



# Amortisierte Analyse

Prof. Dr. S. Albers  
Prof. Dr. Th. Ottmann

# Amortisierung

---

- Betrachte eine Folge  $a_1, a_2, \dots, a_n$  von  $n$  Operation auf Datenstruktur  $D$
- $T_i$  = Ausführungszeit von  $a_i$
- $T = T_1 + T_2 + \dots + T_n$ , Gesamtlaufzeit
- Oft kann die Laufzeit einer einzelnen Operation in einem großen Bereich schwanken, z. B. in  
 $1, \dots, n$ ,  
aber nicht bei allen Operationen der Folge kann der schlechteste Fall auftreten

# Analyse von Algorithmen

---

- Best Case
- Worst Case
- Average Case
- Amortisierte Worst Case

Was sind die **durchschnittlichen** Kosten einer **schlechtest möglichen** Folge von Operationen?

# Amortisierung

---

## Idee:

- Zahle für billige Operation etwas mehr
- Verwende Ersparnes um für teure Operationen zu zahlen

## Drei Methoden:

1. Aggregatmethode
2. Bankkonto – Methode
3. Potentialfunktion – Methode

# 1. Aggregat – Methode: Dualzähler

## Bestimmung der Bitwechselkosten eines Dualzählers

Operation	Zählerstand	Kosten
0	00000	1
1	00001	2
2	00010	1
3	00011	3
4	00100	1
5	00101	2
6	00110	
7	00111	
8	01000	
9	01001	
10	01010	
11	01011	
12	01100	
13	01101	

## 2. Bankkonto – Methode

---

### Idee:

Bezahle **zwei** KE für das Verwandeln einer **0** in eine **1**

→ jede 1 hat eine KE auf dem Konto

### Beobachtung:

In jedem Schritt wird genau eine 0 in eine 1 verwandelt

# Bankkonto – Methode



Operation	Zählerstand
0	0 0 0 0 0
1	0 0 0 0 1
2	0 0 0 1 0
3	0 0 0 1 1
4	0 0 1 0 0
5	0 0 1 0 1
6	0 0 1 1 0
7	0 0 1 1 1
8	0 1 0 0 0
9	0 1 0 0 1
10	0 1 0 1 0

# 3. Potentialfunktion

---

## Potentialfunktion $\phi$

Datenstruktur  $D \rightarrow \phi(D)$

$t_l$  = wirkliche Kosten der  $l$ -ten Operation

$\phi_l$  = Potential nach Ausführung der  $l$ -ten Operation (=  $\phi(D_l)$  )

$a_l$  = amortisierte Kosten der  $l$ -ten Operation

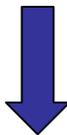
### Definition:

$$a_l = t_l + \phi_l - \phi_{l-1}$$

# Beispiel: Dualzähler

$D_i$  = Stand der  $i$ -ten Operation

$\phi_i = \phi(D_i) = \#$  von Einsen in  $D_i$

$i$ -te Operation	# von Einsen
$D_{i-1}$ : .....0/1.....01.....1 	$B_{i-1}$
$D_i$ : .....0/1.....10.....0	$B_i = B_{i-1} - b_i + 1$

$t_i$  = **wirkliche Bitwechselkosten von Operation  $i$**   
 $= b_i + 1$

# Dualzähler

---

$t_i$  = wirkliche Bitwechselkosten von Operation  $i$

$a_i$  = amortisierte Bitwechselkosten von Operation  $i$

$$a_i = (b_i + 1) + (B_{i-1} - b_i + 1) - B_{i-1}$$

$$= 2$$

$$\Rightarrow \sum t_i \leq 2n$$

# Dynamische Tabellen

---

## Problem:

Verwaltung einer Tabelle unter den Operationen **Einfügen** und **Entfernen**, so dass

- die Tabellengröße der Anzahl der Elemente angepasst werden kann
- immer ein konstanter Anteil der Tabelle mit Elementen belegt ist
- die Kosten für  $n$  Einfüge- oder Entferne-Operationen  $O(n)$  sind.

Organisation der Tabelle: Hashtabelle, Heap, Stack, etc.

**Belegungsfaktor**  $\alpha_T$ : Anteil der Tabellenplätze von  $T$ , die belegt sind.

# Implementation Einfügen

---

```
class dynamic table {  
  
    int [] table;  
  
    int size;           // Größe der Tabelle  
    int num;           // Anz. der Elemente  
  
    dynamicTable() {   // Initialisierung der leeren Tabelle  
        table = new int [1];  
        size = 1;  
        num = 0;  
    }  
}
```

# Implementen Einfügen

---

```
insert ( int x) {  
    if (num == size ) {  
        new Table = new int [2*size];  
        for (i = 0; i < size; i++)  
            füge table[i] in newTable ein;  
        table = newTable;  
        size = 2*size;  
    }  
    füge x in table ein;  
    num = num + 1;  
}
```

# Kosten von $n$ Einfüge-Operationen in eine anfangs leere Tabelle



$t_i$  = Kosten der  $i$ -ten Einfüge-Operation

## Worst case:

$t_i = 1$ , falls die Tabelle vor der Operation  $i$  nicht voll ist

$t_i = (i - 1) + 1$ , falls die Tabelle vor der Operation  $i$  voll ist.

Also verursachen  $n$  Einfüge-Operationen höchstens Gesamtkosten von

$$\sum_{i=1}^n (i) = O(n^2)$$

## Amortisierter Worst-Case:

Aggregat -, Bankkonto -, Potential-Methode

# Potential-Methode

---

$T$  Tabelle mit

- $k = T.num$  Elemente und
- $s = T.size$  Größe

## Potentialfunktion

$$\phi(T) = 2k - s$$

# Potential-Methode

## Eigenschaften

- $\phi_0 = \phi(T_0) = \phi(\text{leere Tabelle}) = -1$
- Für alle  $i \geq 1 : \phi_i = \phi(T_i) \geq 0$   
Weil  $\phi_n - \phi_0 \geq 0$  gilt, ist  
$$\sum a_i \text{ obere Schranke für } \sum t_i$$
- Unmittelbar vor einer Tabellenexpansion ist  $k = s$ ,  
also  $\phi(T) = k = s$ .
- Unmittelbar nach einer Tabellenexpansion ist  $k = s/2$ ,  
also  $\phi(T) = 2k - s = 0$ .

# Berechnung der amortisierten Kosten $a_i$ der $i$ -ten Einfüge-Operation



$k_i$  = # Elemente in T nach der  $i$ -ten Operation

$s_i$  = Tabellengröße von T nach der  $i$ -ten Operation

**Fall 1:**  $i$ -te Operation löst keine Expansion aus

$$k_i = k_{i-1} + 1, s_i = s_{i-1}$$

$$\begin{aligned} a_i &= 1 + (2k_i - s_i) - (2k_{i-1} - s_{i-1}) \\ &= 1 + 2(k_i - k_{i-1}) \\ &= 3 \end{aligned}$$

---

*Fall 2:*  $i$ -te Operation löst Expansion aus

$$k_i = k_{i-1} + 1, s_i = 2s_{i-1}$$

$$a_i = k_{i-1} + 1 + (2k_i - s_i) - (2k_{i-1} - s_{i-1})$$

$$= 3$$

# Einfügen und Entfernen von Elementen

---

**Jetzt:** Kontrahiere Tabelle, wenn Belegung zu gering!

## Ziele:

- (1) Belegungsfaktor bleibt durch eine Konstante nach unten beschränkt
- (2) amortisierte Kosten einer einzelnen Einfüge- oder Entferne-Operation sind konstant.

## 1. Versuch

- Expansion: wie vorher
- Kontraktion: Halbiere Tabellengröße, sobald Tabelle weniger als  $\frac{1}{2}$  voll ist!

# „Schlechte“ Folge von Einfüge- und Entfernenoperationen



Kosten

$n/2$  mal Einfügen  
(Tabelle voll)



$3 n/2$

$I$ : Expansion



$n/2 + 1$

$D, D$ : Kontraktion



$n/2 + 1$

$I, I$ : Expansion



$n/2 + 1$

$D, D$ : Kontraktion



Gesamtkosten der Operationsfolge:  
 $I_{n/2}, I, D, D, I, I, D, D, \dots$  der Länge  $n$  sind

## 2. Versuch

---

**Expansion:** Verdoppele die Tabellengröße, wenn in die volle Tabelle eingefügt wird.

**Kontraktion:** Sobald der Belegungsfaktor unter  $\frac{1}{4}$  sinkt , halbiere die Tabellengröße.

**Folgerung:**

Die Tabelle ist stets wenigstens zu  $\frac{1}{4}$  voll, d.h.

$$\frac{1}{4} \leq \alpha(T) \leq 1$$

Kosten einer Folge von Einfüge- und Entferne-Operationen?

# Analyse Einfügen und Entfernen

---

$k = T.num, \quad s = T.size, \quad \alpha = k/s$

**Potentialfunktion  $\phi$**

$$\phi(T) = \begin{cases} 2k - s, & \text{falls } \alpha \geq 1/2 \\ s/2 - k, & \text{falls } \alpha < 1/2 \end{cases}$$

# Analyse Einfügen und Entfernen

$$\phi(T) = \begin{cases} 2k - s, & \text{falls } \alpha \geq 1/2 \\ s/2 - k, & \text{falls } \alpha < 1/2 \end{cases}$$

Unmittelbar nach einer Expansion oder Kontraktion der Tabelle:

$$s = 2k, \text{ also } \phi(T) = 0$$

# Einfügen

---

*i*-te Operation:  $k_i = k_{i-1} + 1$

Fall 1:  $\alpha_{i-1} \geq \frac{1}{2}$

Fall 2:  $\alpha_{i-1} < \frac{1}{2}$

Fall 2.1:  $\alpha_i < \frac{1}{2}$

Fall 2.2:  $\alpha_i \geq \frac{1}{2}$

# Einfügen

Fall 2.1:  $\alpha_{i-1} < 1/2$ ,  $\alpha_i < 1/2$  (keine Expansion)

**Potentialfunktion  $\phi$**

$$\phi(T) = \begin{cases} 2k - s, & \text{falls } \alpha \geq 1/2 \\ s/2 - k, & \text{falls } \alpha < 1/2 \end{cases}$$

# Einfügen

Fall 2.2:  $\alpha_{j-1} < 1/2$ ,  $\alpha_j \geq 1/2$  (keine Expansion)

**Potentialfunktion  $\phi$**

$$\phi(T) = \begin{cases} 2k - s, & \text{falls } \alpha \geq 1/2 \\ s/2 - k, & \text{falls } \alpha < 1/2 \end{cases}$$

# Entfernen

$$k_i = k_{i-1} - 1$$

Fall 1:  $\alpha_{i-1} < 1/2$

Fall1.1: Entfernen verursacht keine Kontraktion

$$s_i = s_{i-1}$$

**Potentialfunktion  $\phi$**

$$\phi(T) = \begin{cases} 2k - s, & \text{falls } \alpha \geq 1/2 \\ s/2 - k, & \text{falls } \alpha < 1/2 \end{cases}$$

# Entfernen

$$k_i = k_{i-1} - 1$$

Fall 1:  $\alpha_{i-1} < 1/2$

Fall 1.2:  $\alpha_{i-1} < 1/2$  Entfernen verursacht Kontraktion

$$2s_i = s_{i-1} \quad k_{i-1} = s_{i-1}/4$$

**Potentialfunktion  $\phi$**

$$\phi(T) = \begin{cases} 2k - s, & \text{falls } \alpha \geq 1/2 \\ s/2 - k, & \text{falls } \alpha < 1/2 \end{cases}$$

# Entfernen

Fall 2:  $\alpha_{i-1} \geq 1/2$  keine Kontraktion

$$s_i = s_{i-1} \quad k_i = k_{i-1} - 1$$

Fall2.1:  $\alpha_{i-1} \geq 1/2$

**Potentialfunktion  $\phi$**

$$\phi(T) = \begin{cases} 2k - s, & \text{falls } \alpha \geq 1/2 \\ s/2 - k, & \text{falls } \alpha < 1/2 \end{cases}$$

# Entfernen

Fall 2:  $\alpha_{i-1} \geq 1/2$  keine Kontraktion

$$s_i = s_{i-1} \quad k_i = k_{i-1} - 1$$

Fall2.2:  $\alpha_i < 1/2$

**Potentialfunktion  $\phi$**

$$\phi(T) = \begin{cases} 2k - s, & \text{falls } \alpha \geq 1/2 \\ s/2 - k, & \text{falls } \alpha < 1/2 \end{cases}$$