Tutorium
Grundlagen: Algorithmen und Datenstrukturen

Übungsblatt Woche 4

Aufgabe 4.1
Sortierverfahren

Aufgabe 4.1

Gegeben sei eine Zahlenfolge 523, 126, 67, 1, 500, 34, 21, 229, 9, 123, 13, die mit verschiedenen Verfahren sortiert werden soll.

a) Sortieren Sie die Zahlenfolge mit Mergesort. Geben Sie für jede Rekursionsebene jeweils für das Aufspalten der Teilsequenzen und für das Verschmelzen der sortierten Teilsequenzen einen Zwischenschritt an (d.h. bei dieser Eingabesequenz insgesamt circa acht Zwischenschritte), sodass Ihr Vorgehen nachvollzogen werden kann. Falls die Teilsequenz beim Aufspalten eine ungerade Anzahl an Elementen hat, soll die linke Teilsequenz nach dem Aufspalten ein Element mehr enthalten.

b) Sortieren Sie die Zahlenfolge mit Quicksort. Benutzen Sie dafür wie in der Vorlesung gelernt das letzte Element als 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. Wenn eine Teilsequenz nur noch aus maximal 2 Elementen besteht, dürfen Sie der Einfachheit halber die Rekursion für diese Teilsequenz beenden, und bei Bedarf beide Elemente direkt in die jeweils richtige Position tauschen.

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

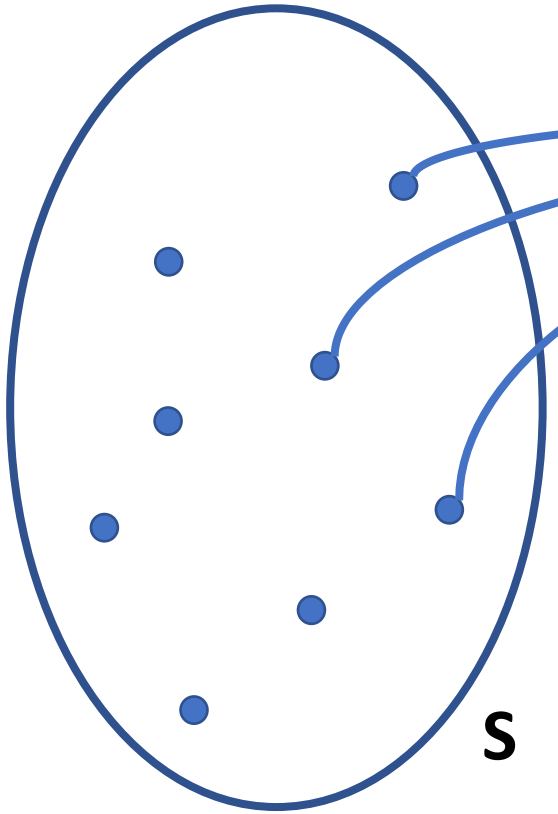
Aufgabe 4.1 (a)

Aufgabe 4.1 (b)

Aufgabe 4.1 (c)

Aufgabe 4.2
Bankkonto Tutorial

Aufgabe 4.2



Menge an Operationen

Folge	σ_1	σ_2	σ_3
Laufzeit	$T(\sigma_1)$	$T(\sigma_2)$	$T(\sigma_3)$
Tokens	$\Delta(\sigma_1)$	$\Delta(\sigma_2)$	$\Delta(\sigma_3)$
am. Laufzeit	$A(\sigma_1)$	$A(\sigma_2)$	$A(\sigma_3)$

$$A(\sigma_i) = T(\sigma_i) + \Delta(\sigma_i)$$

Gesucht ist Δ, wobei:

1) **Summe aller Tokens darf nie negativ sein**
2) **Δ muss möglichst gut gewählt sein**

Aufgabe 4.2

Zu Bedingung 1:

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

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

- Die Amortisierte Laufzeit ist eine obere Schranke für die tatsächliche Laufzeit.
- Die Tokens geben an wieviel die amortisierte Laufzeit die tatsächliche Laufzeit übersteigt (obere Schranke T)

Zu Bedingung 2:

Ziel:
Amortisierte Laufzeit von Operationsfolgen soll möglichst gering sein

Ansatz:
Wähle Δ, sodass die größte am. Laufzeit einer Einzeloperation minimal wird

Erreicht:
Obere Schranke O(m*max(A(σ))) für Worst-Case
(m Anzahl Operationen)

Aufgabe 4.3
Amortisation

Aufgabe 4.3

Wir betrachten eine Folge von $m \geq 1$ Inkrement-Operationen iterativ angewandt auf 0, das heißt, dass die k-te Inkrementoperation σ_k die Zahl $k-1$ zur Zahl k inkrementiert. Wir nehmen an, dass alle Zahlen in *Dezimalschreibweise* geschrieben werden, also mit den Ziffern $0, 1, \ldots, 9$. Außerdem nehmen wir (vereinfachend) an, dass die Änderung einer Stelle (von x zu $x+1$ für alle $x \in \{0, \ldots, 8\}$ bzw. von 9 zu 0) Laufzeit 1 hat und die Laufzeit jeder Inkrement-Operation gerade die Anzahl der Stellen ist, die durch die Operation geändert werden.

(a) Definieren Sie ein zulässiges Amortisationsschema $\Delta : \{\sigma_1, \sigma_2, \ldots\} \to \mathbb{R}$ der Bankkonto-Methode, sodass für jede Inkrementoperation σ_k die amortisierte Laufzeit $A(\sigma_k) = T(\sigma_k) + \Delta(\sigma_k)$ gerade $\frac{10}{9}$ beträgt (hierbei sei $T(\sigma_k)$ die tatsächliche Laufzeit von σ_k).

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

Hinweis: Verwenden Sie in Ihrem Amortisationsschema für jede Operation σ_k die Anzahl $S_{x \to x+1}(\sigma_k)$ der Stellen, die durch σ_k von x zu $x+1$ geändert werden, sowie die Anzahl $S_{9 \to 0}(\sigma_k)$ der Stellen, die durch σ_k von 9 zu 0 geändert werden.

Aufgabe 4.3 (a)

Definieren Sie ein zulässiges Amortisationsschema $\Delta: \{\sigma_1, \sigma_2, \ldots\} \rightarrow \mathbb{R}$ der Bankkonto-Methode, sodass für jede Inkrementoperation σ_k die amortisierte Laufzeit $A(\sigma_k) = T(\sigma_k) + \Delta(\sigma_k)$ gerade $\frac{10}{9}$ beträgt (hierbei sei $T(\sigma_k)$ die tatsächliche Laufzeit von σ_k).

Aufgabe 4.3 (b)

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

Aufgabe 4.4
Dynamisches Array

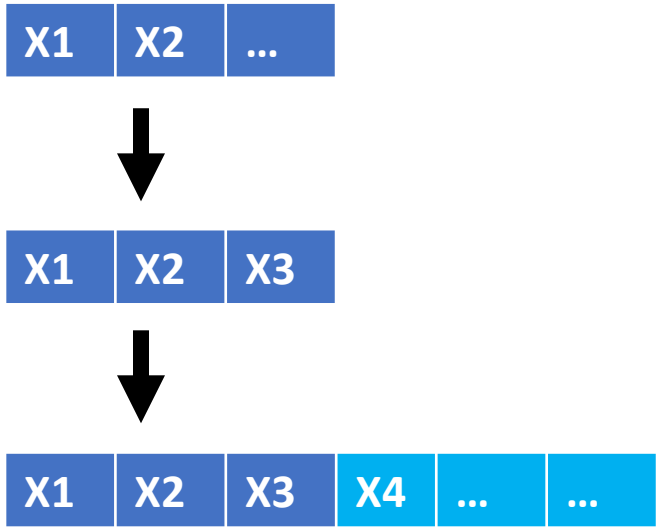
Aufgabe 4.4

Aus der Übung bekanntes dynamisches Array:

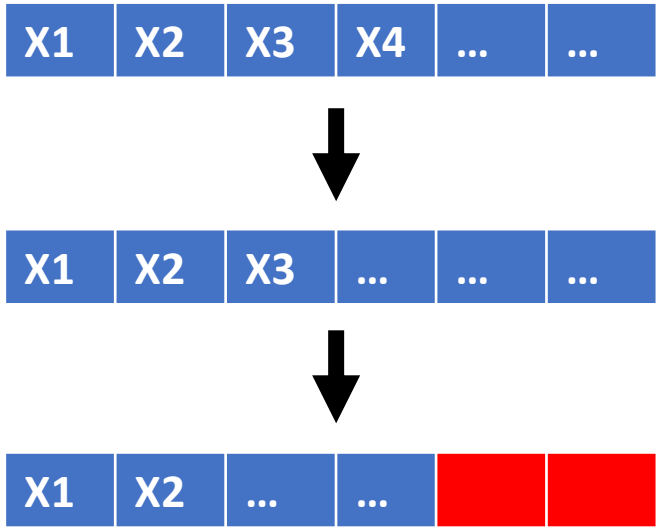
$\alpha > \beta > 1$ und $\alpha, \beta \in \mathbb{N}$ **n**: aktuelle Elementzahl **w**: aktuelle Array-Größe

- Reallocate bei pushBack auf **vollem Array**: Vergrößern zu βn
- Reallocate bei popBack wenn nach Löschen $\alpha n \leq w$: Verkleinern zu βn

Push-Back (α=3)



Pop-Back (β=2)



Aufgabe 4.4

Es ist nicht schwierig zu sehen, dass die tatsächliche Laufzeit von `pushBack` und `popBack` konstant ist (d.h. durch eine Konstante nach oben beschränkt), und die Laufzeit von `reallocate` durch $\mathcal{O}(1) + c \cdot n$ beschränkt ist, wobei c eine Konstante ist und n die Zahl der kopierten Elemente. Wie in der Vorlesung ist es legitim, $c = 1$ zu setzen, indem wir annehmen, dass wir die Laufzeit in einer Einheit messen, die gerade der Laufzeit einer einzelnen Kopieroperation entspricht. Wir erhalten damit:

$$
\begin{aligned}
T(\texttt{pushBack}) &\in \mathcal{O}(1) \\
T(\texttt{popBack}) &\in \mathcal{O}(1) \\
T(\texttt{reallocate}) &= \mathcal{O}(1) + n, \text{ wobei } n = \text{Anzahl der kopierten Elemente}
\end{aligned}
$$

Ziel dieser Aufgabe ist der Nachweis mit einer amortisierten Analyse, dass die Laufzeit von Operationen der Länge m auf einem zu Beginn leeren dynamischen Array in $\mathcal{O}(m)$ liegt (zu Beginn hat das Array Größe 1).

Aufgabe 4.4

$$
\begin{aligned}
\Delta(\texttt{pushBack}) &= \beta/(\beta-1) \\
\Delta(\texttt{popBack}) &= \beta/(\alpha-\beta) \\
\Delta(\texttt{reallocate}) &= -n, \text{ wobei } n = \text{Anzahl der kopierten Elemente}
\end{aligned}
$$

(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` (im Falle von `pushBack` also noch vor der Einfügung des neuen Elements). Machen Sie sich klar, dass das Array *zu diesem Zeitpunkt* Größe $w_1 := \beta \cdot n_1$ hat, und $w_1 - n_1$ Positionen frei sind. 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 (a)

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

$$\Delta(\text{pushBack}) = \beta/(\beta - 1)$$
$$\Delta(popBack) = \beta/(\alpha - \beta)$$
$$\Delta(reallocate) = -n \quad \text{wobei } n = \text{Anzahl der kopierten Elemente}$$

Hinweis: Sei n_1 die Zahl der Elemente ***unmittelbar*** nach einem *reallocate (Array-Größe: $w_1 = \beta n_1$, freier Platz: $w_1 - n_1$)*

Aufgabe 4.4 (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.