Algorithmentechnik - Übung 1

2. Sitzung

Tanja Hartmann | 05. November 2009

INSTITUT FÜR THEORETISCHE INFORMATIK, PROF. DR. DOROTHEA WAGNER

Dynamisches Array - Problem 1 **[Kap.** 0.3.4**]**

(Amortisierte Analyse)

Länge vor Verdopplung:	$l(A)$	Inhalt vor Verdopplung:	$l(A)$	Füllgrad $\alpha = 1$
Länge nach Verdopplung:	$l(A') = 2\, l(A)$	Inhalt nach Verdopplung:	$l(A) + 1$	Füllgrad $\alpha > \frac{1}{2}$
Länge vor Halbierung:	$l(A)$	Inhalt vor Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor + 1$	Füllgrad $\alpha > \lfloor \frac{1}{4} \rfloor$
Länge nach Halbierung:	$l(A') = \lfloor \frac{1}{2}\, l(A) \rfloor$	Inhalt nach Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor$	Füllgrad $\alpha = \lfloor \frac{1}{2} \rfloor$

Dynamisches Array - Problem 1 [**Kap.** 0.3.4]

(Amortisierte Analyse)

Länge vor Verdopplung:	$l(A)$	Inhalt vor Verdopplung:	$l(A)$	Füllgrad $\alpha = 1$
Länge nach Verdopplung:	$l(A') = 2\,l(A)$	Inhalt nach Verdopplung:	$l(A) + 1$	Füllgrad $\alpha > \frac{1}{2}$
Länge vor Halbierung:	$l(A)$	Inhalt vor Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor + 1$	Füllgrad $\alpha > \lfloor \frac{1}{4} \rfloor$
Länge nach Halbierung:	$l(A') = \lfloor \frac{1}{2}\, l(A) \rfloor$	Inhalt nach Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor$	Füllgrad $\alpha = \lfloor \frac{1}{2} \rfloor$

Vorgehensweise (Analyse mit Potentialmethode):

Dynamisches Array - Problem 1 [Kap. 0.3.4]

(Amortisierte Analyse)

Länge vor Verdopplung:	$l(A)$	Inhalt vor Verdopplung:	$l(A)$	Füllgrad $\alpha = 1$
Länge nach Verdopplung:	$l(A') = 2\, l(A)$	Inhalt nach Verdopplung:	$l(A) + 1$	Füllgrad $\alpha > \frac{1}{2}$
Länge vor Halbierung:	$l(A)$	Inhalt vor Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor + 1$	Füllgrad $\alpha > \lfloor \frac{1}{4} \rfloor$
Länge nach Halbierung:	$l(A') = \lfloor \frac{1}{2}\, l(A) \rfloor$	Inhalt nach Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor$	Füllgrad $\alpha = \lfloor \frac{1}{2} \rfloor$

Vorgehensweise (Analyse mit Potentialmethode):

- Analysiere Dynamik . . .

Dynamisches Array - Problem 1 [Kap. 0.3.4]

(Amortisierte Analyse)

Länge vor Verdopplung:	$l(A)$	Inhalt vor Verdopplung:	$l(A)$	Füllgrad $\alpha = 1$
Länge nach Verdopplung:	$l(A') = 2\, l(A)$	Inhalt nach Verdopplung:	$l(A) + 1$	Füllgrad $\alpha > \frac{1}{2}$
Länge vor Halbierung:	$l(A)$	Inhalt vor Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor + 1$	Füllgrad $\alpha > \lfloor \frac{1}{4} \rfloor$
Länge nach Halbierung:	$l(A') = \lfloor \frac{1}{2}\, l(A) \rfloor$	Inhalt nach Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor$	Füllgrad $\alpha = \lfloor \frac{1}{2} \rfloor$

Vorgehensweise (Analyse mit Potentialmethode):

- Analysiere Dynamik ...
- Definiere Potentialfunktion $\mathcal{C} : D_i \mapsto \mathcal{C}(D_i)$ mit $\mathcal{C}(D_n) \geq \mathcal{C}(D_0)$ für beliebiges n

Dynamisches Array - Problem 1 [**Kap.** 0.3.4]

(Amortisierte Analyse)

Länge vor Verdopplung:	$l(A)$	Inhalt vor Verdopplung:	$l(A)$	Füllgrad $\alpha = 1$
Länge nach Verdopplung:	$l(A') = 2\, l(A)$	Inhalt nach Verdopplung:	$l(A) + 1$	Füllgrad $\alpha > \frac{1}{2}$
Länge vor Halbierung:	$l(A)$	Inhalt vor Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor + 1$	Füllgrad $\alpha > \lfloor \frac{1}{4} \rfloor$
Länge nach Halbierung:	$l(A') = \lfloor \frac{1}{2}\, l(A) \rfloor$	Inhalt nach Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor$	Füllgrad $\alpha = \lfloor \frac{1}{2} \rfloor$

Vorgehensweise (Analyse mit Potentialmethode):

- Analysiere Dynamik ...
- Definiere Potentialfunktion $\mathcal{C} : D_i \mapsto \mathcal{C}(D_i)$ mit $\mathcal{C}(D_n) \geq \mathcal{C}(D_0)$ für beliebiges n
- Zeige, dass amortisierte Kosten **obere Schranke** der tatsächlichen Kosten sind (also $\mathcal{C}(D_n) \geq \mathcal{C}(D_0)$)

Dynamisches Array - Problem 1 [Kap. 0.3.4]

(Amortisierte Analyse)

Länge vor Verdopplung:	$l(A)$	Inhalt vor Verdopplung:	$l(A)$	Füllgrad $\alpha = 1$
Länge nach Verdopplung:	$l(A') = 2\, l(A)$	Inhalt nach Verdopplung:	$l(A) + 1$	Füllgrad $\alpha > \frac{1}{2}$
Länge vor Halbierung:	$l(A)$	Inhalt vor Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor + 1$	Füllgrad $\alpha > \lfloor \frac{1}{4} \rfloor$
Länge nach Halbierung:	$l(A') = \lfloor \frac{1}{2}\, l(A) \rfloor$	Inhalt nach Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor$	Füllgrad $\alpha = \lfloor \frac{1}{2} \rfloor$

Vorgehensweise (Analyse mit Potentialmethode):

- Analysiere Dynamik ...
- Definiere Potentialfunktion $\mathcal{C} : D_i \mapsto \mathcal{C}(D_i)$ mit $\mathcal{C}(D_n) \geq \mathcal{C}(D_0)$ für beliebiges n
- Zeige, dass amortisierte Kosten **obere Schranke** der tatsächlichen Kosten sind (also $\mathcal{C}(D_n) \geq \mathcal{C}(D_0)$)
- Zeige für jeden Operatiostyp, dass amortisierte Kosten $\hat{c}_i$ **asymptodisch beschränkt** (z.B. konstant) sind (falls i-te Operation vom jeweiligen Typ)

Dynamisches Array - Problem 1 [Kap. 0.3.4]

(Amortisierte Analyse)

Länge vor Verdopplung:	$l(A)$	Inhalt vor Verdopplung:	$l(A)$	Füllgrad $\alpha = 1$
Länge nach Verdopplung:	$l(A') = 2\, l(A)$	Inhalt nach Verdopplung:	$l(A) + 1$	Füllgrad $\alpha > \frac{1}{2}$
Länge vor Halbierung:	$l(A)$	Inhalt vor Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor + 1$	Füllgrad $\alpha > \lfloor \frac{1}{4} \rfloor$
Länge nach Halbierung:	$l(A') = \lfloor \frac{1}{2}\, l(A) \rfloor$	Inhalt nach Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor$	Füllgrad $\alpha = \lfloor \frac{1}{2} \rfloor$

Analyse der Dynamik:

Dynamisches Array - Problem 1 [Kap. 0.3.4]

(Amortisierte Analyse)

Länge vor Verdopplung:	$l(A)$	Inhalt vor Verdopplung:	$l(A)$	Füllgrad $\alpha = 1$
Länge nach Verdopplung:	$l(A') = 2\,l(A)$	Inhalt nach Verdopplung:	$l(A) + 1$	Füllgrad $\alpha > \frac{1}{2}$
Länge vor Halbierung:	$l(A)$	Inhalt vor Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor + 1$	Füllgrad $\alpha > \lfloor \frac{1}{4} \rfloor$
Länge nach Halbierung:	$l(A') = \lfloor \frac{1}{2}\, l(A) \rfloor$	Inhalt nach Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor$	Füllgrad $\alpha = \lfloor \frac{1}{2} \rfloor$

Analyse der Dynamik:

- Array exakt zur Hälfte gefüllt: “Gleichgewicht”
  $\rightsquigarrow$ Potential $= 0$

Dynamisches Array - Problem 1 [**Kap.** 0.3.4]

(Amortisierte Analyse)

Länge vor Verdopplung:	$l(A)$	Inhalt vor Verdopplung:	$l(A)$	Füllgrad $\alpha = 1$
Länge nach Verdopplung:	$l(A') = 2\, l(A)$	Inhalt nach Verdopplung:	$l(A) + 1$	Füllgrad $\alpha > \frac{1}{2}$
Länge vor Halbierung:	$l(A)$	Inhalt vor Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor + 1$	Füllgrad $\alpha > \lfloor \frac{1}{4} \rfloor$
Länge nach Halbierung:	$l(A') = \lfloor \frac{1}{2}\, l(A) \rfloor$	Inhalt nach Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor$	Füllgrad $\alpha = \lfloor \frac{1}{2} \rfloor$

Analyse der Dynamik:

- Array exakt zur Hälfte gefüllt: “Gleichgewicht”
  $\rightsquigarrow$ Potential $= 0$
- **vor** Halbierung: Hälfte der Elemente wird gelöscht
  $\rightsquigarrow$ Kostendeckung für übrige Elemente

Dynamisches Array - Problem 1 [Kap. 0.3.4]

(Amortisierte Analyse)

Länge vor Verdopplung:	$l(A)$	Inhalt vor Verdopplung:	$l(A)$	Füllgrad $\alpha = 1$
Länge nach Verdopplung:	$l(A') = 2\, l(A)$	Inhalt nach Verdopplung:	$l(A) + 1$	Füllgrad $\alpha > \frac{1}{2}$
Länge vor Halbierung:	$l(A)$	Inhalt vor Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor + 1$	Füllgrad $\alpha > \lfloor \frac{1}{4} \rfloor$
Länge nach Halbierung:	$l(A') = \lfloor \frac{1}{2}\, l(A) \rfloor$	Inhalt nach Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor$	Füllgrad $\alpha = \lfloor \frac{1}{2} \rfloor$

Analyse der Dynamik:

- Array exakt zur Hälfte gefüllt: “Gleichgewicht”
  $\rightsquigarrow$ Potential $= 0$
- **vor** Halbierung: Hälfte der Elemente wird gelöscht
  $\rightsquigarrow$ Kostendeckung für übrige Elemente
- Delete: Potential muss um ≥ 1 wachsen

Dynamisches Array - Problem 1 [Kap. 0.3.4]

(Amortisierte Analyse)

Länge vor Verdopplung:	$l(A)$	Inhalt vor Verdopplung:	$l(A)$	Füllgrad $\alpha = 1$
Länge nach Verdopplung:	$l(A') = 2\, l(A)$	Inhalt nach Verdopplung:	$l(A) + 1$	Füllgrad $\alpha > \frac{1}{2}$
Länge vor Halbierung:	$l(A)$	Inhalt vor Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor + 1$	Füllgrad $\alpha > \lfloor \frac{1}{4} \rfloor$
Länge nach Halbierung:	$l(A') = \lfloor \frac{1}{2}\, l(A) \rfloor$	Inhalt nach Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor$	Füllgrad $\alpha = \lfloor \frac{1}{2} \rfloor$

Analyse der Dynamik:

- Array exakt zur Hälfte gefüllt: “Gleichgewicht”
  $\rightsquigarrow$ Potential $= 0$
- **vor** Halbierung: Hälfte der Elemente wird gelöscht
  $\rightsquigarrow$ Kostendeckung für übrige Elemente
- Delete: Potential muss um ≥ 1 wachsen
- **vor** Verdopplung: Elementanzahl wird verdoppelt
  $\rightsquigarrow$ Kostendeckung für schon vorhandene Elemente

Dynamisches Array - Problem 1 [Kap. 0.3.4]

(Amortisierte Analyse)

Länge vor Verdopplung:	$l(A)$	Inhalt vor Verdopplung:	$l(A)$	Füllgrad $\alpha = 1$
Länge nach Verdopplung:	$l(A') = 2\, l(A)$	Inhalt nach Verdopplung:	$l(A) + 1$	Füllgrad $\alpha > \frac{1}{2}$
Länge vor Halbierung:	$l(A)$	Inhalt vor Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor + 1$	Füllgrad $\alpha > \lfloor \frac{1}{4} \rfloor$
Länge nach Halbierung:	$l(A') = \lfloor \frac{1}{2}\, l(A) \rfloor$	Inhalt nach Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor$	Füllgrad $\alpha = \lfloor \frac{1}{2} \rfloor$

Analyse der Dynamik:

- Array exakt zur Hälfte gefüllt: “Gleichgewicht”
  $\rightsquigarrow$ Potential $= 0$
- **vor** Halbierung: Hälfte der Elemente wird gelöscht
  $\rightsquigarrow$ Kostendeckung für übrige Elemente
- Delete: Potential muss um ≥ 1 wachsen
- **vor** Verdopplung: Elementanzahl wird verdoppelt
  $\rightsquigarrow$ Kostendeckung für schon vorhandene Elemente
- Insert: Potential muss um ≥ 2 wachsen

Dynamisches Array - Problem 1 [Kap. 0.3.4]

(Amortisierte Analyse)

Länge vor Verdopplung:	$l(A)$	Inhalt vor Verdopplung:	$l(A)$	Füllgrad $\alpha = 1$
Länge nach Verdopplung:	$l(A') = 2\, l(A)$	Inhalt nach Verdopplung:	$l(A) + 1$	Füllgrad $\alpha > \frac{1}{2}$
Länge vor Halbierung:	$l(A)$	Inhalt vor Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor + 1$	Füllgrad $\alpha > \lfloor \frac{1}{4} \rfloor$
Länge nach Halbierung:	$l(A') = \lfloor \frac{1}{2}\, l(A) \rfloor$	Inhalt nach Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor$	Füllgrad $\alpha = \lfloor \frac{1}{2} \rfloor$

Potentialfunktion:

$$\mathcal{C}(D_i) = \begin{cases} \frac{1}{2}\, l(A_i) - \text{Inhalt}(A_i), & \text{für } \alpha < \frac{1}{2} \\ 2 \times \text{Inhalt}(A_i) - l(A_i), & \text{für } \alpha \geq \frac{1}{2} \end{cases}$$

Dynamisches Array - Problem 1 [Kap. 0.3.4]

(Amortisierte Analyse)

Länge vor Verdopplung:	$l(A)$	Inhalt vor Verdopplung:	$l(A)$	Füllgrad $\alpha = 1$
Länge nach Verdopplung:	$l(A') = 2\,l(A)$	Inhalt nach Verdopplung:	$l(A) + 1$	Füllgrad $\alpha > \frac{1}{2}$
Länge vor Halbierung:	$l(A)$	Inhalt vor Halbierung:	$\lfloor \frac{1}{4}\,l(A) \rfloor + 1$	Füllgrad $\alpha > \lfloor \frac{1}{4} \rfloor$
Länge nach Halbierung:	$l(A') = \lfloor \frac{1}{2}\,l(A) \rfloor$	Inhalt nach Halbierung:	$\lfloor \frac{1}{4}\,l(A) \rfloor$	Füllgrad $\alpha = \lfloor \frac{1}{2} \rfloor$

Potentialfunktion:

$$\mathcal{C}(D_i) = \begin{cases} \frac{1}{2}\,l(A_i) - \text{Inhalt}(A_i), & \text{für } \alpha < \frac{1}{2} \\ 2 \times \text{Inhalt}(A_i) - l(A_i), & \text{für } \alpha \geq \frac{1}{2} \end{cases}$$

- Gleichgewichtszustand: $\alpha = \frac{1}{2} \rightsquigarrow \mathcal{C}(D_{i-1}) = 0$

Dynamisches Array - Problem 1 [**Kap.** 0.3.4]

(Amortisierte Analyse)

Länge vor Verdopplung:	$l(A)$	Inhalt vor Verdopplung:	$l(A)$	Füllgrad $\alpha = 1$
Länge nach Verdopplung:	$l(A') = 2\, l(A)$	Inhalt nach Verdopplung:	$l(A) + 1$	Füllgrad $\alpha > \frac{1}{2}$
Länge vor Halbierung:	$l(A)$	Inhalt vor Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor + 1$	Füllgrad $\alpha > \lfloor \frac{1}{4} \rfloor$
Länge nach Halbierung:	$l(A') = \lfloor \frac{1}{2}\, l(A) \rfloor$	Inhalt nach Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor$	Füllgrad $\alpha = \lfloor \frac{1}{2} \rfloor$

Potentialfunktion:

$$\mathcal{C}(D_i) = \begin{cases} \frac{1}{2}\, l(A_i) - \text{Inhalt}(A_i), & \text{für } \alpha < \frac{1}{2} \\ 2 \times \text{Inhalt}(A_i) - l(A_i), & \text{für } \alpha \geq \frac{1}{2} \end{cases}$$

- Gleichgewichtszustand: $\alpha = \frac{1}{2} \rightsquigarrow \mathcal{C}(D_{i-1}) = 0$
- **vor** Insert mit Umkopieren: $\alpha = 1 \rightsquigarrow \mathcal{C}(D_{i-1}) = l(A_{i-1})$

Dynamisches Array - Problem 1 [Kap. 0.3.4]

(Amortisierte Analyse)

Länge vor Verdopplung:	$l(A)$	Inhalt vor Verdopplung:	$l(A)$	Füllgrad $\alpha = 1$
Länge nach Verdopplung:	$l(A') = 2\,l(A)$	Inhalt nach Verdopplung:	$l(A) + 1$	Füllgrad $\alpha > \frac{1}{2}$
Länge vor Halbierung:	$l(A)$	Inhalt vor Halbierung:	$\lfloor \frac{1}{4}\,l(A) \rfloor + 1$	Füllgrad $\alpha > \lfloor \frac{1}{4} \rfloor$
Länge nach Halbierung:	$l(A') = \lfloor \frac{1}{2}\,l(A) \rfloor$	Inhalt nach Halbierung:	$\lfloor \frac{1}{4}\,l(A) \rfloor$	Füllgrad $\alpha = \lfloor \frac{1}{2} \rfloor$

Potentialfunktion:

$$\mathcal{C}(D_i) = \begin{cases} \frac{1}{2}\,l(A_i) - \text{Inhalt}(A_i), & \text{für } \alpha < \frac{1}{2} \\ 2 \times \text{Inhalt}(A_i) - l(A_i), & \text{für } \alpha \geq \frac{1}{2} \end{cases}$$

- Gleichgewichtszustand: $\alpha = \frac{1}{2} \rightsquigarrow \mathcal{C}(D_{i-1}) = 0$
- **vor** Insert mit Umkopieren: $\alpha = 1 \rightsquigarrow \mathcal{C}(D_{i-1}) = l(A_{i-1})$
- **vor** Delete mit Umkopieren: $\mathcal{C}(D_{i-1}) = \frac{1}{4}\,l(A_{i-1}) - 1$

Dynamisches Array - Problem 1 [Kap. 0.3.4]

(Amortisierte Analyse)

Länge vor Verdopplung:	$l(A)$	Inhalt vor Verdopplung:	$l(A)$	Füllgrad $\alpha = 1$
Länge nach Verdopplung:	$l(A') = 2\, l(A)$	Inhalt nach Verdopplung:	$l(A) + 1$	Füllgrad $\alpha > \frac{1}{2}$
Länge vor Halbierung:	$l(A)$	Inhalt vor Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor + 1$	Füllgrad $\alpha > \lfloor \frac{1}{4} \rfloor$
Länge nach Halbierung:	$l(A') = \lfloor \frac{1}{2}\, l(A) \rfloor$	Inhalt nach Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor$	Füllgrad $\alpha = \lfloor \frac{1}{2} \rfloor$

Obere Schranke:

$$\mathcal{C}(D_i) = \begin{cases} \frac{1}{2}\, l(A_i) - \text{Inhalt}(A_i), & \text{für } \alpha < \frac{1}{2} \\ 2 \times \text{Inhalt}(A_i) - l(A_i), & \text{für } \alpha \geq \frac{1}{2} \end{cases}$$

Dynamisches Array - Problem 1 [Kap. 0.3.4]

(Amortisierte Analyse)

Länge vor Verdopplung:	$l(A)$	Inhalt vor Verdopplung:	$l(A)$	Füllgrad $\alpha = 1$
Länge nach Verdopplung:	$l(A') = 2\,l(A)$	Inhalt nach Verdopplung:	$l(A) + 1$	Füllgrad $\alpha > \frac{1}{2}$
Länge vor Halbierung:	$l(A)$	Inhalt vor Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor + 1$	Füllgrad $\alpha > \lfloor \frac{1}{4} \rfloor$
Länge nach Halbierung:	$l(A') = \lfloor \frac{1}{2}\, l(A) \rfloor$	Inhalt nach Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor$	Füllgrad $\alpha = \lfloor \frac{1}{2} \rfloor$

Obere Schranke:

$$\mathcal{C}(D_i) = \begin{cases} \frac{1}{2}\, l(A_i) - \text{Inhalt}(A_i), & \text{für } \alpha < \frac{1}{2} \\ 2 \times \text{Inhalt}(A_i) - l(A_i), & \text{für } \alpha \geq \frac{1}{2} \end{cases}$$

- D_0 ist Zustand vor erster Operation: $\mathcal{C}(D_0) = 0$

Dynamisches Array - Problem 1 [Kap. 0.3.4]

(Amortisierte Analyse)

Länge vor Verdopplung:	$l(A)$	Inhalt vor Verdopplung:	$l(A)$	Füllgrad $\alpha = 1$
Länge nach Verdopplung:	$l(A') = 2\, l(A)$	Inhalt nach Verdopplung:	$l(A) + 1$	Füllgrad $\alpha > \frac{1}{2}$
Länge vor Halbierung:	$l(A)$	Inhalt vor Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor + 1$	Füllgrad $\alpha > \lfloor \frac{1}{4} \rfloor$
Länge nach Halbierung:	$l(A') = \lfloor \frac{1}{2}\, l(A) \rfloor$	Inhalt nach Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor$	Füllgrad $\alpha = \lfloor \frac{1}{2} \rfloor$

Obere Schranke:

$$\mathcal{C}(D_i) = \begin{cases} \frac{1}{2}\, l(A_i) - \text{Inhalt}(A_i), & \text{für } \alpha < \frac{1}{2} \\ 2 \times \text{Inhalt}(A_i) - l(A_i), & \text{für } \alpha \geq \frac{1}{2} \end{cases}$$

- D_0 ist Zustand vor erster Operation: $\mathcal{C}(D_0) = 0$
- D_n ist Zustand nach n-ter Operation: $\mathcal{C}(D_n) \geq 0$

Dynamisches Array - Problem 1 [Kap. 0.3.4]

(Amortisierte Analyse)

Länge vor Verdopplung:	$l(A)$	Inhalt vor Verdopplung:	$l(A)$	Füllgrad $\alpha = 1$
Länge nach Verdopplung:	$l(A') = 2\, l(A)$	Inhalt nach Verdopplung:	$l(A) + 1$	Füllgrad $\alpha > \frac{1}{2}$
Länge vor Halbierung:	$l(A)$	Inhalt vor Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor + 1$	Füllgrad $\alpha > \lfloor \frac{1}{4} \rfloor$
Länge nach Halbierung:	$l(A') = \lfloor \frac{1}{2}\, l(A) \rfloor$	Inhalt nach Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor$	Füllgrad $\alpha = \lfloor \frac{1}{2} \rfloor$

Obere Schranke:

$$\mathcal{C}(D_i) = \begin{cases} \frac{1}{2}\, l(A_i) - \text{Inhalt}(A_i), & \text{für } \alpha < \frac{1}{2} \\ 2 \times \text{Inhalt}(A_i) - l(A_i), & \text{für } \alpha \geq \frac{1}{2} \end{cases}$$

- D_0 ist Zustand vor erster Operation: $\mathcal{C}(D_0) = 0$
- D_n ist Zustand nach n-ter Operation: $\mathcal{C}(D_n) \geq 0$
- $\mathcal{C}(D_n) - \mathcal{C}(D_0) \geq 0 \rightsquigarrow$ Obere Schranke!

Dynamisches Array - Problem 1 [Kap. 0.3.4]

(Amortisierte Analyse)

Länge vor Verdopplung:	$l(A)$	Inhalt vor Verdopplung:	$l(A)$	Füllgrad $\alpha = 1$
Länge nach Verdopplung:	$l(A') = 2\, l(A)$	Inhalt nach Verdopplung:	$l(A) + 1$	Füllgrad $\alpha > \frac{1}{2}$
Länge vor Halbierung:	$l(A)$	Inhalt vor Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor + 1$	Füllgrad $\alpha > \lfloor \frac{1}{4} \rfloor$
Länge nach Halbierung:	$l(A') = \lfloor \frac{1}{2}\, l(A) \rfloor$	Inhalt nach Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor$	Füllgrad $\alpha = \lfloor \frac{1}{2} \rfloor$

Beschränktheit amortisierter Kosten:

$$\mathcal{C}(D_i) = \begin{cases} \frac{1}{2}\, l(A_i) - \text{Inhalt}(A_i), & \text{für } \alpha < \frac{1}{2} \\ 2 \times \text{Inhalt}(A_i) - l(A_i), & \text{für } \alpha \geq \frac{1}{2} \end{cases}$$

Dynamisches Array - Problem 1 [Kap. 0.3.4]

(Amortisierte Analyse)

Länge vor Verdopplung:	$l(A)$	Inhalt vor Verdopplung:	$l(A)$	Füllgrad $\alpha = 1$
Länge nach Verdopplung:	$l(A') = 2\, l(A)$	Inhalt nach Verdopplung:	$l(A) + 1$	Füllgrad $\alpha > \frac{1}{2}$
Länge vor Halbierung:	$l(A)$	Inhalt vor Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor + 1$	Füllgrad $\alpha > \lfloor \frac{1}{4} \rfloor$
Länge nach Halbierung:	$l(A') = \lfloor \frac{1}{2}\, l(A) \rfloor$	Inhalt nach Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor$	Füllgrad $\alpha = \lfloor \frac{1}{2} \rfloor$

Beschränktheit amortisierter Kosten:

$$\mathcal{C}(D_i) = \begin{cases} \frac{1}{2}\, l(A_i) - \text{Inhalt}(A_i), & \text{für } \alpha < \frac{1}{2} \\ 2 \times \text{Inhalt}(A_i) - l(A_i), & \text{für } \alpha \geq \frac{1}{2} \end{cases}$$

- Insert ohne Umkopieren:

  $\alpha(A_{i-1}), \alpha(A_i) < \frac{1}{2}$:

  $\alpha(A_{i-1}), \alpha(A_i) \geq \frac{1}{2}$:

  $\alpha(A_{i-1}) < \frac{1}{2}, \alpha(A_i) = \frac{1}{2}$,

Dynamisches Array - Problem 1 [Kap. 0.3.4]

(Amortisierte Analyse)

Länge vor Verdopplung:	$l(A)$	Inhalt vor Verdopplung:	$l(A)$	Füllgrad $\alpha = 1$
Länge nach Verdopplung:	$l(A') = 2\, l(A)$	Inhalt nach Verdopplung:	$l(A) + 1$	Füllgrad $\alpha > \frac{1}{2}$
Länge vor Halbierung:	$l(A)$	Inhalt vor Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor + 1$	Füllgrad $\alpha > \lfloor \frac{1}{4} \rfloor$
Länge nach Halbierung:	$l(A') = \lfloor \frac{1}{2}\, l(A) \rfloor$	Inhalt nach Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor$	Füllgrad $\alpha = \lfloor \frac{1}{2} \rfloor$

Beschränktheit amortisierter Kosten:

$$\mathcal{C}(D_i) = \begin{cases} \frac{1}{2}\, l(A_i) - \text{Inhalt}(A_i), & \text{für } \alpha < \frac{1}{2} \\ 2 \times \text{Inhalt}(A_i) - l(A_i), & \text{für } \alpha \geq \frac{1}{2} \end{cases}$$

- Insert ohne Umkopieren:

$\alpha(A_{i-1}), \alpha(A_i) < \frac{1}{2}$:

$\alpha(A_{i-1}), \alpha(A_i) \geq \frac{1}{2}$:

$\alpha(A_{i-1}) < \frac{1}{2}, \alpha(A_i) = \frac{1}{2}, (\alpha(A_i) > \frac{1}{2}$ nur für $m = 1)$:

Dynamisches Array - Problem 1 [Kap. 0.3.4]

(Amortisierte Analyse)

Länge vor Verdopplung:	$l(A)$	Inhalt vor Verdopplung:	$l(A)$	Füllgrad $\alpha = 1$
Länge nach Verdopplung:	$l(A') = 2\, l(A)$	Inhalt nach Verdopplung:	$l(A) + 1$	Füllgrad $\alpha > \frac{1}{2}$
Länge vor Halbierung:	$l(A)$	Inhalt vor Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor + 1$	Füllgrad $\alpha > \lfloor \frac{1}{4} \rfloor$
Länge nach Halbierung:	$l(A') = \lfloor \frac{1}{2}\, l(A) \rfloor$	Inhalt nach Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor$	Füllgrad $\alpha = \lfloor \frac{1}{2} \rfloor$

Beschränktheit amortisierter Kosten:

$$\mathcal{C}(D_i) = \begin{cases} \frac{1}{2}\, l(A_i) - \text{Inhalt}(A_i), & \text{für } \alpha < \frac{1}{2} \\ 2 \times \text{Inhalt}(A_i) - l(A_i), & \text{für } \alpha \geq \frac{1}{2} \end{cases}$$

- Insert ohne Umkopieren:

$\alpha(A_{i-1}), \alpha(A_i) < \frac{1}{2}$: $\hat{c}_i = 0$

$\alpha(A_{i-1}), \alpha(A_i) \geq \frac{1}{2}$: $\hat{c}_i = 3$

$\alpha(A_{i-1}) < \frac{1}{2}, \alpha(A_i) = \frac{1}{2}, (\alpha(A_i) > \frac{1}{2}$ nur für $m = 1)$: $\hat{c}_i = 0$

Dynamisches Array - Problem 1 [Kap. 0.3.4]

(Amortisierte Analyse)

Länge vor Verdopplung:	$l(A)$	Inhalt vor Verdopplung:	$l(A)$	Füllgrad $\alpha = 1$
Länge nach Verdopplung:	$l(A') = 2\, l(A)$	Inhalt nach Verdopplung:	$l(A) + 1$	Füllgrad $\alpha > \frac{1}{2}$
Länge vor Halbierung:	$l(A)$	Inhalt vor Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor + 1$	Füllgrad $\alpha > \lfloor \frac{1}{4} \rfloor$
Länge nach Halbierung:	$l(A') = \lfloor \frac{1}{2}\, l(A) \rfloor$	Inhalt nach Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor$	Füllgrad $\alpha = \lfloor \frac{1}{2} \rfloor$

Beschränktheit amortisierter Kosten:

$$\mathcal{C}(D_i) = \begin{cases} \frac{1}{2}\, l(A_i) - \text{Inhalt}(A_i), & \text{für } \alpha < \frac{1}{2} \\ 2 \times \text{Inhalt}(A_i) - l(A_i), & \text{für } \alpha \geq \frac{1}{2} \end{cases}$$

- Delete ohne Umkopieren:

$\alpha(A_{i-1}), \alpha(A_i) < \frac{1}{2}$:

$\alpha(A_{i-1}), \alpha(A_i) \geq \frac{1}{2}$:

$\alpha(A_{i-1}) = \frac{1}{2}, \alpha(A_i) < \frac{1}{2}, (\alpha(A_{i-1}) > \frac{1}{2}$ nur für $m = 1$) :

Dynamisches Array - Problem 1 [Kap. 0.3.4]

(Amortisierte Analyse)

Länge vor Verdopplung:	$l(A)$	Inhalt vor Verdopplung:	$l(A)$	Füllgrad $\alpha = 1$
Länge nach Verdopplung:	$l(A') = 2\, l(A)$	Inhalt nach Verdopplung:	$l(A) + 1$	Füllgrad $\alpha > \frac{1}{2}$
Länge vor Halbierung:	$l(A)$	Inhalt vor Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor + 1$	Füllgrad $\alpha > \lfloor \frac{1}{4} \rfloor$
Länge nach Halbierung:	$l(A') = \lfloor \frac{1}{2}\, l(A) \rfloor$	Inhalt nach Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor$	Füllgrad $\alpha = \lfloor \frac{1}{2} \rfloor$

Beschränktheit amortisierter Kosten:

$$\mathcal{C}(D_i) = \begin{cases} \frac{1}{2}\, l(A_i) - \text{Inhalt}(A_i), & \text{für } \alpha < \frac{1}{2} \\ 2 \times \text{Inhalt}(A_i) - l(A_i), & \text{für } \alpha \geq \frac{1}{2} \end{cases}$$

- Delete ohne Umkopieren:

$\alpha(A_{i-1}), \alpha(A_i) < \frac{1}{2}$: $\hat{c}_i = 2$

$\alpha(A_{i-1}), \alpha(A_i) \geq \frac{1}{2}$: $\hat{c}_i = -1$

$\alpha(A_{i-1}) = \frac{1}{2}, \alpha(A_i) < \frac{1}{2}, (\alpha(A_{i-1}) > \frac{1}{2}$ nur für $m = 1)$: $\hat{c}_i = 2$

Dynamisches Array - Problem 1 [Kap. 0.3.4]

(Amortisierte Analyse)

Länge vor Verdopplung:	$l(A)$	Inhalt vor Verdopplung:	$l(A)$	Füllgrad $\alpha = 1$
Länge nach Verdopplung:	$l(A') = 2\, l(A)$	Inhalt nach Verdopplung:	$l(A) + 1$	Füllgrad $\alpha > \frac{1}{2}$
Länge vor Halbierung:	$l(A)$	Inhalt vor Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor + 1$	Füllgrad $\alpha > \lfloor \frac{1}{4} \rfloor$
Länge nach Halbierung:	$l(A') = \lfloor \frac{1}{2}\, l(A) \rfloor$	Inhalt nach Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor$	Füllgrad $\alpha = \lfloor \frac{1}{2} \rfloor$

Beschränktheit amortisierter Kosten:

$$\mathcal{C}(D_i) = \begin{cases} \frac{1}{2}\, l(A_i) - \text{Inhalt}(A_i), & \text{für } \alpha < \frac{1}{2} \\ 2 \times \text{Inhalt}(A_i) - l(A_i), & \text{für } \alpha \geq \frac{1}{2} \end{cases}$$

- Delete mit Umkopieren:

  $\alpha(A_{i-1}) < \frac{1}{2}, \alpha(A_i) \geq \frac{1}{2}$:

Dynamisches Array - Problem 1 [Kap. 0.3.4]

(Amortisierte Analyse)

Länge vor Verdopplung:	$l(A)$	Inhalt vor Verdopplung:	$l(A)$	Füllgrad $\alpha = 1$
Länge nach Verdopplung:	$l(A') = 2\, l(A)$	Inhalt nach Verdopplung:	$l(A) + 1$	Füllgrad $\alpha > \frac{1}{2}$
Länge vor Halbierung:	$l(A)$	Inhalt vor Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor + 1$	Füllgrad $\alpha > \lfloor \frac{1}{4} \rfloor$
Länge nach Halbierung:	$l(A') = \lfloor \frac{1}{2}\, l(A) \rfloor$	Inhalt nach Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor$	Füllgrad $\alpha = \lfloor \frac{1}{2} \rfloor$

Beschränktheit amortisierter Kosten:

$$\mathcal{C}(D_i) = \begin{cases} \frac{1}{2}\, l(A_i) - \text{Inhalt}(A_i), & \text{für } \alpha < \frac{1}{2} \\ 2 \times \text{Inhalt}(A_i) - l(A_i), & \text{für } \alpha \geq \frac{1}{2} \end{cases}$$

- Delete mit Umkopieren:

  $\alpha(A_{i-1}) < \frac{1}{2}, \alpha(A_i) \geq \frac{1}{2}$: $\hat{c}_i = 2$

Dynamisches Array - Problem 1 **[Kap.** 0.3.4**]**

(Amortisierte Analyse)

Länge vor Verdopplung:	$l(A)$	Inhalt vor Verdopplung:	$l(A)$	Füllgrad $\alpha = 1$
Länge nach Verdopplung:	$l(A') = 2\, l(A)$	Inhalt nach Verdopplung:	$l(A) + 1$	Füllgrad $\alpha > \frac{1}{2}$
Länge vor Halbierung:	$l(A)$	Inhalt vor Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor + 1$	Füllgrad $\alpha > \lfloor \frac{1}{4} \rfloor$
Länge nach Halbierung:	$l(A') = \lfloor \frac{1}{2}\, l(A) \rfloor$	Inhalt nach Halbierung:	$\lfloor \frac{1}{4}\, l(A) \rfloor$	Füllgrad $\alpha = \lfloor \frac{1}{2} \rfloor$

Beschränktheit amortisierter Kosten:

$$\mathcal{C}(D_i) = \begin{cases} \frac{1}{2}\, l(A_i) - \text{Inhalt}(A_i), & \text{für } \alpha < \frac{1}{2} \\ 2 \times \text{Inhalt}(A_i) - l(A_i), & \text{für } \alpha \geq \frac{1}{2} \end{cases}$$

- Delete mit Umkopieren:

  $\alpha(A_{i-1}) < \frac{1}{2}, \alpha(A_i) \geq \frac{1}{2}$: $\hat{c}_i = 2$
- Insert mit Umkopieren:

  $\alpha(A_{i-1}) = 1, \alpha(A_i) \geq \frac{1}{2}$: $\hat{c}_i = 3$

Master-Theorem - Problem 2 **[Kap.** 0.4.5**]**

(Laufzeit rekursiver Funktionen)

Prüfen Sie die Anwendbarkeit der beiden Master-Theorem-Versionen auf die folgenden Rekurrenzgleichungen:

(a) $T(n) = 2\ T(\lfloor n/2 \rfloor) + 2\ T(\lceil n/2 \rceil) + n^2$

(b) $T(n) = 7\ T(n/7) + n\ \ln(n)$

Begründen Sie Ihre Antwort.

Master-Theorem - Problem 2 [Kap. 0.4.5]

(Laufzeit rekursiver Funktionen)

Prüfen Sie die Anwendbarkeit der beiden Master-Theorem-Versionen auf die folgenden Rekurrenzgleichungen:

(a) $\quad T(n) = 2\ T(\lfloor n/2 \rfloor) + 2\ T(\lceil n/2 \rceil) + n^2 \quad = \quad 4\ T(n/2) + n^2$

(b) $\quad T(n) = 7\ T(n/7) + n\ \ln(n)$

Begründen Sie Ihre Antwort.

Master-Theorem - Problem 2 [Kap. 0.4.5]

(Laufzeit rekursiver Funktionen)

Prüfen Sie die Anwendbarkeit der beiden Master-Theorem-Versionen auf die folgenden Rekurrenzgleichungen:

(a) $T(n) = 2\ T(\lfloor n/2 \rfloor) + 2\ T(\lceil n/2 \rceil) + n^2 \quad = \quad 4\ T(n/2) + n^2$

(b) $T(n) = 7\ T(n/7) + n\ \ln(n)$

Begründen Sie Ihre Antwort.

(a) Allgemeinere Form des Master-Theorems:

$$\begin{aligned} f(n) &= n^2 \in \Theta(n^k) \quad \text{mit } k = 2 \geq 0 \\ 0 < \alpha_i &= \frac{1}{2} < 1 \quad \text{für } i = 1, \ldots, 4 \\ \sum_{i=1}^{4} (\alpha_i)^k &= 4 \left(\frac{1}{2}\right)^2 = 4\,\frac{1}{4} = 1 \end{aligned}$$

Master-Theorem - Problem 2 [Kap. 0.4.5]

(Laufzeit rekursiver Funktionen)

Prüfen Sie die Anwendbarkeit der beiden Master-Theorem-Versionen auf die folgenden Rekurrenzgleichungen:

(a) $T(n) = 2\ T(\lfloor n/2 \rfloor) + 2\ T(\lceil n/2 \rceil) + n^2 \quad = \quad 4\ T(n/2) + n^2$

(b) $T(n) = 7\ T(n/7) + n\ \ln(n)$

Begründen Sie Ihre Antwort.

(a) Allgemeinere Form des Master-Theorems:

$$\begin{aligned} f(n) &= n^2 \in \Theta(n^k) \quad \text{mit } k = 2 \geq 0 \\ 0 < \alpha_i &= \frac{1}{2} < 1 \quad \text{für } i = 1, \ldots, 4 \\ \sum_{i=1}^{4} (\alpha_i)^k &= 4 \left(\frac{1}{2}\right)^2 = 4\,\frac{1}{4} = 1 \end{aligned}$$

$\Rightarrow$ Fall (ii) anwendbar: $T(n) \in \Theta(n^2 log(n))$

Master-Theorem - Problem 2 [Kap. 0.4.5]

(Laufzeit rekursiver Funktionen)

Prüfen Sie die Anwendbarkeit der beiden Master-Theorem-Versionen auf die folgenden Rekurrenzgleichungen:

(a) $T(n) = 2\ T(\lfloor n/2 \rfloor) + 2\ T(\lceil n/2 \rceil) + n^2 \quad = \quad 4\ T(n/2) + n^2$

(b) $T(n) = 7\ T(n/7) + n \ln(n)$

Begründen Sie Ihre Antwort.

(a) Master-Theorem:

$$\begin{aligned} a &= 4 \geq 1 \\ b &= 2 > 1 \\ log_2(4) &= 2 \\ f(n) &= n^2 \in \Theta(n^2) = \Theta(n^{log_2(4)}) \end{aligned}$$

Master-Theorem - Problem 2 [Kap. 0.4.5]

(Laufzeit rekursiver Funktionen)

Prüfen Sie die Anwendbarkeit der beiden Master-Theorem-Versionen auf die folgenden Rekurrenzgleichungen:

(a) $T(n) = 2\ T(\lfloor n/2 \rfloor) + 2\ T(\lceil n/2 \rceil) + n^2 \quad = \quad 4\ T(n/2) + n^2$

(b) $T(n) = 7\ T(n/7) + n\ \ln(n)$

Begründen Sie Ihre Antwort.

(a) Master-Theorem:

$$\begin{aligned} a &= 4 \geq 1 \\ b &= 2 > 1 \\ log_2(4) &= 2 \\ f(n) &= n^2 \in \Theta(n^2) = \Theta(n^{log_2(4)}) \end{aligned}$$

$\Rightarrow$ Fall (ii) anwendbar: $T(n) \in \Theta(n^2 log(n))$

Master-Theorem - Problem 2 [Kap. 0.4.5]

(Laufzeit rekursiver Funktionen)

Prüfen Sie die Anwendbarkeit der beiden Master-Theorem-Versionen auf die folgenden Rekurrenzgleichungen:

(a) $T(n) = 2\ T(\lfloor n/2 \rfloor) + 2\ T(\lceil n/2 \rceil) + n^2 \quad = \quad 4\ T(n/2) + n^2$

(b) $T(n) = 7\ T(n/7) + n \ln(n)$

Begründen Sie Ihre Antwort.

(b) Master-Theorem:

Master-Theorem - Problem 2 [Kap. 0.4.5]

(Laufzeit rekursiver Funktionen)

Prüfen Sie die Anwendbarkeit der beiden Master-Theorem-Versionen auf die folgenden Rekurrenzgleichungen:

(a) $T(n) = 2\ T(\lfloor n/2 \rfloor) + 2\ T(\lceil n/2 \rceil) + n^2 \quad = \quad 4\ T(n/2) + n^2$

(b) $T(n) = 7\ T(n/7) + n\ \ln(n)$

Begründen Sie Ihre Antwort.

(b) Master-Theorem:

$$\begin{aligned} a &= 7 \geq 1 \\ b &= 7 > 1 \\ log_7(7) &= 1 \end{aligned}$$

Master-Theorem - Problem 2 [Kap. 0.4.5]

(Laufzeit rekursiver Funktionen)

Prüfen Sie die Anwendbarkeit der beiden Master-Theorem-Versionen auf die folgenden Rekurrenzgleichungen:

(a) $T(n) = 2\ T(\lfloor n/2 \rfloor) + 2\ T(\lceil n/2 \rceil) + n^2 \quad = \quad 4\ T(n/2) + n^2$

(b) $T(n) = 7\ T(n/7) + n\ \ln(n)$

Begründen Sie Ihre Antwort.

(b) Master-Theorem:

$$\begin{aligned} a &= 7 \geq 1 \\ b &= 7 > 1 \\ log_7(7) &= 1 \\ f(n) &= n \ln(n) \notin \Theta(n) = \Theta(n^{log_7(7)}) \end{aligned} \qquad \text{(ii)}$$

Master-Theorem - Problem 2 [Kap. 0.4.5]

(Laufzeit rekursiver Funktionen)

Prüfen Sie die Anwendbarkeit der beiden Master-Theorem-Versionen auf die folgenden Rekurrenzgleichungen:

(a) $T(n) = 2\ T(\lfloor n/2 \rfloor) + 2\ T(\lceil n/2 \rceil) + n^2 \quad = \quad 4\ T(n/2) + n^2$

(b) $T(n) = 7\ T(n/7) + n\ \ln(n)$

Begründen Sie Ihre Antwort.

(b) Master-Theorem:

$$
\begin{aligned}
a &= 7 \geq 1 \\
b &= 7 > 1 \\
log_7(7) &= 1 \\
f(n) &= n \ln(n) \notin \Theta(n) = \Theta(n^{log_7(7)}) && \text{(ii)} \\
f(n) &= n \ln(n) \notin O(n^{1-\epsilon}) = O(n^{log_7(7)-\epsilon}) \quad \text{für } \epsilon > 0 && \text{(iii)}
\end{aligned}
$$

Master-Theorem - Problem 2 [Kap. 0.4.5]

(Laufzeit rekursiver Funktionen)

Prüfen Sie die Anwendbarkeit der beiden Master-Theorem-Versionen auf die folgenden Rekurrenzgleichungen:

(a) $T(n) = 2\ T(\lfloor n/2 \rfloor) + 2\ T(\lceil n/2 \rceil) + n^2 \quad = \quad 4\ T(n/2) + n^2$

(b) $T(n) = 7\ T(n/7) + n\ \ln(n)$

Begründen Sie Ihre Antwort.

(b) Master-Theorem:

$$
\begin{array}{rcll}
a & = & 7 \geq 1 & \\
b & = & 7 > 1 & \\
log_7(7) & = & 1 & \\
f(n) & = & n \ln(n) \notin \Theta(n) = \Theta(n^{log_7(7)}) & \text{(ii)} \\
f(n) & = & n \ln(n) \notin O(n^{1-\epsilon}) = O(n^{log_7(7)-\epsilon}) \quad \text{für } \epsilon > 0 & \text{(iii)} \\
f(n) & = & n \ln(n) \notin \Omega(n^{1+\epsilon}) = \Omega(n^{log_7(7)+\epsilon}) \quad \text{für } \epsilon > 0 & \text{(i)}
\end{array}
$$

Master-Theorem - Problem 2 [Kap. 0.4.5]

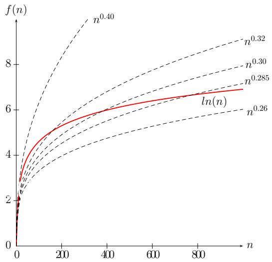
(Laufzeit rekursiver Funktionen)

Prüfen Sie die Anwendbarkeit der beiden Master-Theorem-Versionen auf die folgenden Rekurrenzgleichungen:

(a) $T(n) = 2\ T(\lfloor n/2 \rfloor) + 2\ T(\lceil n/2 \rceil) + n^2 \quad = \quad 4\ T(n/2) + n^2$

(b) $T(n) = 7\ T(n/7) + n\ \ln(n)$

Begründen Sie Ihre Antwort.



(b) Master-Theorem:

$$
\begin{aligned}
a &= 7 \geq 1 \\
b &= 7 > 1 \\
log_7(7) &= 1 \\
f(n) &= n \ln(n) \notin \Theta(n) = \Theta(n^{log_7(7)}) && \text{(ii)} \\
f(n) &= n \ln(n) \notin O(n^{1-\epsilon}) = O(n^{log_7(7)-\epsilon}) \quad \text{für } \epsilon > 0 && \text{(iii)} \\
f(n) &= n \ln(n) \notin \Omega(n^{1+\epsilon}) = \Omega(n^{log_7(7)+\epsilon}) \quad \text{für } \epsilon > 0 && \text{(i)}
\end{aligned}
$$

Master-Theorem - Problem 2 [Kap. 0.4.5]

(Laufzeit rekursiver Funktionen)

Prüfen Sie die Anwendbarkeit der beiden Master-Theorem-Versionen auf die folgenden Rekurrenzgleichungen:

(a) $T(n) = 2\ T(\lfloor n/2 \rfloor) + 2\ T(\lceil n/2 \rceil) + n^2 \quad = \quad 4\ T(n/2) + n^2$

(b) $T(n) = 7\ T(n/7) + n\ \ln(n)$

Begründen Sie Ihre Antwort.

(b) Master-Theorem:

$\Rightarrow$ Master-Theorem **nicht** anwendbar!

$$
\begin{array}{rcll}
f(n) & = & n \ln(n) \notin \Theta(n) = \Theta(n^{log_7(7)}) & \text{(ii)} \\
f(n) & = & n \ln(n) \notin O(n^{1-\epsilon}) = O(n^{log_7(7)-\epsilon}) \quad \text{für } \epsilon > 0 & \text{(iii)} \\
f(n) & = & n \ln(n) \notin \Omega(n^{1+\epsilon}) = \Omega(n^{log_7(7)+\epsilon}) \quad \text{für } \epsilon > 0 & \text{(i)}
\end{array}
$$

Master-Theorem - Problem 2 [Kap. 0.4.5]

(Laufzeit rekursiver Funktionen)

Prüfen Sie die Anwendbarkeit der beiden Master-Theorem-Versionen auf die folgenden Rekurrenzgleichungen:

(a) $T(n) = 2\ T(\lfloor n/2 \rfloor) + 2\ T(\lceil n/2 \rceil) + n^2 \quad = \quad 4\ T(n/2) + n^2$

(b) $T(n) = 7\ T(n/7) + n\ \ln(n)$

Begründen Sie Ihre Antwort.

(b) Allgemeinere Form des Master-Theorems:

$$f(n) \quad = \quad n \ln(n) \notin \Theta(n^k) \quad \forall k \geq 0$$

Master-Theorem - Problem 2 [Kap. 0.4.5]

(Laufzeit rekursiver Funktionen)

Prüfen Sie die Anwendbarkeit der beiden Master-Theorem-Versionen auf die folgenden Rekurrenzgleichungen:

(a) $T(n) = 2\ T(\lfloor n/2 \rfloor) + 2\ T(\lceil n/2 \rceil) + n^2 \quad = \quad 4\ T(n/2) + n^2$

(b) $T(n) = 7\ T(n/7) + n\ \ln(n)$

Begründen Sie Ihre Antwort.

(b) Allgemeinere Form des Master-Theorems:

$$f(n) \quad = \quad n \ln(n) \notin \Theta(n^k) \quad \forall k \geq 0$$

$\Rightarrow$ Allgemeinere Form des Master-Theorems **nicht** anwendbar!

Binäre Suche - Problem 3 [Kap. 0.4.5]

(Laufzeit rekursiver Funktionen)

Gegeben sei eine sortierte Sequenz $A := (a_1, \ldots, a_n)$ von n Elementen. Die Binäre Suche sucht nun nach einem Element x in A, indem sie x mit dem mittleren Element in A vergleicht und so entscheidet, ob und wenn ja in welcher Hälfte die Suche rekursiv fortgesetzt wird.
Modellieren Sie die Binäre Suche in **Pseudocode** und erstellen Sie die zugehörige **Rekurrenzgleichung**. Bestimmen Sie eine asymptotisch scharfe obere und eine asymptotisch scharfe untere **Schranke** der Laufzeit.

Binäre Suche - Problem 3 [**Kap.** 0.4.5]

(Laufzeit rekursiver Funktionen)

Gegeben sei eine sortierte Sequenz $A := (a_1, \ldots, a_n)$ von n Elementen. Die Binäre Suche sucht nun nach einem Element x in A, indem sie x mit dem mittleren Element in A vergleicht und so entscheidet, ob und wenn ja in welcher Hälfte die Suche rekursiv fortgesetzt wird.
Modellieren Sie die Binäre Suche in **Pseudocode** und erstellen Sie die zugehörige **Rekurrenzgleichung**. Bestimmen Sie eine asymptotisch scharfe obere und eine asymptotisch scharfe untere **Schranke** der Laufzeit.

Algorithmus 1 : $\textsc{BinärSuche}(A[1\ldots n], x)$

Ausgabe *Sequenzindex von x bzw.* -1 *falls x nicht enthalten ist*
wenn $n = 1$ UND $x \neq A[1]$ **dann**
> **return** -1

sonst
> **wenn** $x = A[\lfloor n/2 \rfloor]$ **dann**
>> **return** $\lfloor n/2 \rfloor$
>
> **sonst**
>> **wenn** $x < A[\lfloor n/2 \rfloor]$ **dann**
>>> $\textsc{BinärSuche}(A[1\ldots \lfloor n/2 \rfloor - 1], x)$
>>
>> **sonst**
>>> $\textsc{BinärSuche}(A[\lfloor n/2 \rfloor + 1\ldots n], x)$

Binäre Suche - Problem 3 [Kap. 0.4.5]

(Laufzeit rekursiver Funktionen)

Gegeben sei eine sortierte Sequenz $A := (a_1, \ldots, a_n)$ von n Elementen. Die Binäre Suche sucht nun nach einem Element x in A, indem sie x mit dem mittleren Element in A vergleicht und so entscheidet, ob und wenn ja in welcher Hälfte die Suche rekursiv fortgesetzt wird.
Modellieren Sie die Binäre Suche in **Pseudocode** und erstellen Sie die zugehörige **Rekurrenzgleichung**. Bestimmen Sie eine asymptotisch scharfe obere und eine asymptotisch scharfe untere **Schranke** der Laufzeit.

Rekurrenzgleichung: $T(n) = T(n/2) + f(n)$ mit $f(n)$ konstant

Binäre Suche - Problem 3 [Kap. 0.4.5]

(Laufzeit rekursiver Funktionen)

Gegeben sei eine sortierte Sequenz $A := (a_1, \ldots, a_n)$ von n Elementen. Die Binäre Suche sucht nun nach einem Element x in A, indem sie x mit dem mittleren Element in A vergleicht und so entscheidet, ob und wenn ja in welcher Hälfte die Suche rekursiv fortgesetzt wird.
Modellieren Sie die Binäre Suche in **Pseudocode** und erstellen Sie die zugehörige **Rekurrenzgleichung**. Bestimmen Sie eine asymptotisch scharfe obere und eine asymptotisch scharfe untere **Schranke** der Laufzeit.

Rekurrenzgleichung: $T(n) = T(n/2) + f(n)$ mit $f(n)$ konstant

Allgemeinere From des Mastertheorems:

$$\begin{aligned} f(n) &\in \Theta(1) = \Theta(n^k) \quad \text{mit } k = 0 \geq 0 \\ 0 < \alpha &= \frac{1}{2} < 1 \\ (\alpha_i)^k &= 1 \end{aligned}$$

Binäre Suche - Problem 3 [Kap. 0.4.5]

(Laufzeit rekursiver Funktionen)

Gegeben sei eine sortierte Sequenz $A := (a_1, \ldots, a_n)$ von n Elementen. Die Binäre Suche sucht nun nach einem Element x in A, indem sie x mit dem mittleren Element in A vergleicht und so entscheidet, ob und wenn ja in welcher Hälfte die Suche rekursiv fortgesetzt wird.
Modellieren Sie die Binäre Suche in **Pseudocode** und erstellen Sie die zugehörige **Rekurrenzgleichung**. Bestimmen Sie eine asymptotisch scharfe obere und eine asymptotisch scharfe untere **Schranke** der Laufzeit.

Rekurrenzgleichung: $T(n) = T(n/2) + f(n)$ mit $f(n)$ konstant

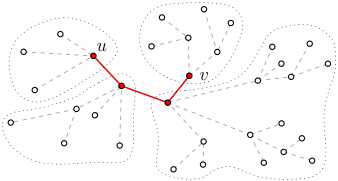
Allgemeinere From des Mastertheorems:

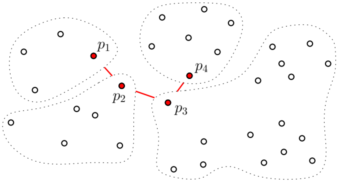
$$\begin{aligned} f(n) &\in \Theta(1) = \Theta(n^k) \quad \text{mit } k = 0 \geq 0 \\ 0 < \alpha &= \frac{1}{2} < 1 \\ (\alpha_i)^k &= 1 \end{aligned}$$

$\Rightarrow$ Fall (ii) anwendbar: $T(n) \in \Theta(n^0 log(n)) = \Theta(log(n))$

Pfad aus Superknoten - Problem 4 [1.1]

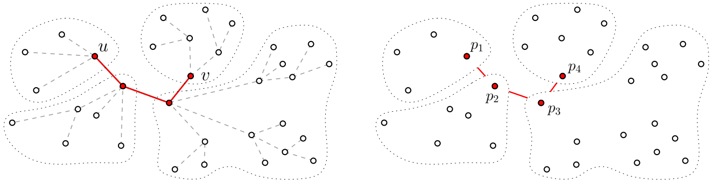
(Union-Find)





Pfad aus Superknoten - Problem 4 [1.1]

(Union-Find)

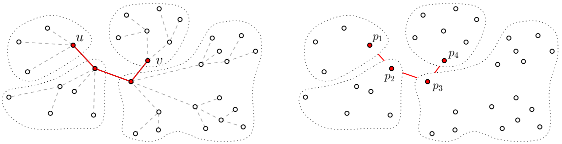


Idee: Verwende modifizierte Tiefensuche. Starte bei Knoten u.

- Durchsuche Unterbäume des aktuell betrachteten Knotens, ob v enthalten ist.
- Speichere alle besuchten Knoten in Pfadliste.
- Falls v in Unterbaum nicht gefunden, lösche Knoten des Unterbaums aus Pfadliste, kontrahiere Unterbaum (durch Vereinigung)

Pfad aus Superknoten

(Union-Find)



Bemerkung:
Pfad-Liste ist gobal
Union-Find-Struktur ist global,
speichert zu Beginn jeden Knoten des Baumes als einelementige Menge

---

Algorithmus 2 : GETPATHANDSUPERNODES(u, v, T)

Seiteneffekt *Pfad als Liste, Superknoten als Union-Find*
markiere u als besucht
hänge u an Pfad an
wenn $u = v$ **dann**
    Vereinige alle Knoten - fertig!
sonst
    DFS-PATH(u, v, T)   `// durchsuche UBs von u`

Pfad aus Superknoten

Algorithmus 3 : DFS-PATH(r, v, T)

```
Ausgabe GEFUNDEN als Boolsche Variable
GEFUNDEN ← FALSE                          // vor Abstieg in UBs von r
solange r noch unmarkierten Nachbarn s hat tue
    INUNTERBAUMGEFUNDEN ← FALSE           // vor Abst. in UBs von s
    markiere s
    hänge s an Pfad an
    wenn s = v dann
        GEFUNDEN ← TRUE                   // weitersuchen für Kontaktion
    INUNTERBAUMGEFUNDEN = DFS-PATH(s, v, T)   // Abstieg von s
    wenn INUNTERBAUMGEFUNDEN dann
        GEFUNDEN ← TRUE                   // UBs von s sind UBs von r
    wenn nicht INUNTERBAUMGEFUNDEN und nicht GEFUNDEN dann
        Lösche letztes Element aus Pfad   // Aufstieg von s nach r
        UNION(FIND(r), FIND(s))
return GEFUNDEN
```

Heapvarianten - Problem 5 [Kap. 1.3]

(Heap-Datenstruktur)

- Die klassische Heap-Eigenschaft $A[\mathit{Vorgänger}[i]] \geq A[i]$ kann auch als Max-Heap-Eigenschaft bezeichnet werden. Der klassische Heap wird somit zu einem Max-Heap. Wie sieht konsequenterweise die **Heap-Eigenschaft eines Min-Heaps** aus?
- Sei A ein Max-Heap (gespeichert in einem Array) und A' enthalte die Elemente aus A in **umgekehrter Reihenfolge**. Ist A' ein Min-Heap? Begründen Sie Ihre Antwort.

Heap-Varianten - Problem 5 **[Kap.** 1.3**]**

(Heap-Datenstruktur)

- Die klassische Heap-Eigenschaft $A[\mathit{Vorgänger}[i]] \geq A[i]$ kann auch als Max-Heap-Eigenschaft bezeichnet werden. Der klassische Heap wird somit zu einem Max-Heap. Wie sieht konsequenterweise die **Heap-Eigenschaft eines Min-Heaps** aus?
- Sei A ein Max-Heap (gespeichert in einem Array) und A' enthalte die Elemente aus A in **umgekehrter Reihenfolge**. Ist A' ein Min-Heap? Begründen Sie Ihre Antwort.

Min-Heap-Eigenschaft: $A[\mathit{Vorgänger}[i]] \leq A[i]$

Heapvarianten - Problem 5 [Kap. 1.3]

(Heap-Datenstruktur)

- Die klassische Heap-Eigenschaft $A[\mathit{Vorgänger}[i]] \geq A[i]$ kann auch als Max-Heap-Eigenschaft bezeichnet werden. Der klassische Heap wird somit zu einem Max-Heap. Wie sieht konsequenterweise die **Heap-Eigenschaft eines Min-Heaps** aus?
- Sei A ein Max-Heap (gespeichert in einem Array) und A' enthalte die Elemente aus A in **umgekehrter Reihenfolge**. Ist A' ein Min-Heap? Begründen Sie Ihre Antwort.

Min-Heap-Eigenschaft: $A[\mathit{Vorgänger}[i]] \leq A[i]$

Min-Heap durch Umkehrung?

Antwort:

Heap-Varianten - Problem 5 [Kap. 1.3]

(Heap-Datenstruktur)

- Die klassische Heap-Eigenschaft $A[\mathit{Vorgänger}[i]] \geq A[i]$ kann auch als Max-Heap-Eigenschaft bezeichnet werden. Der klassische Heap wird somit zu einem Max-Heap. Wie sieht konsequenterweise die **Heap-Eigenschaft eines Min-Heaps** aus?
- Sei A ein Max-Heap (gespeichert in einem Array) und A' enthalte die Elemente aus A in **umgekehrter Reihenfolge**. Ist A' ein Min-Heap? Begründen Sie Ihre Antwort.

Min-Heap-Eigenschaft: $A[\mathit{Vorgänger}[i]] \leq A[i]$
Min-Heap durch Umkehrung?
Antwort: Nein! (Gegenbeispiel)

Heap-Varianten - Problem 5 [Kap. 1.3]

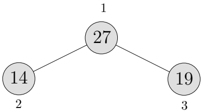
(Heap-Datenstruktur)

- Die klassische Heap-Eigenschaft $A[\mathit{Vorgänger}[i]] \geq A[i]$ kann auch als Max-Heap-Eigenschaft bezeichnet werden. Der klassische Heap wird somit zu einem Max-Heap. Wie sieht konsequenterweise die **Heap-Eigenschaft eines Min-Heaps** aus?
- Sei A ein Max-Heap (gespeichert in einem Array) und A' enthalte die Elemente aus A in **umgekehrter Reihenfolge**. Ist A' ein Min-Heap? Begründen Sie Ihre Antwort.

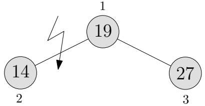
Min-Heap-Eigenschaft: $A[\mathit{Vorgänger}[i]] \leq A[i]$

Min-Heap durch Umkehrung?

Antwort: Nein! (Gegenbeispiel)



27	14	19
1	2	3



19	14	27
1	2	3

Heap-Varianten - Problem 5 [Kap. 1.3]

(Heap-Datenstruktur)

Ergänzen Sie an untenstehendem Heap A (Max-Heap in Baumform) die zugehörigen **Array-Indizes**. Wenden Sie DELETE (A, 2) an, indem Sie die einzelnen Zustände, die der Baum durchläuft, darstellen.

Heapvarianten - Problem 5 [Kap. 1.3]

(Heap-Datenstruktur)

Ergänzen Sie an untenstehendem Heap A (Max-Heap in Baumform) die zugehörigen **Array-Indizes**. Wenden Sie DELETE (A, 2) an, indem Sie die einzelnen Zustände, die der Baum durchläuft, darstellen.

Achtung:
Indizierung beginnt mit 1, **nicht** mit 0!!!
DELETE (A, i) hat als Parameter i den Index, **nicht** den Wert!!!

Heap-Varianten - Problem 5 [Kap. 1.3]

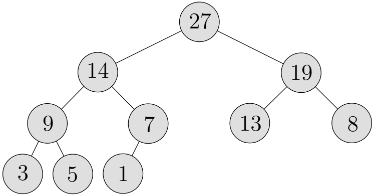
(Heap-Datenstruktur)

Ergänzen Sie an untenstehendem Heap *A* (Max-Heap in Baumform) die zugehörigen **Array-Indizes**. Wenden Sie DELETE (*A*, 2) an, indem Sie die einzelnen Zustände, die der Baum durchläuft, darstellen.

Achtung:

Indizierung beginnt mit 1, **nicht** mit 0!!!

DELETE (A, i) hat als Parameter *i* den Index, **nicht** den Wert!!!



Heap-Varianten - Problem 5 [**Kap.** 1.3]

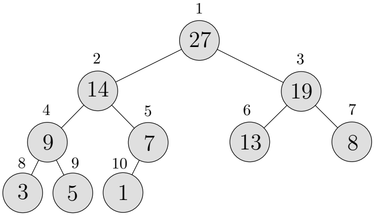
(Heap-Datenstruktur)

Ergänzen Sie an untenstehendem Heap *A* (Max-Heap in Baumform) die zugehörigen **Array-Indizes**. Wenden Sie DELETE (*A*, 2) an, indem Sie die einzelnen Zustände, die der Baum durchläuft, darstellen.

Achtung:

Indizierung beginnt mit 1, **nicht** mit 0!!!

DELETE (*A*, *i*) hat als Parameter *i* den Index, **nicht** den Wert!!!



Heap-Varianten - Problem 5 [Kap. 1.3]

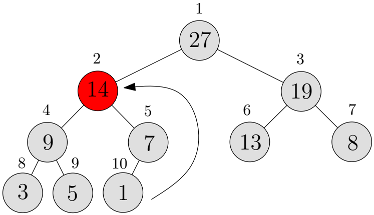
(Heap-Datenstruktur)

Ergänzen Sie an untenstehendem Heap A (Max-Heap in Baumform) die zugehörigen **Array-Indizes**. Wenden Sie DELETE (A, 2) an, indem Sie die einzelnen Zustände, die der Baum durchläuft, darstellen.

Achtung:

Indizierung beginnt mit 1, **nicht** mit 0!!!

DELETE (A, i) hat als Parameter i den Index, **nicht** den Wert!!!



Heap-Varianten - Problem 5 [Kap. 1.3]

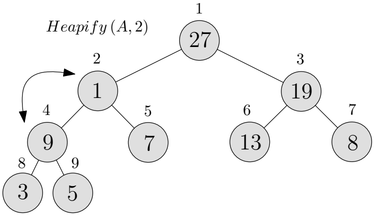
(Heap-Datenstruktur)

Ergänzen Sie an untenstehendem Heap A (Max-Heap in Baumform) die zugehörigen **Array-Indizes**. Wenden Sie DELETE (A, 2) an, indem Sie die einzelnen Zustände, die der Baum durchläuft, darstellen.

Achtung:

Indizierung beginnt mit 1, **nicht** mit 0!!!

DELETE (A, i) hat als Parameter i den Index, **nicht** den Wert!!!



Heap-Varianten - Problem 5 [Kap. 1.3]

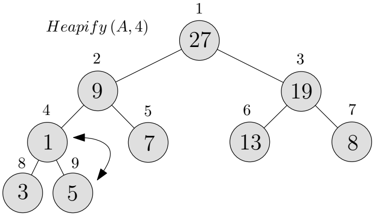
(Heap-Datenstruktur)

Ergänzen Sie an untenstehendem Heap *A* (Max-Heap in Baumform) die zugehörigen **Array-Indizes**. Wenden Sie DELETE (*A*, 2) an, indem Sie die einzelnen Zustände, die der Baum durchläuft, darstellen.

Achtung:

Indizierung beginnt mit 1, **nicht** mit 0!!!

DELETE (*A*, *i*) hat als Parameter *i* den Index, **nicht** den Wert!!!



Heap-Varianten - Problem 5 [**Kap.** 1.3]

(Heap-Datenstruktur)

Ergänzen Sie an untenstehendem Heap *A* (Max-Heap in Baumform) die zugehörigen **Array-Indizes**. Wenden Sie DELETE (*A*, 2) an, indem Sie die einzelnen Zustände, die der Baum durchläuft, darstellen.

Achtung:

Indizierung beginnt mit 1, **nicht** mit 0!!!

DELETE (A, i) hat als Parameter i den Index, **nicht** den Wert!!!