implementierungen von Selection Sort und Quicksort worst-case Eingaben? Falls ja, welche?

4. Testen Sie die Korrektheit Ihrer Implementierungen anhand von verschiedenen Testfällen. Erläutern Sie kurz, wie Sie diese Testfälle gewählt haben.

5. Messen Sie die Laufzeit der beiden Sortieralgorithmen auf Ihrem Rechner. Dazu können Sie den in `sorting.ipynb` bereitgestellten Code verwenden.

6. Erstellen Sie einen geeignent Plot, der die Laufzeit der beiden Algorithmen in Abhängigkeit von der Eingabelänge darstellt. Ab welcher Eingabelänge macht sich der Unterschied in der asymptotischen Laufzeit in der Praxis bemerkbar?

7. Erstellen Sie für die beiden Sortieralgorithmen jeweils einen Plot, in dem gemessene Laufzeit durch die entsprechende asymptotische Laufzeitentwicklung approximiert wird. Dazu müssen Sie die Konstante C ermitteln, die wir in der $O(\cdot)$-Notation unter den Tisch kehren. Welche Konstanten ermitteln Sie für die beiden Algorithmen?

8. Python ist eine sehr gut lesbare, aber auch eine sehr ineffiziente Programmiersprache. Daher würde man einen Sortieralgorithmus in der Praxis niemals in nativem Python implementieren. Implementieren Sie Quicksort nun in einer effizienten Programmiersprache Ihrer Wahl (also z.B. Java, C/C++, Julia). Vergleichen Sie die Laufzeit mit Ihrer Python Implementierung, in dem Sie einen geeigneten Plot erstellen.

9. Vergleichen Sie die Laufzeit ihrer effizienten Implementierungen mit der Laufzeit der nativen `sort()` Funktion für Listen in Python. Dazu erstellen Sie natürlich wieder einen geeigneten Plot. Interpretieren Sie das Ergebnis.

10. Wie sehen sie das Verhältnis zwischen einer effizienten Implementierung und der theoretischen Analyse der asymptotischen Laufzeit? Was ist die Größenordnung des konstanten Faktors, der durch die Wahl der Programmiersprache bestimmt wird?