Marburg University

Department of Computer Science
Prof. Dr. Sebastian Wild

Date: 2026-06-13
Version: 2026-06-14 13:18

Exercise Sheet 9 zur Vorlesung Algorithmen und Datenstrukturen (Sommer 2026)

Abgabe: *Bis 2026-06-20 18:00, on ILIAS.*

1. Aufgabe

20 Punkte

Beschreiben Sie eine Implementierung von Mergesort für verkettete Listen. Ihre Beschreibung sollte eine Erläuterung der Grundidee des Algorithmus, Pseudocode für die Implementierung sowie Argumente zur Korrektheit und Effizienz enthalten.

Für volle Punkte muss Ihre Implementierung mit konstant viel Zusatzspeicher auskommen (zusätzlich zur Eingabe, die Ihnen als einfach verkettete Liste gegegeben ist) und darf auch nur konstant viele neue Objekte allozieren.

Tipp: Verwenden Sie Bottom-Up-Mergesort.

2. Aufgabe

30 + 30 Punkte

In der folgenden Aufgabe müssen Sie eine Datenstruktur rein *mithilfe von Stacks (Stapeln)* implementieren. Das bedeutet, Ihre Datenstruktur darf nur $O(1)$ zusätzliche Objekte speichern (z. B. Integer etc.), der gesamte weitere Speicherbedarf muss innerhalb einer gegebenen Anzahl von Stacks liegen, auf die Sie über das Interface Zugriff haben. Diese Stacks verfügen über die üblichen Operationen, `push`, `pop` und `top`.

a) Implementieren Sie eine Queue (Warteschlange) mithilfe von 2 Stacks. Die Queue muss eine `enqueue`-Operation und eine `dequeue`-Operation unterstützen; dabei für die `enqueue`-Operation ein Element am Ende der Warteschlange ein, die `dequeue`-Operation entfernt ein Element vom Anfang der Warteschlange. Diese Operationen dürfen in beliebiger Reihenfolge aufgerufen werden. Jede Operation muss *amortisiert* $O(1)$ Kosten haben.

b) Implementieren Sie eine Deque (double-ended queue / beidseitige Warteschlange) mithilfe von 3 Stacks. Wie in der Vorlesung, repräsentiert eine Deque eine Liste von Werten und unterstützt 4 Operationen: Einfügen und Entfernen eines Elements am Anfang bzw. Ende der Deque. Diese Operationen dürfen in beliebiger Reihenfolge aufgerufen werden. Jede Operation muss *amortisiert* $O(1)$ Kosten haben.

3. Aufgabe

10 + 10 + 20 + 20 Punkte

Im magischen Königreich der ewigen Komplexität gibt es eine riesige Steinstatue der aktuellen Jahreszahl in Dezimaldarstellung. Jedes Jahr sind Sie damit beauftragt, diese Statue zu aktualisieren. Ein Zauberer des Königreichs ist in der Lage, die Ersatzziffern augenblicklich zu meißeln – aber er ist zu faul, sie korrekt aufzustellen, sodass diese Aufgabe Ihnen obliegt.

Angenommen, Sie haben damit im Jahr 0 begonnen (als noch keine Statue vorhanden war) und setzen dies bis (einschließlich) zum Jahr n fort. Wir interessieren uns für die Gesamtanzahl der Zifferänderungen, die Sie in dieser Zeit vornehmen mussten.

Beispiel: Für $n = 12$ geschieht Folgendes. Sie beginnen im Jahr 0, als noch keine Statue vorhanden ist. In jedem der Jahre 1, 2, 3, 4, 5, 6, 7, 8, 9 findet genau 1 Zifferänderung statt, was Kosten von 9 Zifferänderungen ergibt. Im Jahr 10 finden 2 Zifferänderungen statt, dann in den Jahren 11 und 12 jeweils eine Zifferänderung. Insgesamt gibt es $9 + 2 + 1 + 1 = 13$ Zifferänderungen.

Hinweis: Die folgende Tatsache könnte nützlich sein: Für Zahlen $a_1, \ldots, a_k$, die jeweils im Bereich $1, \ldots, v$ liegen, ist $a_1 + \cdots + a_k$ gleich der Summe $s_1 + \cdots + s_v$, wobei s_i die Anzahl der Elemente a_j mit $a_j \geq i$ ist.

a) Wie lautet die Anzahl an Zifferänderungen für $n = 100$?

b) Wie lautet diese Zahl für $n = 123134$?

c) Geben Sie eine asymptotisch scharfe Schranke für diese Zahl in Abhängigkeit von n an.

d) Wie lautet die Anzahl an Zifferänderungen für

$$n = 15290963789043709 40?$$

(Falls Sie ein Programm zur Berechnung verwenden, fügen Sie es bitte zusammen mit einer Erklärung seiner Funktionsweise zu Ihrer Abgabe hinzu.)

4. Aufgabe

30 Punkte

Erstellen Sie eine Datenstruktur, die eine Folge von ganzen Zahlen repräsentiert und die folgenden Operationen jeweils in amortisiert $O(1)$ Zeit unterstützt.

- Füge einen Wert v am Anfang der Folge ein.
- Füge einen Wert v am Ende der Folge ein.
- Entferne das erste Element der Folge und gib es zurück.
- Entferne das letzte Element der Folge und gib es zurück.

- Kombiniere zwei Folgen auf folgende Weise: Wenn wir eine Folge $x_1, \ldots, x_n$ mit einer Folge $y_1, \ldots, y_m$ kombinieren, wobei $n \leq m$, so soll die Folge

$$x_1 + y_1, \ldots, x_n + y_n, y_{n+1}, \ldots, y_m$$

  erzeugt werden. Falls $n > m$, wird symmetrisch vorgegangen. Diese Operation *zerstört* die beiden ursprünglichen Folgen.