Datenstrukturen, Algorithmen und Programmierung 2

Amin Coja-Oghlan

June 29, 2023

Lehrstuhl Informatik 2
Fakultät für Informatik

Amortisierte Analyse

Worum geht es?

- Datenstrukturen sind wichtige Hilfsmittel der Algorithmik
- Beispiele: Stacks, Splay Trees, Fibonacci Heaps...
- Die meisten Operationen benötigen wenig Zeit
- Ab und zu ist aber “Aufräumen” nötig
- *Idee:* Analyse der Gesamtlaufzeit durch kreative Buchführung

Beispiel: Stapel

Datenstruktur Stapel

- Funktionalität: `push`, `pop`, `empty`
- Abzubilden im Hauptspeicher ('random access')
- Zahl der zu speichernden Einträge a priori unbekannt
- *Ziel:* Zeit– und Speicherplatzeffizienz.

Beispiel: Stapel

$\square$

Umsetzung

- Reserviere einen Speicherblock der Größe n
- Anfangs $n = 1$
- Wenn der Block voll ist, reserviere einen Block der Größe $2n$
- Kopiere den Inhalt des alten Blocks in den neuen
- $\rightsquigarrow$ einzelne `push`-Operation zeitintensiv
- (Freigeben von Speicherplatz normalerweise einfacher.)

Beispiel: Stapel

Umsetzung

- Reserviere einen Speicherblock der Größe n
- Anfangs $n = 1$
- Wenn der Block voll ist, reserviere einen Block der Größe $2n$
- Kopiere den Inhalt des alten Blocks in den neuen
- $\rightsquigarrow$ einzelne `push`-Operation zeitintensiv
- (Freigeben von Speicherplatz normalerweise einfacher.)

Beispiel: Stapel

Umsetzung

- Reserviere einen Speicherblock der Größe n
- Anfangs $n = 1$
- Wenn der Block voll ist, reserviere einen Block der Größe $2n$
- Kopiere den Inhalt des alten Blocks in den neuen
- $\rightsquigarrow$ einzelne `push`-Operation zeitintensiv
- (Freigeben von Speicherplatz normalerweise einfacher.)

Beispiel: Stapel

Umsetzung

- Reserviere einen Speicherblock der Größe n
- Anfangs $n = 1$
- Wenn der Block voll ist, reserviere einen Block der Größe $2n$
- Kopiere den Inhalt des alten Blocks in den neuen
- $\rightsquigarrow$ einzelne `push`-Operation zeitintensiv
- (Freigeben von Speicherplatz normalerweise einfacher.)

Beispiel: Stapel

Umsetzung

- Reserviere einen Speicherblock der Größe n
- Anfangs $n = 1$
- Wenn der Block voll ist, reserviere einen Block der Größe $2n$
- Kopiere den Inhalt des alten Blocks in den neuen
- $\rightsquigarrow$ einzelne `push`-Operation zeitintensiv
- (Freigeben von Speicherplatz normalerweise einfacher.)

Beispiel: Stapel

Umsetzung

- Reserviere einen Speicherblock der Größe n
- Anfangs $n = 1$
- Wenn der Block voll ist, reserviere einen Block der Größe $2n$
- Kopiere den Inhalt des alten Blocks in den neuen
- $\rightsquigarrow$ einzelne `push`-Operation zeitintensiv
- (Freigeben von Speicherplatz normalerweise einfacher.)

Beispiel: Stapel

Umsetzung

- Reserviere einen Speicherblock der Größe n
- Anfangs $n = 1$
- Wenn der Block voll ist, reserviere einen Block der Größe $2n$
- Kopiere den Inhalt des alten Blocks in den neuen
- $\rightsquigarrow$ einzelne `push`-Operation zeitintensiv
- (Freigeben von Speicherplatz normalerweise einfacher.)

Beispiel: Stapel

Umsetzung

- Reserviere einen Speicherblock der Größe n
- Anfangs $n = 1$
- Wenn der Block voll ist, reserviere einen Block der Größe $2n$
- Kopiere den Inhalt des alten Blocks in den neuen
- $\rightsquigarrow$ einzelne `push`-Operation zeitintensiv
- (Freigeben von Speicherplatz normalerweise einfacher.)

Beispiel: Stapel

Umsetzung

- Reserviere einen Speicherblock der Größe n
- Anfangs $n = 1$
- Wenn der Block voll ist, reserviere einen Block der Größe $2n$
- Kopiere den Inhalt des alten Blocks in den neuen
- $\rightsquigarrow$ einzelne `push`-Operation zeitintensiv
- (Freigeben von Speicherplatz normalerweise einfacher.)

Beispiel: Stapel

Beispiel: $N\times$`push`, $N\times$`pop`

- Nehmen wir an, $N = 2^k$ ist eine Potenz von 2.
- Dann wird bei den `push`-Operationen

$$2, \quad 3, \quad 5, \quad 9, \quad 17, \quad \ldots, \quad 2^{k-1}+1$$

jeweils die Größe des Speicherblocks verdoppelt.
- Der Zeitbedarf für die letzte Verdopplung allein ist

$$2^{k-1}+1 > N/2.$$

- *Naive Analyse:* Zeitbedarf $O(N^2)$

Beispiel: Stapel

Amortisierte Analyse

- Angenommen insgesamt N `push`-Operationen
- *Behauptung:* Gesamtlaufzeit $O(N)$

Beispiel: Stapel

Amortisierte Analyse

- Angenommen insgesamt N `push`-Operationen
- *Behauptung:* Gesamtlaufzeit $O(N)$
- Direkte Analyse: Zeitaufwand für Reallokation.
- Mit $\ell = \lceil \log_2 N \rceil$ ist dieser beschränkt durch

$$\sum_{i=1}^{\ell} 2^{i-1} = \sum_{i=0}^{\ell-1} 2^i \leq 2^\ell = O(N).$$

Beispiel: Stapel

Amortisierte Analyse

- Angenommen insgesamt N `push`-Operationen
- Gesamtlaufzeit $O(N)$
- Die *amortisierte Laufzeit* pro `push` ist also $O(1)$

Amortisierte Analyse

Informelle Definition

- Angenommen die tatsächlichen Kosten einer Folge von Operationen sind

$$c_1, c_2, \ldots, c_N \geq 0$$

- Sei $c'_1, \ldots, c'_N$ eine weitere Folge, so daß

$$c_1 + c_2 + \cdots + c_i \leq c'_1 + c'_2 + \cdots + c'_i \qquad \text{für alle } 1 \leq i \leq N.$$

- Dann ist $c'_1, \ldots, c'_N$ eine *Amortisierung* von $c_1, \ldots, c_N$.
- Die *amortisierten Gesamtkosten* sind $c'_1 + \cdots + c'_N$.

Amortisierte Analyse

Woche	Kosten	Amortisierung
1	$c_1 = 0$	$c'_1 = 10$
2	$c_2 = 0$	$c'_2 = 10$
3	$c_3 = 0$	$c'_3 = 10$
4	$c_4 = 0$	$c'_4 = 10$
5	$c_5 = 50$	$c'_5 = 10$

Beispiel Fahrrad

- Angenommen ich radle täglich zur Uni
- Die meisten Tage entstehen keine Kosten
- Aber gelegentlich entstehen (relativ) hohe Reparaturkosten
- *Idee:* lege diese Kosten auf alle Tage/Kilometer um...
- ...um die Gesamtkosten einfacher kalkulieren zu können

Amortisierte Analyse

Potentialfunktionen

- Wie findet man eine geeignete Amortisierung?
- Hilfsmittel: Potentialfunktion $\Phi : \mathcal{Z} \to \mathbb{R}_{\geq 0}$
- *Idee:* Φ misst die 'Unordnung' des aktuellen Zustands
- Für eine leere Datenstruktur gelte $\Phi(\emptyset) = 0$
- Wenn $z_0 = \emptyset, z_1, \ldots, z_N$ die Zustände der Datenstruktur sind, definiere

$$c_i' = c_i + \Phi(z_i) - \Phi(z_{i-1})$$

Amortisierte Analyse

Potentialfunktionen

- Wir definieren

$$c'_i = c_i + \Phi(z_i) - \Phi(z_{i-1}) \qquad (i \geq 1)$$

- *Behauptung:* $c_1 + \cdots + c_i \leq c'_1 + \cdots + c'_i$ für alle i
- Zum Beweis berechnen wir

$$\begin{aligned} c'_1 + \cdots + c'_i &= (c_1 + \Phi(z_1) - \Phi(z_0)) + \cdots + (c_i + \Phi(z_i) - \Phi(z_{i-1})) \\ &= c_1 + \cdots + c_i + (\Phi(z_1) - \Phi(z_0)) + \cdots + (\Phi(z_i) - \Phi(z_{i-1})) \\ &= c_1 + \cdots + c_i + \Phi(z_i) - \Phi(z_0) \\ &= c_1 + \cdots + c_i + \Phi(z_i) \geq c_1 + \cdots + c_i \end{aligned}$$

Beispiel Stapel

Analyse via Potentialfunktion

- Der Zustand z_i besteht aus

$$b_i = \#\text{belegte Plätze}, \qquad n_i = \text{reservierter Speicherplatz}$$

- Wir definieren

$$\Phi(z_i) = \max\{0, 2b_i - n_i\}$$

Beispiel Stapel

Analyse via Potentialfunktion

- Die Potentialfunktion ist $\Phi(z_i) = \max\{0, 2b_i - n_i\}$
- Amortisierte Kosten:

$$
\begin{aligned}
c_i' &= 3 && \text{für } \mathrm{push} \text{ ohne Vergrößern} \\
c_i' &= 3 && \text{für } \mathrm{push} \text{ mit Vergrößern} \\
c_i' &= -1 && \text{für } \mathrm{pop}
\end{aligned}
$$

- Also $c_i' = O(1)$ für alle i
- Folglich $c_1 + \cdots + c_N = O(N)$

Weitere Beispiele

- Splay trees: amortisiert $O(\log n)$ `search`, `insert`, `delete`
- Dictionary: amortisiert $O(\log n)$ `insert`
- Fibonacci heap: amortisiert $O(1)$ `insert`, `findMin`, `decKey`; $O(\log n)$ `delMin`

Anwendung: Dijkstra-Algorithms

Satz

Der Dijkstra-Algorithmus hat Laufzeit $O(|E_G| + |V_G| \log |V_G|)$.

Beweis

- Jeder Knoten wird nur einmal der Halde hinzugefügt/entfernt
- Gesamtaufwand dafür ist $O(|V_G| \log |V_G|)$
- Ferner wird für jede Kante einmal `decKey` aufgerufen
- Laufzeit dafür $O(|E|)$

Zusammenfassung

- Amortisierung ist ein Analysewerkzeug für Datenstrukturen
- Ziel: Gesamtlaufzeit über viele Operationen bestimmen
- Hilfsmittel: Potentialfunktionen
- Einschränkungen: z.B. kritische Reaktionszeiten