Amortisierte Laufzeitanalyse

Besart Sylejmani, 0720032

Nouzad Mohammad, 0820679

Überblick

- 1 Einführung
  - Beispiel: Stapeloperationen
  - Beispiel: Binärzähler
- 2 Aggregat-Methode
- 3 Account-Methode
- 4 Potenzialmethode

Einführung

- Die **amortisierte Laufzeitanalyse** betrachtet die durchschnittlichen Kosten von Operationen in Folgen
- Unterschiede zur der Allgemeinen Analyse
- Die amortisierte Analyse wird eingesetzt zur Analyse von Operationen in Datenstrukturen.

Einführung

Beispiel: Stapeloperationen

- Unser Stapel besitzt die folgenden Operationen:

  PUSH(S, x), POP(S), MULTIPOP(S, k), STACK-EMPTY(S)

- 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: MULTIPOP(S, k) → T(N)
  - insgesamt: $O(N^2)$

Einführung

Beispiel: Binärzähler (1/2)

- Wollen k-Bit Zähler implementieren, der bei 0 beginnt
- Realisieren Zähler durch ein k-elementiges Array A[0..k- 1] von Bits. Niedrigstes Bit in A[0] und höchstes Bit in A[k-1] gespeichert.
- Array stellt damit zu jedem Zeitpunkt Zahl

$$x = \sum A[i]\, 2^i$$

Einführung

Beispiel: Binärzähler (2/2)

- **INCREMENT (A)**

  i = 0

  while i < länge [A] und A[i] = 1

  do {A[i] = 0 ; i = i+1 }

- Laufzeit auf Sequenz von N Operationen?
  - im worst case: INCREMENT (A)→ dauert O(N)
  - insgesamt: O(kN)

Aggregat-Methode

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**.

Aggregat-Methode

Beispiel: Analyse der Sequenz von N Operationen:

- Folge von **N PUSH**, **POP und MULTIPOP**

  Operationen kann höchstens **O(N) kosten**
- Die Gesamtzeit einer Sequenz ist also O(N)
- Die amortisierten Kosten aller drei Stapel-operationen sind somit O(1)

Account-Methode

Methode:

- für jede Operation amortisierte Kosten
- Die Differenz speziellen Objekten in der Datenstruktur als **Kredit** zugewiesen
- tatsächlichen Kosten **<=** amortisierten Kosten

$$\sum_{i=1}^{n} c_i \leq \sum_{i=1}^{n} a_i$$

Account-Methode

Analyse für Multi-Stack:

- Zuordnung der Kosten:

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

- jede Operation in der Sequenz ist bezahlbar ?

Account-Methode

Analyse für Zähler:

- Zuordnung der Kosten:

  die amortisierten Kosten der Flips von 0

  auf 1 auf **2** gesetzt. Für den Flip von 1 auf 0 sind die amortisierten Kosten **0**.

- jede Operation in der Sequenz ist **bezahlbar** ?

Potentialmethode

Methode:

- Wie bei der Buchhaltungsmethode werden Kosten im voraus bezahlt.
- Die im Voraus bezahlten Kosten sind im so genannten **Potential** einer Datenstruktur gesammelt.
- Potenzialfunktion zu Beginn gleich 0 und danach immer **nicht-negativ.**

Potentialmethode

Methode:

- Funktion Φ bildet jede Datenstruktur D_i auf eine reelle Zahl Φ_i ab, die dem Potenzial entspricht
- amortisierte Kosten :
  - $a_i = t_i + \Phi_i - \Phi_{i-1}$
- Sei **O1,...,On** eine Folge von **n** Operationen

Potentialmethode

Methode:

- Summe der amortisierten Kosten

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

- Vorgehen:
  - Potenzialfunktion wählen
  - $\Phi_n - \Phi_0 \geq 0$ Und damit $\Sigma t_i \leq \Sigma a_i$

Potentialmethode

Analyse für Multi-Stack:

- Zuordnung der Potenzialfunktion: $\phi_0 = 0$
- 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)**:
  - Potenzialdifferenz: $\Phi_i - \Phi_{i-1} = k'$
  - Amortisierte Kosten: $a_i = t_i + \Phi_i - \Phi_{i-1} = k' - k' = 0$

Potentialmethode

Analyse für Zähler:

- Das Potenzial setzen als b_i=Anzahl der Einser
- i-te INCREMENT-OP. Setzt z_i Bits auf 0
- Die tatsächlichen Kosten sind z_i+1
- Im Fall $b_i=0$ 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: $a_i = t_i + \Phi_i - \Phi_{i-1} \leq (z_i+1)+(1-z_i)=2$

$\Phi_i\geq 0$ $\Phi_0=0$

Quellen

- http://de.wikipedia.org/wiki/AmortisierteLaufzeitanalyse
- http://i11www.iti.uni-karlsruhe.de/_media/teaching/winter2006/algotech/skript.pdf
- T.H. Cormen, C.E. Leiserson, R. Rivest, C. Stein: Algorithmen – Eine Einführung, Oldenbourg, 2004

Danke für Ihre Aufmerksamkeit