Grundlagen: Algorithmen und Datenstrukturen

Woche 4

Tobias Eppacher

School of Computation, Information and Technology

19. Mai 2025

Table of contents

Aufgabe 4.1 - Sortierverfahren

Zahlenfolge: $[523, 126, 67, 1, 500, 34, 21, 229, 9, 123, 13]$

(a) Mergesort:
Ein Zwischenschritt je Aufteilen und Verschmelzen (hier ca. 8 Schritte).
Teilen ungerader Länge $\rightarrow$ Linke Seite eins mehr.

(b) Quicksort:
Wie in der Vorlesung $\rightarrow$ letztes Element $=$ Pivot.
Machen Sie kenntlich welches Pivot in jedem Schritt verwendet wird und geben Sie das Array nach jedem Umsortieren an.
Sortierschritte auf zwei nicht überlagernden Abschnitten der Zahlenfolge können im selben Schritt dargestellt werden.
Bei Teilsequenzen der Länge 2 kann direkt ohne Rekursion getauscht werden.

(c) Welches Sortierverfahren ist im Worst- bzw. Averagecase schneller?

Aufgabe 4.1 - Sortierverfahren (a)

523	126	67	1	500	34	21	229	9	123	13

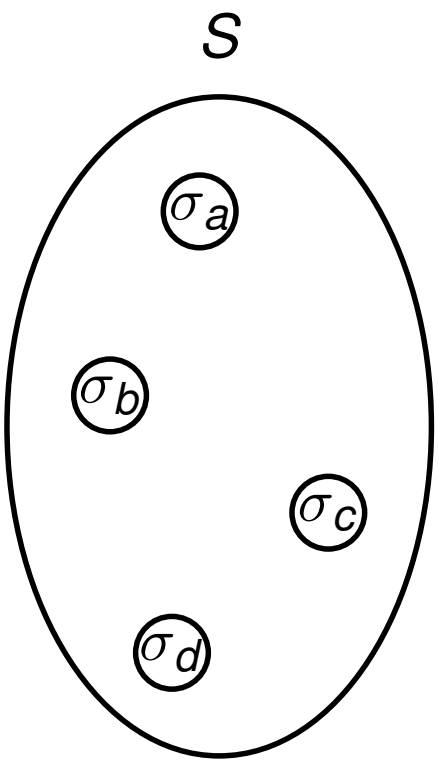
Aufgabe 4.1 - Sortierverfahren (b)

523	126	67	1	500	34	21	229	9	123	13

Aufgabe 4.1 - Sortierverfahren (c)

Welches Sortierverfahren ist im Worst- bzw. Averagecase schneller?

Aufgabe 4.2 - Bankkonto Tutorial



Menge von Operationen: S

(Obere Schranke für) Laufzeit von Operation $\sigma \in S$: $T(\sigma)$

(Obere Schranke für) Laufzeit einer Folge von m Operationen:
$T(\sigma_1, \sigma_2, \ldots, \sigma_m) := \sum_{i=1}^{m} T(\sigma_i)$

Ziel der Amortisierten Analyse

Möglichst genaue obere Schranke für $T(\sigma_1, \sigma_2, \ldots, \sigma_m)$ finden

Aufgabe 4.2 - Bankkonto Tutorial

Bankkonto Methode

Idee: Verteile Laufzeit von langsamen auf schnellere Operationen

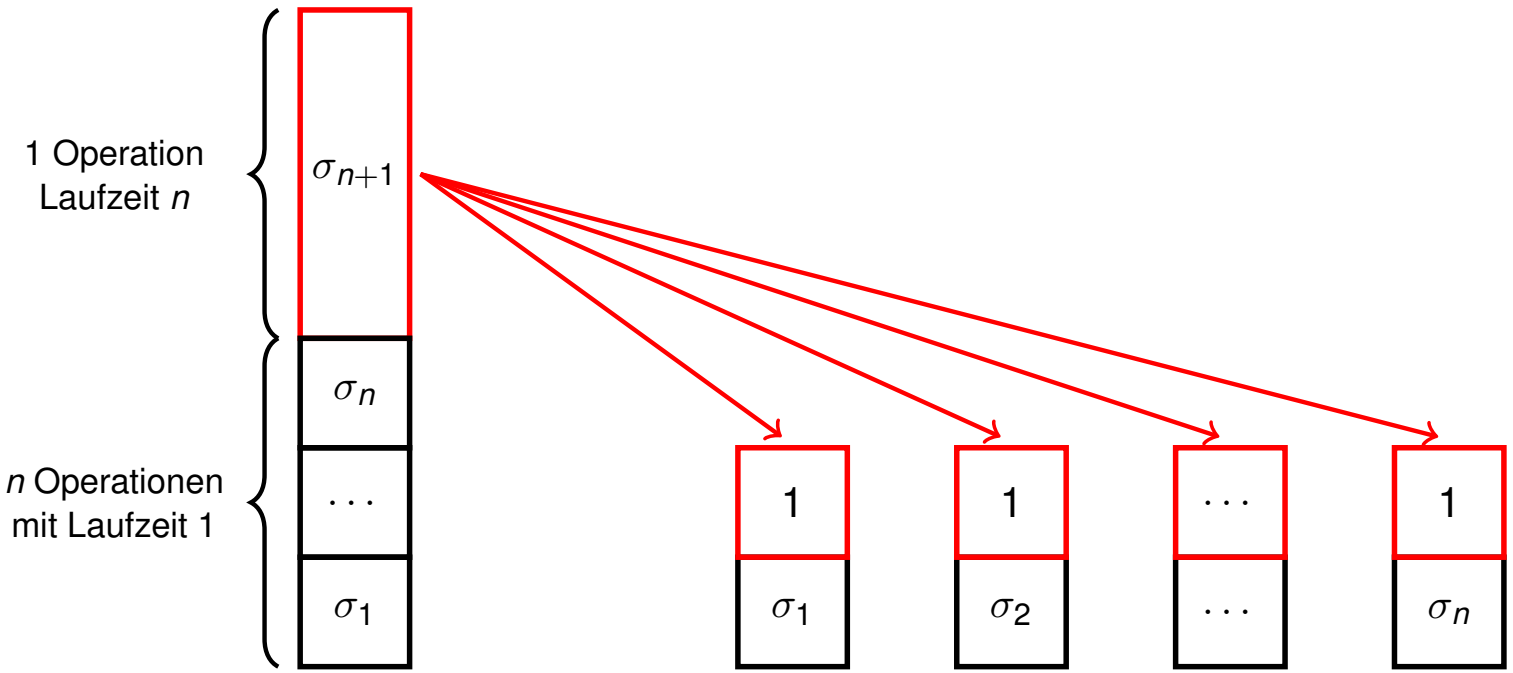
$\Rightarrow$ Bessere Abschätzung der Laufzeit

Aufgabe 4.2 - Bankkonto Tutorial

Bankkonto Methode

Idee: Verteile Laufzeit von langsamen auf schnellere Operationen

$\Rightarrow$ Bessere Abschätzung der Laufzeit



Aufgabe 4.2 - Bankkonto Tutorial

- Schnelle Operationen können Laufzeit übernehmen
  $\Rightarrow$ Zahlen Tokens auf Konto ein (”nicht verwendete Zeit”)
- Mehrere schnelle Operationen sparen Tokens an
- Folgende langsame Operation kann Tokens abheben
  $\Rightarrow$ Gesparte Zeit wird aufgebraucht

Wir definieren die Funktion $\Delta : S \to \mathbb{R}$ die bestimmt, wieviele Tokens eine Operation einzahlt oder abhebt.
Diese muss zwei Eigenschaften erfüllen:

1. Für legale Operationsfolgen: $\sum_{i=1}^{m} \Delta(\sigma_i) \geq 0$
   (Konto darf nie negativ werden)
2. Δ ist möglichst gut gewählt. *(Dazu gleich mehr)*

Aufgabe 4.2 - Bankkonto Tutorial

Amortisierte Laufzeit

Für eine Operation:

$$A(\sigma) := T(\sigma) + \Delta(\sigma)$$

Für eine Folge von Operationen:

$$A(\sigma_1, \ldots \sigma_m) = \sum_{i=1}^{m} A(\sigma_i) = \sum_{i=1}^{m} T(\sigma) + \Delta(\sigma)$$

$$= T(\sigma_1, \ldots \sigma_m) + \underbrace{\sum_{i=1}^{m} \Delta(\sigma_i)}_{\geq 0} \geq T(\sigma_1, \ldots \sigma_m)$$

$\Rightarrow$ Amortisierte Laufzeit ist eine *obere Schranke* für die tatsächliche Laufzeit

Aufgabe 4.2 - Bankkonto Tutorial

Wie wählt man Δ?

Häufig sinnvoll:
Amortisierte Laufzeit der schlechtesten Operation möglichst niedrig halten (i.e. minimiere $\max_{\sigma \in S}(A(\sigma))$)

Beispiel:

- Operationsfolge: $S = \{\sigma_1, \sigma_2, \ldots, \sigma_m\}$
- Schema mit $\max_{\sigma \in S}(A(\sigma))$ möglichst klein, gefunden

$\Rightarrow$ Asymptotische Worst-Case Laufzeit ist in

$$\mathcal{O}(m \cdot \max_{\sigma \in S}(A(\sigma)))$$

Aufgabe 4.3 - Amortisation

Der Zähler

Wir starten einen Zähler mit 0 und haben eine Folge von m Inkrement-Operationen. (D.h. Operation σ_k: Zähler $k-1 \to k$)

Zähler arbeitet in Dezimalschreibweise (Ziffern $0-9$ pro Stelle)

Änderung einer Stelle hat Laufzeit 1

Beispiele:

σ_{134} : Zähler 133 $\to$ 134 $\implies$ Laufzeit 1
σ_{240} : Zähler 239 $\to$ 240 $\implies$ Laufzeit 2
σ_{9999} : Zähler 09999 $\to$ 10000 $\implies$ Laufzeit 5

Aufgabe 4.3 - Amortisation

(a) Definieren Sie ein zulässiges Amortisationsschema $\Delta : \sigma_1, \sigma_2, \ldots \to \mathbb{R}$ der Bankkonto-Methode, sodass für jede Inkrementoperation σ_k gilt:

$$A(\sigma_k) = T(\sigma_k) + \Delta(\sigma_k) = \frac{10}{9}$$

(b) Zeigen Sie, dass Ihr Amortisationsschema zulässig ist, indem Sie nachweisen, dass das Tokenkonto stets nichtnegativ ist.

Verwenden Sie folgende Definitionen:

$S_{x \to x+1}(\sigma_k)$ #Stellen die σ_k von x zu $x+1$ setzt

$S_{9 \to 0}(\sigma_k)$ #Stellen die σ_k von 9 zu 0 setzt

Aufgabe 4.3 - Amortisation (a)

Aufgabe 4.3 - Amortisation (b)

Aufgabe 4.4 - Dynamisches Array

$\alpha > \beta > 0$	Wachstums-/Verkleinerungs-Faktoren $(\alpha, \beta \in \mathbb{N})$
n	Aktuelle Anzahl der Elemente im Array
w	Größe des Arrays

Operationen:

- `pushBack`: Einfügen am Ende des Arrays
- `popBack`: Löschen des letzten Elements
- `reallocate`: Vergrößern/Verkleinern des Arrays

Wann wird `reallocate` aufgerufen?

- Array voll ($n = w$) und `pushBack` wird aufgerufen
  $\Rightarrow$ neues Array mit Größe βn + Kopieren und Einfügen
- Nach `popBack` zu leer (d.h. $\alpha \cdot n \leq w$)
  $\Rightarrow$ neues Array mit Größe βn + Kopieren

Aufgabe 4.4 - Dynamisches Array

(a) Zeigen Sie, dass dieses Amortisationsschema zulässig ist, indem Sie zeigen, dass das Tokenkonto zu jedem Zeitpunkt nichtnegativ ist.

Hinweis: Bezeichne n_1 die Zahl der Elemente unmittelbar nach einem `reallocate` (bei `pushBack` also vor dem Einfügen des neuen Elements). Die Arraygröße zu diesem Zeitpunkt ist $w_1 := \beta \cdot n_1$ und es sind $w_1 - n_1$ Positionen frei. Das nächste `reallocate` wird erst dann aufgerufen, wenn für die Zahl n der Elemente entweder $n = w_1$ oder $\alpha n \leq w_1$ gilt.

(b) Zeigen Sie, dass unter diesem Amortisationsschema die amortisierte Laufzeit jeder Operation in $\mathcal{O}(1)$ liegt, und folgern Sie, dass die Worst-Case-Laufzeit für Operationsfolgen der Länge m in $\mathcal{O}(m)$ liegt.

Aufgabe 4.4 - Dynamisches Array (a)

Aufgabe 4.4 - Dynamisches Array (a)

Aufgabe 4.4 - Dynamisches Array (a)

Aufgabe 4.4 - Dynamisches Array (b)

E-Aufgaben

- Aufgabe 4.5 - Quicksort Worstcase
  - Worstcase Suche für verschiedene Pivot-Wahlen
- Aufgabe 4.6 - Noch mehr Spaß mit Δmortisation
  - Weitere Übung zum Finden von Amortisationschema

Hausaufgaben

- Hausaufgabe 3 - Dynamisches Array
  (Deadline: 28.05.2025)
- Hausaufgabe 4 - Verbessertes Sortieren
  (Deadline: 28.05.2025)

Fragen?

- Nach Übung gerne bei mir melden
- Tutoriumschannel oder DM an mich auf Zulip
- Vorlesungschannels von GAD auf Zulip (insbesondere bei Hausaufgaben)

Feedback oder Verbesserungsvorschläge?
Gerne nach dem Tutorium mit mir quatschen oder DM auf Zulip

Bis nächste Woche!