

Tutorium Grundlagen: Algorithmen und Datenstrukturen

Übungsblatt Woche 10

Aufgabe 10.1

Universelles Hashing

Aufgabe 10.1

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

f_1 :	Brillenping.	Zwergping.	Eselsping.	Kaiserp ping.	Goldschopfp ping.
f_2 :	Brillenping.	Zwergping.	Eselsping.	Kaiserp ping.	Goldschopfp ping.
f_3 :	Brillenping.	Zwergping.	Eselsping.	Kaiserp ping.	Goldschopfp ping.
f_4 :	Brillenping.	Zwergping.	Eselsping.	Kaiserp ping.	Goldschopfp ping.
g_1 :	Brillenping.	Zwergping.	Eselsping.	Kaiserp ping.	Goldschopfp ping.
g_2 :	Brillenping.	Zwergping.	Eselsping.	Kaiserp ping.	Goldschopfp ping.
g_3 :	Brillenping.	Zwergping.	Eselsping.	Kaiserp ping.	Goldschopfp ping.
g_4 :	Brillenping.	Zwergping.	Eselsping.	Kaiserp ping.	Goldschopfp ping.
g_5 :	Brillenping.	Zwergping.	Eselsping.	Kaiserp ping.	Goldschopfp ping.

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

- Geben Sie für die Familie $H_1 = \{f_1, f_2, f_3, f_4\}$ das kleinste c an, so dass H_1 c-universell ist.
- Finden Sie eine möglichst kleine Familie $H_2 \subseteq \{g_1, g_2, g_3, g_4, g_5\}$, die 1-universell ist. Untermauern Sie Ihre Aussagen mit glaubwürdigen Argumenten.

$$\frac{|\{f \in H : f(x) = f(y)\}|}{|H|} \leq \frac{c}{m} \quad \forall x \neq y$$

Aufgabe 10.1 (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.

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

$\underbrace{\mathcal{H}}_{4} \quad \underbrace{m}_{4}$

$$|\{f \in \mathcal{H} : f(x) = f(y)\}| \leq c \quad \forall x \neq y$$

2

$\leq c$

\Downarrow
 $\underline{2}$ (so klein
wie möglich)

- Diese Ungleichung muss für alle $x \neq y$ gelten.

- Da c konstant ist, suchen wir die größte Zahl für die rechte Seite um eine untere Schranke für c zu erhalten \Rightarrow Höchste Kollisionszahl für ein Wertepaar

\mathcal{H}

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		x	
Brillenpinguin/Kaiserpinguin		x				x	x		
Brillenpinguin/Goldschopfpinguin	x						x		
Zwergpinguin/Eselspinguin	x			x				x	
Zwergpinguin/Kaiserpinguin		x				x		x	
Zwergpinguin/Goldschopfpinguin		x			x		x		
Eselspinguin/Kaiserpinguin						x			x
Eselspinguin/Goldschopfpinguin					x			x	
Kaiserpinguin/Goldschopfpinguin		x	x					x	

Aufgabe 10.1 (a)

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.

Schubfachprinzip:
 (min. 1 Kollision in jeder Funktion, da 5 Werte in 4 Plätzen gehasht werden)

- Angenommen Linker Nenner ist genau 1 (minimal)

$$\frac{1}{|\mathcal{H}|} \leq \frac{1}{4}$$

$\Rightarrow |\mathcal{H}| \geq 4 \Rightarrow$ min. 4 Funktionen in \mathcal{H}_2 nötig

- Angenommen alle Funktionen g_1, \dots, g_5 in \mathcal{H}_2 :
 $\Rightarrow |\mathcal{H}| = 5$, maximale Kollisionen bei Paar sind 2

$$\Rightarrow \frac{2}{5} \leq \frac{1}{4} \xrightarrow{\text{Widerspruch}}$$

\Rightarrow 4 Funktionen in \mathcal{H}_2 & maximal eine Kollision pro Paar

$$1 \leq \frac{|\{f \in \mathcal{H} : f(x) = f(y)\}|}{|\mathcal{H}|} \leq \frac{c}{m} \quad \forall x \neq y$$

1
c
m
4

→

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	

g_3 hat für jedes Paar mit 2 Kollisionen eine dieser verursacht

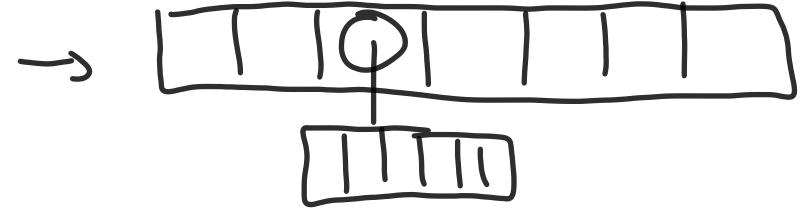
\Rightarrow L^o entferne $g_3 \Rightarrow$ 4 Funktionen, max. 1 Kollision pro Paar ✓

$$\Rightarrow \mathcal{H}_2 = \{g_1, g_2, g_4, g_5\}$$

Aufgabe 10.2

Die perfekte Hashtabelle

Aufgabe 10.2



Konstruieren Sie eine statische perfekte Hashtabelle für die Elemente:

$$n=12$$

(16, 10, 11)	(8, 2, 15)	(7, 12, 8)	(1, 10, 3)	(13, 11, 14)	(6, 11, 14)
(7, 3, 16)	(2, 2, 8)	(10, 5, 15)	(7, 3, 14)	(2, 10, 1)	(14, 11, 6)

Jedes Element x besteht aus den Stellen (x_0, x_1, x_2) . Verwenden Sie jeweils passend eine der Hashfunktionen:

$$\begin{aligned} & \rightarrow (\sum_{i=0}^2 2^i x_i) \bmod 17 \\ & (\sum_{i=0}^2 a_i x_i) \bmod 7 \text{ mit } \mathbf{a} = (0, 0, 1) \text{ oder } \mathbf{a} = (6, 6, 2) \\ & (\sum_{i=0}^2 a_i x_i) \bmod 3 \text{ mit } \mathbf{a} = (1, 0, 0) \text{ oder } \mathbf{a} = (0, 2, 2). \end{aligned}$$

Erinnerung: perfektes statisches Hashing

- • Ziehe Hashfunktion aus einer c -universellen Familie $H_{[\sqrt{2}cn]}$ bis Kollisionszahl kleiner gleich $\sqrt{2}n$
- • Jedes Bucket hat dann Größe $m_l = cb_l(b_l - 1) + 1$, wobei b_l die Elementanzahl im jeweiligen Bucket ist
 - Ziehe für jedes Bucket aus einer c -universellen Familie H_{m_l} bis im Bucket alle Schlüssel injektiv abgebildet werden

In dieser Aufgabe sind die möglichen Hashfunktionen schon gegeben

Aufgabe 10.2

$$\begin{aligned} & \text{(1)} (\sum_{i=0}^2 2^i x_i) \bmod 17 \\ & (\sum_{i=0}^2 a_i x_i) \bmod 7 \text{ mit } \mathbf{a} = (0, 0, 1) \text{ oder } \mathbf{a} = (6, 6, 2) \\ & (\sum_{i=0}^2 a_i x_i) \bmod 3 \text{ mit } \mathbf{a} = (1, 0, 0) \text{ oder } \mathbf{a} = (0, 2, 2). \end{aligned}$$

1. Stufe : • Größe $\lceil \sqrt{2} \cdot c \cdot n \rceil = \lceil \sqrt{2} \cdot c \cdot 12 \rceil = \lceil 16,97 \cdot c \rceil = 17 \cdot c$
Berechte $c \geq 1 \Rightarrow$ nur (1) kommt mit $\bmod 17$ infrage (impliziert $c=1$)

Andere Hashfunktionen sind für zu kleine Tabelle ausgelegt

- Maximal $\sqrt{2} \cdot n$ Kollisionen
→ Überprüfung auf nächster Seite

Aufgabe 10.2

Element	Bucket
(7, 3, 14)	1
(2, 2, 8)	4 } 2K.
(8, 2, 15)	4 }
(13, 11, 14)	6
(2, 10, 1)	9 } 6K.
(14, 11, 6)	9 }
(7, 3, 16)	9 }
(16, 10, 11)	12 } 6.K.
(7, 12, 8)	12 }
(10, 5, 15)	12 }
(1, 10, 3)	16 } 2K.
(6, 11, 14)	16 }

$$\begin{aligned} & (\sum_{i=0}^2 2^i x_i) \bmod 17 \\ & (\sum_{i=0}^2 a_i x_i) \bmod 7 \text{ mit } \mathbf{a} = (0, 0, 1) \text{ oder } \mathbf{a} = (6, 6, 2) \\ & (\sum_{i=0}^2 a_i x_i) \bmod 3 \text{ mit } \mathbf{a} = (1, 0, 0) \text{ oder } \mathbf{a} = (0, 2, 2). \end{aligned}$$

• Kollision wenn $h(x) = h(y)$ für $x \neq y \rightarrow$ Für Wertepaar a, b zwei Kollisionen wenn $h(a) = h(b)$
 $\hookrightarrow x=a, y=b$ oder $x=b, y=a$

$\Rightarrow c(h) = 2+6+6+2 = 16 \leq 16,97 \approx \sqrt{2} \cdot 12 = \sqrt{2} \cdot n \quad \checkmark$

Bucketgrößen für Buckets mit $b_L \geq 1$ Elementen: $m_L = c \cdot b_L \cdot (b_L - 1) + 1$ (Wir nehmen $c=1$ an
 sollte generell beachtet werden)

$$\left. \begin{array}{l} m_4 = 2 \cdot (2-1) + 1 = 3 \\ m_9 = 3 \cdot (3-1) + 1 = 7 \\ m_{12} = 3 \cdot (3-1) + 1 = 7 \\ m_{16} = 2 \cdot (2-1) + 1 = 3 \end{array} \right\} \begin{array}{l} \text{An Größen erkennbar: } m_4 \text{ & } m_{16} \text{ benötigen mod 3} \\ \text{ } \qquad \qquad \qquad m_9 \text{ & } m_{12} \text{ benötigen mod 7} \end{array}$$

\Rightarrow Teste die verschiedenen a bis keine Kollision im Bucket vorkommt. Dadurch ergeben sich:
 Bucket 4: $a = (0, 2, 2)$ Bucket 16: $a = (1, 0, 0)$ | Bucket 9: $a = (0, 0, 1)$ Bucket 12: $a = (6, 6, 2)$

\Rightarrow Finale Hashwerte auf nächster Seite

Aufgabe 10.2

Element	Bucket	Position in Bucket
(7, 3, 14)	1	0
(2, 2, 8)	4	2
(8, 2, 15)	4	1
(13, 11, 14)	6	0
(2, 10, 1)	9	1
(14, 11, 6)	9	6
(7, 3, 16)	9	2
(16, 10, 11)	12	3
(7, 12, 8)	12	4
(10, 5, 15)	12	1
(1, 10, 3)	16	1
(6, 11, 14)	16	0

Aufgabe 10.3

Linear Probing

Aufgabe 10.3

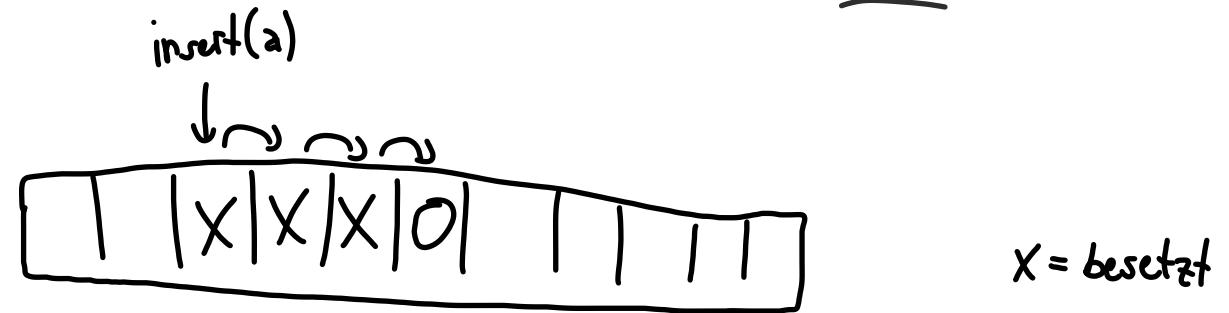
Veranschaulichen Sie Hashing mit Linear Probing. Die Größe der Hashtabelle ist dabei jeweils $m = 13$. Führen Sie die folgenden Operationen aus:

insert 16, 3, 12, 17, 29, 10, 24
delete 16
insert 5, 1, 15
delete 10
insert 14
delete 1

Verwenden Sie die Hashfunktion:

→ Beim Löschen soll die dritte Methode aus der Vorlesung verwendet werden, d.h. die Wiederherstellung der folgenden Invariante:

Für jedes Element e in der Hashtabelle mit Schlüssel $k(e)$, aktueller Position j und optimaler Position $i = h(k(e))$ sind alle Positionen $i, (i + 1) \bmod m, (i + 2) \bmod m, \dots, j$ der Hashtabelle belegt. Bei dieser Aufgabe soll keine dynamische Größenanpassung der Hashtabelle stattfinden.



Beim Entfernen müssen vom entfernten Element bis zum nächsten freien Feld alle Elemente auf die Invarianz überprüft und potentiell verschoben werden.

$$h(x) = 3x \bmod 13$$

Aufgabe 10.3

1. Operation: `insert(16)` mit opt. Position: 9

0	1	2	3	4	5	6	7	8	9	10	11	12
									<u>16</u>			

2. Operation: `insert(3)` mit opt. Position: 9

0	1	2	3	4	5	6	7	8	9	10	11	12
									<u>16</u>	<u>3</u>		

3. Operation: `insert(12)` mit opt. Position: 10

0	1	2	3	4	5	6	7	8	9	10	11	12
									<u>16</u>	<u>3</u>	<u>12</u>	

4. Operation: `insert(17)` mit opt. Position: 12

0	1	2	3	4	5	6	7	8	9	10	11	12
									<u>16</u>	<u>3</u>	<u>12</u>	<u>17</u>

5. Operation: `insert(29)` mit opt. Position: 9

0	1	2	3	4	5	6	7	8	9	10	11	12
									<u>29</u>		<u>16</u>	<u>3</u>

6. Operation: `insert(10)` mit opt. Position: 4

0	1	2	3	4	5	6	7	8	9	10	11	12
									<u>29</u>	<u>10</u>		<u>16</u>

7. Operation: `insert(24)` mit opt. Position: 7

0	1	2	3	4	5	6	7	8	9	10	11	12
									<u>29</u>	<u>10</u>	<u>24</u>	<u>16</u>

8. Operation: `delete(16)` mit opt. Position: 9

0	1	2	3	4	5	6	7	8	9	10	11	12
									<u>10</u>		<u>24</u>	<u>3</u>

9. Operation: `insert(5)` mit opt. Position: 2

0	1	2	3	4	5	6	7	8	9	10	11	12
									<u>5</u>	<u>10</u>		<u>24</u>

10. Operation: `insert(1)` mit opt. Position: 3

0	1	2	3	4	5	6	7	8	9	10	11	12
									<u>5</u>	<u>10</u>		<u>24</u>

11. Operation: `insert(15)` mit opt. Position: 6

0	1	2	3	4	5	6	7	8	9	10	11	12
									<u>5</u>	<u>10</u>	<u>15</u>	<u>24</u>

12. Operation: `delete(10)` mit opt. Position: 4

0	1	2	3	4	5	6	7	8	9	10	11	12
									<u>5</u>	<u>10</u>		<u>15</u>

13. Operation: `insert(14)` mit opt. Position: 3

0	1	2	3	4	5	6	7	8	9	10	11	12
									<u>5</u>	<u>10</u>	<u>14</u>	

14. Operation: `delete(1)` mit opt. Position: 3

0	1	2	3	4	5	6	7	8	9	10	11	12
									<u>5</u>	<u>10</u>	<u>14</u>	

Aufgabe 10.3

1. Operation: `insert(16)` mit opt. Position: 9

0	1	2	3	4	5	6	7	8	9	10	11	12
									16			

2. Operation: `insert(3)` mit opt. Position: 9

0	1	2	3	4	5	6	7	8	9	10	11	12
									16	3		

3. Operation: `insert(12)` mit opt. Position: 10

0	1	2	3	4	5	6	7	8	9	10	11	12
									16	3	12	

4. Operation: `insert(17)` mit opt. Position: 12

0	1	2	3	4	5	6	7	8	9	10	11	12
									16	3	12	17

5. Operation: `insert(29)` mit opt. Position: 9

0	1	2	3	4	5	6	7	8	9	10	11	12
									29	16	3	12

6. Operation: `insert(10)` mit opt. Position: 4

0	1	2	3	4	5	6	7	8	9	10	11	12
									29	10	16	3

7. Operation: `insert(24)` mit opt. Position: 7

0	1	2	3	4	5	6	7	8	9	10	11	12
									29	10	16	3

8. Operation: `delete(16)` mit opt. Position: 9

0	1	2	3	4	5	6	7	8	9	10	11	12
									10	24	3	12

9. Operation: `insert(5)` mit opt. Position: 2

0	1	2	3	4	5	6	7	8	9	10	11	12
					5	10			24	3	12	29

10. Operation: `insert(1)` mit opt. Position: 3

0	1	2	3	4	5	6	7	8	9	10	11	12
					5	1	10		24	3	12	29

11. Operation: `insert(15)` mit opt. Position: 6

0	1	2	3	4	5	6	7	8	9	10	11	12
					5	1	10		15	24	3	12

12. Operation: `delete(10)` mit opt. Position: 4

0	1	2	3	4	5	6	7	8	9	10	11	12
					5	1			15	24	3	12

13. Operation: `insert(14)` mit opt. Position: 3

0	1	2	3	4	5	6	7	8	9	10	11	12
					5	1	14		15	24	3	12

14. Operation: `delete(1)` mit opt. Position: 3

0	1	2	3	4	5	6	7	8	9	10	11	12
					5	14			15	24	3	12

Aufgabe 10.4

Double Hashing

Aufgabe 10 .4

Doppel-Hashing ist eine Methode zur Kollisionsbehandlung. Bei Kollisionen kommt eine Sondierungsfunktion zum Einsatz, die eine sekundäre Hashfunktion beinhaltet:

$$s(x, i) = i \cdot h_2(x), i \in \mathbb{N}_0$$

Diese Sondierungsfunktion wird angewendet, falls der durch die primäre Hashfunktion $h_1(x)$ berechnete Index bereits besetzt ist. Dabei wird i beginnend bei 0 bei jedem Versuch um 1 erhöht.

Die vollständige Hashfunktion lautet dann:

$$h(x, i) = (h_1(x) + s(x, i)) \bmod m$$

i startet bei jedem insert() und find() wieder bei 0

Verwenden Sie im Folgenden die Hashfunktionen

$$h_1(x) = (3x + 1) \bmod m$$

$$h_2(x) = 1 + (x \bmod (m - 1))$$

- Geben Sie die vollständige Hashfunktion $h(x, i)$ für eine Tabelle der Länge $m = 13$ an.
- Veranschaulichen Sie schrittweise das Einfügen der Schlüssel 12, 23, 13, 56, 26, 45, 24, 94, 42 in eine Hashtabelle der Länge $m = 13$. (Siehe Vordruck)

(a) Simples Einsetzen : $h(x, i) = \underbrace{(3x+1) \bmod 13}_{h_1(x)} + i \cdot \underbrace{(1+(x \bmod 12))) \bmod 13}_{\begin{array}{l} i \\ \cdot \\ h_2(x) \end{array}} \underbrace{s(x, i)}_{}$

Aufgabe 10.4

	ins(12)	ins(23)	ins(13)	ins(56)	ins(26)	ins(45)	ins(24)	ins(94)	ins(42)
0:				56					
1:			13						
2:									
3:									
4:				26					
5:		23							
6:					45				
7:									
8:						24			
9:									
10:							94		
11:	12								
12:								42	

Hashes

$i=0$	M	5	1	0	1	6	8	10	10
$i=1$		4	3	9	4	3	9	8	4
$i=2$					7	0	10	6	11
\vdots						11	4	5	12

Aufgabe 10.4

	ins(12)	ins(23)	ins(13)	ins(56)	ins(26)	ins(45)	ins(24)	ins(94)	ins(42)
0:				56	56	56	56	56	56
1:			13	13	13	13	13	13	13
2:									
3:									
4:					26	26	26	26	26
5:		23	23	23	23	23	23	23	23
6:						45	45	45	45
7:									
8:							24	24	24
9:									
10:								94	94
11:	12	12	12	12	12	12	12	12	12
12:									42