



Abgabe: **17.05.2016** (bis 23:59 Uhr)

Oh my GAD!
Wir wünschen erholsame Pfingstferien :-)

Aufgabe 4.1 (P) Bankkonto-Methode

Diese Tutoraufgabe ist ein Kurztutorial für die Bankkonto-Methode, die in der Vorlesung für dynamische Arrays verwendet wurde. Die zweite Tutoraufgabe veranschaulicht die Theorie an verallgemeinerten dynamischen Arrays.

Sei S eine Menge von Operationen, und bezeichne $T(\sigma)$ (eine obere Schranke für) die Laufzeit einer Operation $\sigma \in S$ (diese Laufzeit kann vom aktuellen Zustand des Objekts, auf dem die Operation wirkt, sowie von Argumenten abhängen). Analog dazu bezeichne $T(\sigma_1, \sigma_2, \dots, \sigma_m) := \sum_{i=1}^m T(\sigma_i)$ (die korrespondierende obere Schranke für) die Laufzeit der Operationsfolge $(\sigma_1, \sigma_2, \dots, \sigma_m)$.

Ziel einer amortisierten Analyse ist, eine möglichst gute obere Schranke für $T(\sigma_1, \sigma_2, \dots, \sigma_m)$ zu finden. Bei der Konto-Methode bestimmen wir dazu eine Funktion $\Delta : S \rightarrow \mathbb{R}$, die folgende Eigenschaften besitzt:

- (i) Für alle legalen (d.h. ausführbaren) Operationsfolgen $(\sigma_1, \dots, \sigma_m)$ gilt $\sum_{i=1}^m \Delta(\sigma_i) \geq 0$.
- (ii) Δ ist möglichst gut gewählt.

Was Eigenschaft (ii) bedeutet, wird später noch spezifiziert. Für $\sigma \in S$ ist $\Delta(\sigma)$ die *Veränderung des Tokenkontos* durch die Operation σ . Wenn $\Delta(\sigma) > 0$, dann zahlt die Operation auf das Konto ein, falls $\Delta(\sigma) < 0$, dann hebt sie vom Konto ab. $A(\sigma) := T(\sigma) + \Delta(\sigma)$ nennen wir dann die *amortisierte Laufzeit* von σ . (In der Vorlesung wird $A(\sigma)$ auch Tokenlaufzeit genannt.) Entsprechend ist $A(\sigma_1, \dots, \sigma_m) := \sum_{i=1}^m A(\sigma_i)$ die amortisierte Laufzeit der Operationsfolge $(\sigma_1, \dots, \sigma_m)$.

Eigenschaft (i) sagt aus, dass das Tokenkonto nie negativ ist. Die Sinnhaftigkeit eines stets nichtnegativen Kontos ergibt sich aus folgender Beobachtung:

$$\begin{aligned} A(\sigma_1, \dots, \sigma_m) &= \sum_{i=1}^m A(\sigma_i) = \sum_{i=1}^m (T(\sigma_i) + \Delta(\sigma_i)) \\ &= T(\sigma_1, \dots, \sigma_m) + \underbrace{\sum_{i=1}^m \Delta(\sigma_i)}_{\geq 0} \geq T(\sigma_1, \dots, \sigma_m) \end{aligned}$$

Die amortisierte Laufzeit der Operationsfolge ist also eine *obere Schranke* für ihre tatsächliche Laufzeit. Jede Tokeneinheit steht für eine gewisse konstante Menge an Laufzeit. Damit



Abbildung 1: Pinguine

wird auch klar, was die (möglicherweise nicht ganzzahlige) Tokenzahl auf dem Konto aussagt: Es ist genau der Wert, um den die amortisierte Laufzeit die durch T gegebene (obere Schranke für die) tatsächliche Laufzeit übersteigt.

Nun zu Eigenschaft (ii). Das wichtigste Ziel ist, dass die amortisierten Laufzeiten $A(\sigma_1, \dots, \sigma_m)$ von Operationsfolgen asymptotisch möglichst klein sind, damit man gute obere Schranken für die tatsächliche Laufzeit von Operationsfolgen erhält.

Zu diesem Zwecke ist es oft zielführend, Δ so zu wählen, dass $\max_{\sigma \in S}(A(\sigma))$ möglichst klein ist, d.h. die Operation mit der schlechten amortisierte Laufzeit soll möglichst geringe amortisierte Laufzeit besitzen. Wenn dieses Ziel erreicht ist, ist $\mathcal{O}(m \cdot \max_{\sigma \in S}(A(\sigma)))$ eine obere Schranke für die asymptotische Laufzeit von Worst-Case-Operationsfolgen (wobei die Länge m der Operationsfolgen die asymptotische Variable ist). Bei einer hinreichend guten Wahl von Δ und Analyse ist diese obere Schranke oft nicht schlecht.

Aufgabe 4.2 (P) c-universelles Hashing mit Pinguinen

Die Pinguingattungen (vgl. auch Abbildung 1) {Brillenpinguin, Zwergpinguin, Eselspinguin, Kaiserpinguin, Goldschopfpinguin} sollen in einer Hash-Tabelle der Größe $m = 4$ untergebracht werden. Es seien folgende Hashfunktionen gegeben:

f_1 :	Brillenping.	$\mapsto 4$	Zwergp ping.	$\mapsto 2$	Eselsping.	$\mapsto 2$	Kaiserp ping.	$\mapsto 1$	Goldschopf ping.	$\mapsto 4$
f_2 :	Brillenping.	$\mapsto 3$	Zwergp ping.	$\mapsto 4$	Eselsping.	$\mapsto 2$	Kaiserp ping.	$\mapsto 3$	Goldschopf ping.	$\mapsto 4$
f_3 :	Brillenping.	$\mapsto 2$	Zwergp ping.	$\mapsto 2$	Eselsping.	$\mapsto 4$	Kaiserp ping.	$\mapsto 1$	Goldschopf ping.	$\mapsto 1$
f_4 :	Brillenping.	$\mapsto 1$	Zwergp ping.	$\mapsto 3$	Eselsping.	$\mapsto 3$	Kaiserp ping.	$\mapsto 4$	Goldschopf ping.	$\mapsto 4$
g_1 :	Brillenping.	$\mapsto 1$	Zwergp ping.	$\mapsto 1$	Eselsping.	$\mapsto 3$	Kaiserp ping.	$\mapsto 2$	Goldschopf ping.	$\mapsto 3$
g_2 :	Brillenping.	$\mapsto 2$	Zwergp ping.	$\mapsto 4$	Eselsping.	$\mapsto 2$	Kaiserp ping.	$\mapsto 3$	Goldschopf ping.	$\mapsto 4$
g_3 :	Brillenping.	$\mapsto 4$	Zwergp ping.	$\mapsto 4$	Eselsping.	$\mapsto 1$	Kaiserp ping.	$\mapsto 4$	Goldschopf ping.	$\mapsto 2$
g_4 :	Brillenping.	$\mapsto 3$	Zwergp ping.	$\mapsto 1$	Eselsping.	$\mapsto 2$	Kaiserp ping.	$\mapsto 3$	Goldschopf ping.	$\mapsto 3$
g_5 :	Brillenping.	$\mapsto 4$	Zwergp ping.	$\mapsto 2$	Eselsping.	$\mapsto 2$	Kaiserp ping.	$\mapsto 2$	Goldschopf ping.	$\mapsto 3$

In der Vorlesung haben wir den Begriff der c -universellen Hashfunktionen kennengelernt.

- (a) Geben Sie für die Familie $\mathcal{H}_1 = \{f_1, f_2, f_3, f_4\}$ das kleinste c an, so dass \mathcal{H}_1 c -universell ist.
- (b) Finden Sie eine möglichst kleine Familie $\mathcal{H}_2 \subseteq \{g_1, g_2, g_3, g_4, g_5\}$, die 1-universell ist.

Untermauern Sie Ihre Aussagen mit glaubwürdigen Argumenten.

Lösungsvorschlag 4.2

Eine Familie \mathcal{H} von Hashfunktionen ist nach Definition c -universell, wenn für alle Paare x, y mit $x \neq y$ gilt

$$\frac{|\{f \in \mathcal{H} : f(x) = f(y)\}|}{|\mathcal{H}|} \leq \frac{c}{m}.$$

Für $m = 4$ und $|\mathcal{H}_1| = 4$ bestimmt die Anzahl der Kollisionen pro Paar (x, y) direkt das c . Wir stellen zunächst eine Tabelle der Paare und ihren Kollisionen auf:

Paar	f_1	f_2	f_3	f_4	g_1	g_2	g_3	g_4	g_5
Brillenpinguin/Zwergpinguin			x		x		x		
Brillenpinguin/Eselspinguin						x			
Brillenpinguin/Kaiserpinguin		x					x	x	
Brillenpinguin/Goldschopfpinguin	x							x	
Zwergpinguin/Eselspinguin	x			x					x
Zwergpinguin/Kaiserpinguin						x		x	
Zwergpinguin/Goldschopfpinguin		x				x			
Eselspinguin/Kaiserpinguin									x
Eselspinguin/Goldschopfpinguin					x				
Kaiserpinguin/Goldschopfpinguin			x	x				x	

Da es für \mathcal{H}_1 für die Werte Zwergpinguin/Eselspinguin sowie Kaiserpinguin/Goldschopfpinguin jeweils zwei Kollisionen gibt und sonst zwischen alle Wertepaaren höchstens eine Kollision auftritt, ist \mathcal{H}_1 also 2-universell.

Da \mathcal{H}_2 1-universell sein soll und jede Funktion aufgrund des Schubfachprinzips eine Kollision verursachen muss, muss also \mathcal{H}_2 mindestens 4 Funktionen beinhalten.

Definieren wir nun $\mathcal{H}_2 := \{g_1, g_2, g_4, g_5\}$ so ist diese Familie 1-universell, da für jedes Paar nur eine Kollision auftritt.

Aufgabe 4.3 (P) Universelles Hashing mit Chaining

Veranschaulichen Sie Hashing mit Chaining. Die Größe m der Hash-Tabelle ist in den folgenden Beispielen jeweils die Primzahl 11. Die folgenden Operationen sollen nacheinander ausgeführt werden.

Insert 3, 11, 9, 7, 14, 56, 4, 12, 15, 8, 1
Delete 56
Insert 25

Der Einfachheit halber sollen die Schlüssel der Elemente die Elemente selbst sein.

- a) Verwenden Sie zunächst die Hashfunktion

$$g(x) = 5x \mod m.$$

b) Verwenden Sie jetzt die Hashfunktion

$$h(x) = \mathbf{a} \cdot \mathbf{x} \mod m$$

nach dem aus der Vorlesung bekannten Verfahren für einfache universelle Hashfunktionen, wobei $\mathbf{a} = (7, 5)$ und $\mathbf{x} = (\lfloor \frac{x}{2^w} \rfloor \mod 2^w, x \mod 2^w)$ für $w = \lfloor \log_2 m \rfloor = \lfloor 3.45\dots \rfloor = 3$ gilt und der Ausdruck $\mathbf{a} \cdot \mathbf{x}$ ein Skalarprodukt bezeichnet. Vergleichen Sie die verwendete Hashfunktion mit der aus der vorigen Teilaufgabe.

Lösungsvorschlag 4.3

a) $g(x) = 5x \mod 11$:

Der Einfachheit halber berechnen wir als erstes die Positionen der betrachteten Elemente. Die Funktion k soll zum Element e den Schlüssel liefern, hier gilt also $k(e) = e$.

$k(e)$	1	3	4	7	8	9	11	12	14	15	25	56
$g(k(e))$	5	4	9	2	7	1	0	5	4	9	4	5

Insert(3):

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

Insert(11):

0	1	2	3	4	5	6	7	8	9	10
11				3						

Insert(9):

0	1	2	3	4	5	6	7	8	9	10
11	9			3						

Insert(7):

0	1	2	3	4	5	6	7	8	9	10
11	9	7		3						

Insert(14):

0	1	2	3	4	5	6	7	8	9	10
11	9	7		3						

Insert(56):

0	1	2	3	4	5	6	7	8	9	10
11	9	7		3	56					

Insert(4):

0	1	2	3	4	5	6	7	8	9	10
11	9	7		3	56				4	

Insert(12):

0	1	2	3	4	5	6	7	8	9	10
11	9	7		3	56				4	

Insert(15):

0	1	2	3	4	5	6	7	8	9	10
11	9	7		3	56				4	

Insert(8):

0	1	2	3	4	5	6	7	8	9	10
11	9	7		3	56			8	4	

Insert(1):

0	1	2	3	4	5	6	7	8	9	10
11	9	7		3	56			8	4	

Delete(56):

0	1	2	3	4	5	6	7	8	9	10
11	9	7		3	12			8	4	

Insert(25):

0	1	2	3	4	5	6	7	8	9	10
11	9	7		3	12			8	4	

b) $h(x) = \mathbf{a} \cdot \mathbf{x} \mod 11$:

Wir berechnen wieder als erstes die Positionen der betrachteten Elemente. Zum Beispiel gilt $h(k(14)) = h(14) = ((7, 5) \cdot (\lfloor \frac{14}{8} \rfloor \mod 8, 14 \mod 8)) \mod 11 = ((7, 5) \cdot (1 \mod 8, 6 \mod 8)) \mod 11 = (7 \cdot 1 + 5 \cdot 6) \mod 11 = 37 \mod 11 = 4$.

$k(e)$	1	3	4	7	8	9	11	12	14	15	25	56
$h(k(e))$	5	4	9	2	7	1	0	5	4	9	4	5

Einfügen und Löschen läuft dann analog zur vorigen Teilaufgabe. Es gilt $7 = (5 \cdot 8) \bmod 11 = (5 \cdot 2^w) \bmod m$, d.h. die Hashfunktion entspricht mit den Parametern 7 und 5 von **a** genau der in a) gegebenen Hashfunktion $5x \bmod 11$.