

Kap. 2: Amortisierte Analyse von Algorithmen



Professor Dr. Petra Mutzel
Lehrstuhl für Algorithm Engineering, LS11
Fakultät für Informatik, TU Dortmund

6. VO A&D WS 08/09 30. Oktober 2008

Literatur für diese VO

T.H. Cormen, C.E. Leiserson, R. Rivest, C. Stein:
Algorithmen – Eine Einführung, Oldenbourg, 2004 (oder
das englische Original, MIT Press)

Überblick

2.1 Einführung

- Beispiel: Stapeloperationen
- Beispiel: Binärzähler

2.2 Aggregat-Analyse

2.3 Account-Methode (Bankkonto-Methode)

2.4 Potenzialmethode

2.1 Einführung

- Bei einer **amortisierten Analyse** wird die mittlere Laufzeit pro Operation über eine Sequenz von N Datenstruktur-Operationen im worst case ermittelt.
- Sie kann beweisen, dass die Kosten einer einzelnen Operation klein sind, wenn man über eine Sequenz von Operationen mittelt, auch wenn eine einzelne Operation in der Sequenz kostspielig sein kann.
- Sie **garantiert** die mittlere Performanz jeder Operation im **schlechtesten Fall**.

Beispiel 1: Stapeloperationen

- Unser Stapel besitzt die folgenden Operationen:
- **PUSH** (s, x) : legt Objekt x auf den Stapel S
- **POP** (s) : entnimmt das oberste Element von S und gibt es aus
- **MULTIPOP** (s, k) : entnimmt die k obersten Elemente von S bzw. falls S weniger als k Elemente besitzt, dann entnimmt es alle Elemente aus S , und gibt die Elemente aus.
- **STACK-EMPTY** (s) : falls S leer ist, dann Ausgabe von 1, sonst 0.

Beispiel 1: Stapeloperationen

MULTIPOP (S, k)

- (1) while not STACK-EMPTY(S) und $k \neq 0$
- (2) do { POP (S) ; $k=k-1$ }

Laufzeit auf Sequenz von N Operationen?

- im worst case: ein Aufruf von **MULTIPOP** (S, k) kann $\Theta(N)$ dauern (falls $k=\Theta(N)$)
- es kann bis zu N Aufrufe geben
- \rightarrow insgesamt: $O(N^2)$

amortisierte Analyse liefert eine bessere Abschätzung

Beispiel 2: Binärzähler

- Inkrementieren eines k-Bit-Binärzählers, der von 0 an aufwärts zählt:
- Feld $A[0..k-1]$ mit $\text{länge}[A]=k$
- Binärzahl x hat kleinstes Bit in $A[0]$: $x = \sum A[i] 2^i$

INCREMENT (A)

- $i=0$
- **while** $i < \text{länge}[A]$ und $A[i]=1$
- **do** { $A[i]=0$; $i=i+1$ }
- **if** $i < \text{länge}[A]$ then $A[i]=1$

Beispiel 2: Binärzähler

INCREMENT (A)

- $i=0$
- **while** $i < \text{länge}[A]$ und $A[i]=1$
- **do** { $A[i]=0$; $i=i+1$ }
- **if** $i < \text{länge}[A]$ then $A[i]=1$

Laufzeit auf Sequenz von N Operationen?

- im worst case: ein Aufruf von **INCREMENT (A)** kann $\Theta(k)$ dauern
- bei N Aufrufen: \rightarrow insgesamt: $O(kN)$

amortisierte Analyse liefert eine bessere Abschätzung

2.2 Aggregat-Analyse

Methode:

- Es wird gezeigt, dass eine Sequenz von N (beliebigen) Operationen für alle N im schlechtesten Fall insgesamt Zeit $T(N)$ benötigt.
- Deswegen betragen die mittleren oder amortisierten Kosten pro Operation $T(N)/N$.

Bemerkungen:

- Diese amortisierten Kosten gelten für jede Operation, auch wenn es verschiedene Operationstypen in der Sequenz gibt
- Dies ist der wesentliche Unterschied zu den anderen beiden Methoden (s. 2.3 und 2.4)

Aggregat-Analyse für Beispiel 1

Analyse der Sequenz von N Operationen:

- Eine beliebige Folge von N **PUSH**, **POP** und **MULTIPOP** Operationen auf einem anfangs leeren Stapel kann höchstens $O(N)$ kosten.
 - Denn: jedes Objekt kann höchstens so oft entnommen werden, wie es hinzugefügt wurde
- Die Gesamtzeit einer Sequenz ist also $O(N)$.
- Die mittleren Kosten einer Operation sind also $O(N)/N=O(1)$
- Die amortisierten Kosten aller drei Stapeloperationen sind somit $O(1)$.

Aggregat-Analyse für Beispiel 2

Analyse der Sequenz von N Operationen:

- $A[0]$ klappt bei jedem Aufruf um.
- $A[1]$ klappt bei jedem 2.-ten Aufruf um.
- $A[2]$ klappt bei jedem 4.-ten (2^2) Aufruf um: $N/4$ Mal
- $A[i]$ klappt bei jedem 2^i -ten Aufruf um: $N/2^i$ Mal
- Gesamtzahl für das Umklappen in einer Sequenz:

$$\sum_{i=0}^{k-1} \left\lfloor \frac{N}{2^i} \right\rfloor < N \sum_{i=0}^{\infty} \frac{1}{2^i} = 2N$$
- Laufzeit für eine Sequenz von N Operationen ist deshalb im worst case $O(N)$.
- Die amortisierten Kosten pro Operation sind $O(N)/N=O(1)$

2.3 Account-Methode (Bankkonto)

Methode:

- Verschiedenen Operationen werden verschiedene Kosten zugeordnet, die evtl. auch größer oder kleiner als die tatsächlichen Kosten sein können.
- Wenn die amortisierten Kosten die tatsächlichen Kosten übersteigen, dann wird die Differenz speziellen Objekten in der Datenstruktur als **Kredit (Guthaben)** zugewiesen, der später verwendet werden kann.
- Dieser dient für Operationen, deren tatsächliche Kosten höher sind als deren amortisierte Kosten.
- Der Gesamtkredit darf niemals negativ sein.
- Es muss dabei zu jedem Zeitpunkt und für jede Sequenz gelten: die Summe der tatsächlichen Kosten \leq Summe der amortisierten Kosten.

Account-Methode für Beispiel 1

Zuordnung der Kosten:

Operation	tatsächliche Kosten	amortisierte Kosten
PUSH (S, x)	1	2
POP (S)	1	0
MULTIPOP (S, k)	min(k,s)	0

Zu zeigen: jede Operation in der Sequenz ist bezahlbar:

- PUSH (S, x) bezahlt die Operation mit 1 Einheit und legt 1 Einheit auf den Stapel als Guthaben.
- Nach p PUSH-Ops. liegen p Einheiten auf dem Stapel.
- Jede POP (S) oder MULTIPOP (S, k) Operation verbraucht jeweils 1 bzw. k Einheiten aus dem Guthaben. Aber diese sind vorhanden.

Account-Methode für Beispiel 2

Zuordnung der Kosten:

- Das Setzen eines Bits auf 1 erhält amortisierte Kosten 2. Das Rücksetzen des Bits auf 0 kostet 0 amort. Einheiten.
- Die tatsächlichen Kosten des Setzens oder Rücksetzens eines Bits sind dagegen jeweils 1.
- Wenn ein Bit gesetzt wurde, verwenden wir 1 Einheit für die Bezahlung und 1 bleibt als Guthaben bei diesem Bit.

Zu zeigen: jede Operation in der Sequenz ist bezahlbar:

- Bei INCREMENT werden alle Kosten für das Zurücksetzen durch die Guthaben auf den 1-Bits bezahlt (jedes 1-Bit besitzt ein solches Guthaben)
- Es wird höchstens ein Bit auf 1 gesetzt, d.h. die amortisierten Kosten einer INCREMENT Operation sind 2.
- Amortisierte Gesamtkosten aller N Operationen: O(N)

2.4 Potenzial-Methode

Methode:

- Verschiedenen Operationen werden verschiedene Kosten zugeordnet, die evtl. auch größer oder kleiner als die tatsächlichen Kosten sein können.
- Hier wird jedoch die Differenz zwischen amortisierten und tatsächlichen Kosten nicht speziellen **Elementen** zugewiesen, sondern der **Datenstruktur als Ganzes** in Form eines **Potenzials** zur Verfügung gestellt.
- Dieses dient für Operationen, deren tatsächliche Kosten höher sind als deren amortisierte Kosten.
- Die Potentialfunktion bildet jede Datenstruktur auf eine reelle Zahl ab, die dem Potenzial entspricht.
- Normalerweise ist die Potenzialfunktion zu Beginn gleich 0 und danach immer nicht-negativ.

Amortisierte Analyse via Potenzialmethode

- Dazu Potenzial mathematisch beschreiben:
 - Funktion Φ bildet jede Datenstruktur D_i auf eine reelle Zahl Φ_i ab, die dem Potenzial entspricht
- **Amortisierte Kosten** a_i der i-ten Operation sind definiert als Summe der tatsächliche Kosten t_i und des Zuwachses im Potenzial aufgrund der Operation:

$$a_i = t_i + \Phi_i - \Phi_{i-1}$$

Sei o_1, \dots, o_n eine Folge von n Operationen

- a_i amortisierte Kosten für Operation o_i
- t_i tatsächliche Kosten für Operation o_i
- Φ_i Potenzial direkt nach Operation o_i
- Φ_0 Potenzial vor o_1

Amortisierte Analyse: Vorgehen

Dann ist die Summe der amortisierten Kosten

$$\sum_{i=1}^n a_i = \sum_{i=1}^n (t_i + \Phi_i - \Phi_{i-1}) = \Phi_n - \Phi_0 + \sum_{i=1}^n t_i$$

- Wir wählen eine Potenzialfunktion, die
 - nichtnegativ und
 - am Anfang gleich Null ist
- Dann ist $\Phi_n - \Phi_0 \geq 0$ und damit $\sum t_i \leq \sum a_i$, also die tatsächlichen Kosten immer kleiner gleich den amortisierten Kosten.

Potenzial-Methode für Beispiel 1

Zuordnung der Potenzialfunktion:

- Potenzialfunktion auf einem Stapel: **Anzahl der Objekte im Stapel**; zu Beginn: $\Phi_0=0$

Amortisierte Kosten für PUSH (S, x) :

- Potenzialdifferenz: $\Phi_i - \Phi_{i-1} = (s+1) - s = 1$
- Amortisierte Kosten: $a_i = t_i + \Phi_i - \Phi_{i-1} = 1 + 1 = 2$

Amortisierte Kosten für MULTIPOP (S, k) :

- Annahme: es werden k' Elemente entfernt
- Potenzialdifferenz: $\Phi_i - \Phi_{i-1} = k'$
- Amortisierte Kosten: $a_i = t_i + \Phi_i - \Phi_{i-1} = k' - k' = 0$

Die amortisierten Kosten aller Operationen sind also in O(1)

Potenzial-Methode für Beispiel 2

Zuordnung der Kosten:

- Wir setzen das Potenzial nach der i -ten INCREMENT-Operation als $b_i = \text{Anzahl der Einsen}$ im Binärzähler nach der i -ten Operation.
- Die i -te INCREMENT Operation setzt z_i Bits zurück (auf 0)
- Die tatsächlichen Kosten sind dann höchstens z_i+1 , da zusätzlich höchstens ein Bit auf 1 gesetzt wird.
- Im Fall $b_i=0$ setzt i -te Operation alle k Bits zurück und es gilt $b_{i-1}=z_i=k$. Falls $b_i>0$, dann gilt $b_i=b_{i-1}-z_i+1$.
- In jedem Fall gilt: $\Phi_i - \Phi_{i-1} \leq (b_{i-1} - z_i + 1) - b_{i-1} = 1 - z_i$
- Amortisierte Kosten: $a_i = t_i + \Phi_i - \Phi_{i-1} \leq (z_i+1) + (1-z_i) = 2$ und es gilt immer $\Phi_i \geq 0$ und $\Phi_0 = 0$

Gesamtkosten von insgesamt N Operationen: $O(N)$

ENDE