

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)

Aufteilen

$$\left\{ \begin{array}{ccccccccc}
 523 & 126 & 67 & 1 & 500 & 34 & | & 21 & 229 & 9 & 123 & 13 \\
 523 & 126 & 67 & | & 1 & 500 & 34 & || & 21 & 229 & 9 & | & 123 & 13 \\
 523 & 126 & | & 67 & || & 1 & 500 & | & 34 & ||| & 21 & 229 & | & 9 & || & 123 & | & 13 \\
 523 & | & 126 & || & 67 & ||| & 1 & | & 500 & || & 34 & |||| & 21 & 229 & || & 9 & ||| & 123 & || & 13
 \end{array} \right.$$

Verschmelzen

$$\left\{ \begin{array}{ccccccccc}
 126 & 523 & | & 67 & || & 1 & 500 & | & 34 & ||| & 21 & 229 & | & 9 & || & 123 & | & 13 \\
 67 & 126 & 523 & | & 1 & 34 & 500 & || & 9 & 21 & 229 & | & 13 & 123 \\
 1 & 34 & 67 & 126 & 500 & 523 & | & 9 & 13 & 21 & 123 & 229 \\
 1 & 9 & 13 & 21 & 34 & 67 & 123 & 126 & 229 & 500 & 523
 \end{array} \right.$$

Aufgabe 4.1 (b)

$p=13$: $[523 \quad 126 \quad 67 \quad 1 \quad 500 \quad 34 \quad 21 \quad 229 \quad 9 \quad 123 \quad 13]$

$p=67$ $\{1\} \{9\} \{13\} [126 \quad 500 \quad 34 \quad 21 \quad 229 \quad 523 \quad 123 \quad 67]$

$p=500$ $\{1\} \{9\} \{13\} \{21\} \{34\} \{67\} [126 \quad 229 \quad 523 \quad 123 \quad 500]$

$p=123$ $\{1\} \{9\} \{13\} \{21\} \{34\} \{67\} \{126 \quad 229 \quad 123\} [500] [523]$
1 9 13 21 34 67 123 126 229 500 523

Teilarrays mit nur 2 Elementen können in dieser Aufgabe direkt ohne Zwischenschritt
sortiert werden

Aufgabe 4.1 (c)

Mergesort : Worst-Case = Best-Case = Average-Case : $O(n \cdot \log(n))$

↪ Algorithmus nicht abhängig von Aufbau des Arrays (für alle Eingaben gleicher Länge gleich)

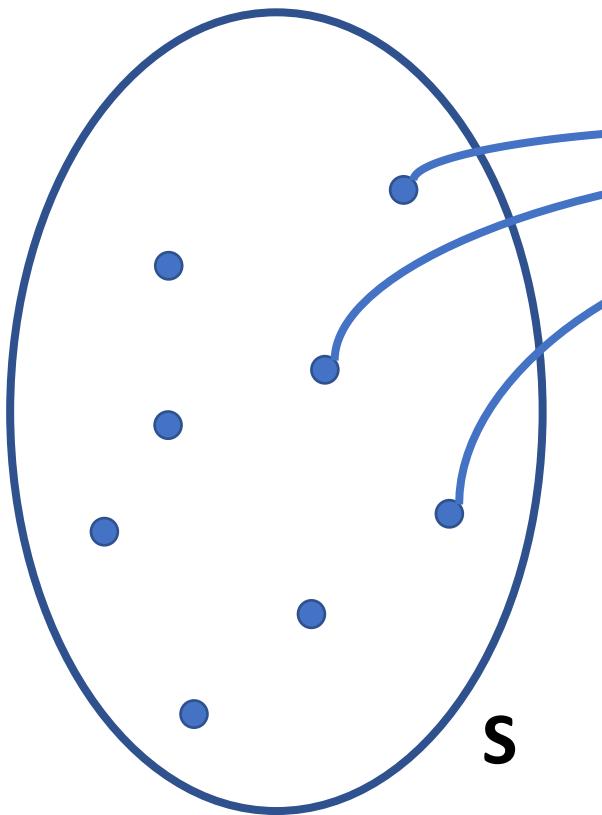
Quicksort : Worst-Case : $O(n^2)$ → Mit unserer Pivotwahl auf schon sortiertem Array

Avg-Case : $O(n \log(n))$ → Das Pivotelement ist dann ein zufälliges Element und darum teilt es das Array relativ gleichmäßig

Aufgabe 4.2

Bankkonto Tutorial

Aufgabe 4.2



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	$\underline{\underline{A(\sigma_1)}}$	$\underline{\underline{A(\sigma_2)}}$	$\boxed{\underline{\underline{A(\sigma_3)}}}$

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

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:

$$\begin{aligned} A(\sigma_1, \dots, \sigma_m) &= \sum_{i=1}^m A(\sigma_i) = \sum_{i=1}^m (T(\sigma_i) + \Delta(\sigma_i)) \\ &= T(\sigma_1, \dots, \sigma_m) + \underbrace{\sum_{i=1}^m \Delta(\sigma_i)}_{\geq 0} \geq T(\sigma_1, \dots, \sigma_m) \end{aligned}$$

- 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 $O(m * \max(A(\sigma)))$ 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, …, 9. Außerdem nehmen wir (vereinfachend) an, dass die Änderung einer Stelle (von x zu $x + 1$ für alle $x \in \{0, \dots, 8\}$ 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.

›

- Definieren Sie ein zulässiges Amortisationsschema $\Delta : \{\sigma_1, \sigma_2, \dots\} \rightarrow \mathbb{R}$ 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).
- 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 $S_{x \rightarrow x+1}(\sigma_k)$ der Stellen, die durch σ_k von x zu $x + 1$ geändert werden, sowie die Anzahl $S_{9 \rightarrow 0}(\sigma_k)$ der Stellen, die durch σ_k von 9 zu 0 geändert werden.

Aufgabe 4.3 (a)

Definieren Sie ein zulässiges Amortisationsschema $\Delta: \{\sigma_1, \sigma_2, \dots\} \rightarrow \mathbb{R}$ 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).

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

$\Rightarrow \Delta(\sigma_k) = \frac{10}{9} - T(\sigma_k)$

$= \frac{10}{9} - S_{x \rightarrow x+1}(\sigma_k) - S_{g \rightarrow 0}(\sigma_k)$

Laufzeit ist Anzahl der veränderten Stellen $\Rightarrow \# \text{Stellen}(x \rightarrow x+1) + \# \text{Stellen}(g \rightarrow 0)$

$T(\sigma_k) = \underbrace{S_{x \rightarrow x+1}(\sigma_k)}_{\text{Aus Angabe}} + \underbrace{S_{g \rightarrow 0}(\sigma_k)}_{\text{Aus Angabe}}$

- $A(\sigma_k) = T(\sigma_k) + \Delta(\sigma_k)$
 $= \underbrace{S_{x \rightarrow x+1}(\sigma_k)}_{\text{Aus Angabe}} + \underbrace{S_{g \rightarrow 0}(\sigma_k)}_{\text{Aus Angabe}} + \frac{10}{9} - \underbrace{S_{x \rightarrow x+1}(\sigma_k)}_{\text{Aus Angabe}} - \underbrace{S_{g \rightarrow 0}(\sigma_k)}_{\text{Aus Angabe}} = \frac{10}{9} \checkmark$

Aufgabe 4.3 (b)

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

- Wir halten fest:

$$\Delta(\zeta_k) = 10/g - S_{x \rightarrow x+1} - S_{g \rightarrow 0}$$

→ Beachte, bei einer Inkrementoperation ist $S_{x \rightarrow x+1}$ immer = 1 (die erste Stelle von rechts die nicht von 9 zu 0 geändert wird)

- Man könnte darum zusammenfassen: $\underbrace{10/g - 1}_{+ 1/g \text{ für das Erhöhen dieser einen Stelle}} - \underbrace{S_{g \rightarrow 0}}_{- 1 \text{ Abzug pro Stelle die rückgesetzt wird}}$

⇒ Anders betrachtet:

- eine Stelle bezahlt 9 mal $1/g$ ein bevor sie rückgesetzt wird.
- Beim Rücksetzen werden die eingezahlten $9/g = 1$ wieder rausgenommen

{ Fazit: Die Tokenzahl beträgt immer Summe aller Ziffern $\cdot \frac{1}{g}$
⇒ ≥ 0 ✓

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

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

w : aktuelle Array-Größe

Vergrößern zu βn

Verkleinern zu βn

Push-Back (~~$\alpha=3$~~)

Bsp. : $\alpha = 3$ $\beta = 2$



Pop-Back (~~$\beta=2$~~)



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:

$$T(\text{pushBack}) \in \mathcal{O}(1)$$

$$T(\text{popBack}) \in \mathcal{O}(1)$$

$$T(\text{reallocate}) = \mathcal{O}(1) + n, \text{ wobei } n = \text{Anzahl der kopierten Elemente}$$

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

$$\Delta(\text{pushBack}) = \beta / (\beta - 1)$$

$$\Delta(\text{popBack}) = \beta / (\alpha - \beta)$$

$\Delta(\text{reallocate}) = -n$, wobei n = Anzahl der kopierten Elemente

- (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 **realloc**ate (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 **realloc**ate 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.

$$\Delta(\text{pushBack}) = \beta/(\beta - 1)$$

$$\Delta(\text{popBack}) = \beta/(\alpha - \beta)$$

$$\Delta(\text{reallocate}) = -n \quad \text{wobei } n = \text{Anzahl der kopierten Elemente}$$

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

- Nach letzten *reallocate*: #Elemente: n_1 Größe Array: $w_1 = \beta n_1$
- Pushback und Popback zählen nur ein (kein Problem)
 - ↳ Idee:
 - betrachte minimale Anzahl an pushbacks/popbacks bis zum nächsten *reallocate*
 - Finde eingeschlossene Summe, und schaue nach ob es für ein weiteres *reallocate* ausreicht
- Pushback: $(w_1 - n_1)$ frei Stellen die mit Pushback gefüllt werden müssen für ein neues *realloc*.
 $\Rightarrow (w_1 - n_1) \cdot \frac{\beta}{\beta-1} = (\beta n_1 - n_1) \frac{\beta}{\beta-1} = n_1 (\beta-1) \frac{\beta}{\beta-1} = n_1 \cdot \beta = w_1 \Rightarrow$ Genau so viele Elemente muss *realloc* kopieren
Wir haben also genug Tokens und bleiben positiv.
- Popback: Sei n die Zahl der Elemente beim nächsten *realloc*
 $\Rightarrow (n_1 - n)$ Elemente müssen mit popback entfernt werden
um *realloc* durchzuführen
 $\Rightarrow (n_1 - n) \cdot \frac{\beta}{\alpha-\beta} = \frac{\beta n_1 - \beta n}{\alpha-\beta} = \frac{w_1 - \beta n}{\alpha-\beta} \geq \frac{\alpha n - \beta n}{\alpha-\beta} = \frac{(\alpha-\beta)n}{\alpha-\beta} = n$ Mindestens n Tokens
genug eingeschlossen

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.

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

$$\Rightarrow A(\text{pushBack}) = \mathcal{O}(1) + \frac{\beta}{\beta-1} \rightarrow \text{Alles Konstanten} \Rightarrow \mathcal{O}(1)$$
$$A(\text{popBack}) = \mathcal{O}(1) + \frac{\beta}{\alpha-\beta}$$

$$A(\text{reallocate}) = \underbrace{\mathcal{O}(1)}_{T(\dots)} + \underbrace{n - n}_{\Delta(\dots)} = \mathcal{O}(1)$$

$\Rightarrow \sigma_i \in \{\text{pushBack}, \text{popBack}, \text{reallocate}\}$
und für alle dieser Operationen $A(\dots) = \mathcal{O}(1)$ gilt

$$\Rightarrow A(\sigma_1, \dots, \sigma_m) = \sum_{i=1}^m A(\sigma_i) = \sum_{i=1}^m \mathcal{O}(1) = m \cdot \mathcal{O}(1) = \mathcal{O}(m)$$