Hashbasierte Datenstrukturen.
1. Teil: Hashtabellen[a]

https://youtu.be/nsT4WwH1y-k

Name: ..... Vorname: .....

Individuelle Parameter: k_1: ....., k_2: ....., k_3: .....

P_{Koll}: .....

1. Hashfunktionen

Hashfunktionen sind dadurch gekennzeichnet, dass sie für einen alphanumerischen Input einen "Funktionswert" mit genau definierter Länge ergeben. Wie verwenden hier die SHA256 (National Institute of Standards and Technology). Eine von einer Hashfunktion geforderte Eigenschaft ist, dass sie "chaotisch" reagiert, d.h. geringfügig unterschiedliche Eingaben sollen völlig unterschiedliche Hashwerte erzeugen.

1.1 Hashkollisionen. Das Geburtstagsproblem

Man kann zeigen (https://youtu.be/T1ecmyc3AJ8), dass bei einer Hashfunktion mit 2^n möglichen die Wahrscheinlichkeit P_{Koll} für mindestens eine Kollision mit der Anzahl x von Hashwerten erreicht ist wie folgt:

$$x \approx \sqrt{2^n \ln\left(\frac{1}{1 - P_{Koll}}\right)}$$

Für den SHA256 gilt $n = 256$. Bestimme x für den oben angegebenen Zahlenwert von P_{Koll}. (Beispiel: $P_{Koll} = ½ \rightarrow x \approx 2.833 \cdot 10^{38}$).

Ergebnis: $x \approx$

[a] Unterlagen auf https://www.mathepauker.com/Musterex/Benedict/Projektarbeiten/Hashtabellen.pdf

©2020 mathepunk

1.2 Umkehrfunktion von SHA256

Hashfunktionen sind so genannte Einwegfunktionen. Mit der so genannten "Wörterbuchmethode" kann man jedoch für Milliarden von Hashwerten die Ausgangzeichenfolge eruieren.

Bestimme mit https://mathepauker.com/Blockchain/hash.html den SHA256 deiner aktuellen Wohngemeinde. Benütze den berechneten Hashwert in einem SHA256-Decoder, wie z.B. https://md5decrypt.net/en/Sha256/#answer als Input. Wird der eingegebene Wohnort angezeigt?

Antwort: **Wohnort:** ……..

Ergebnis:

Beispiel: Der Wohnort Reichenburg ergibt einen SHA256 Hashwert wie folgt:
f0592a82d48af147c06d80fc2ff81bbd56a8588f6320f90dd51f9a6b17ee4f19
Wenn man diesen Hashwert beim oben angegebenen SHA256-Decoder eingibt, so erhält man tatsächlich die Ausgangzeichenfolge "Reichenburg". Man muss vorsichtig sein, dass man bei der Berechnung des Hashwerts für SHA256 am Schluss nicht die Eingabetaste drückt, denn diese würde zur Ausgangzeichenfolge hinzugefügt.

2. Hashtabellen mit Hashkollisionen

Bei einer Hashtabelle werden Paare von Schlüssel und Dateien abgespeichert. Der Schlüssel ist im Prinzip der Name der Datei, aber er kann in der Hashtabelle auch gehasht abgespeichert werden und wird dadurch zu einem Schlüssel. Wir betrachten im Folgenden nur die Schlüssel, d.h. wir tun so als ob die Dateien leer seien. Wenn in der Hashtabelle L Einträge zur Verfügung stehen, dann soll die Hashfunktion einen Wertebereich von 1 bis L oder 0 bis L – 1 aufweisen. Wir arbeiten hier mit dem SHA256. Daraus machen wir eine für Hashtabellen geeignete Hashfunktion h wie folgt:

$$h_L(s) = \text{SHA256}(s) \bmod L + 1$$

Erstelle für 15 Wörter aus der Zufallswörtertabelle im Anhang zwei Proto-Hashtabellen für $L = 20$ und $L = 25$ wie unten gezeigt. Die ersten fünf der 15 Wörter sind Wort Nr. k_1, $k_1 + 1$, $k_1 + 2$, $k_1 + 3$ und $k_1 + 4$. Die zweiten fünf der 15 Wörter sind Wort Nr. k_2, $k_2 + 1$, $k_2 + 2$, $k_2 + 3$ und $k_2 + 4$. Die dritten fünf der 15 Wörter sind Wort Nr. k_3, $k_3 + 1$, $k_3 + 2$, $k_3 + 3$ und $k_3 + 4$. Im nachfolgenden Beispiel gilt $k_1 = 1$, $k_2 = 10$ und $k_3 = 25$. Die 15 Wörter sind dann für $k \in \{1;2;3;4;5;10;11;12;13;14;25;26;27;28;29\}$.

©2020 mathepunk

2.1. Proto-Hashtabellen

Beispiel für Proto-Hashtabellen:

k	Schlüssel (s)	Hashwert für L = 20 $h_{20}(s)$	Hashwert für L = 25 $h_{25}(s)$
1	soap	13	18
2	continue	10	20
3	industrious	18	18
4	pollution	11	21
5	breakable	3	8
10	substance	7	22
11	remarkable	16	11
12	invincible	10	10
13	evanescent	1	1
14	quack	5	20
25	imminent	1	1
26	quarrelsome	10	20
27	eggs	2	7
28	jagged	20	5
29	committee	12	2

Verwende für die Berechnung der Hashfunktion h(s) die Website

https://mathepauker.com/Blockchain/Merkle-tree.html

©2020 mathepunk

Proto-Hashtabellen

Schlüssel Nr.	Schlüssel (s)	Hashwert $h_{20}(s)$	Hashwert $h_{25}(s)$
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			

©2020 mathepunk

2.2 Behebung von Kollisionen

Bei Hashtabellen ist die Anzahl der Einträge im Allgemeinen so, dass die Wahrscheinlichkeit von Hashkollisionen gross ist. Eine Hashkollision liegt dann vor, wenn zwei oder mehr Schlüssel denselben Hashwert haben. Ein Überläufer ist ein Datensatz, dem durch den Hashwert des Schlüssels ein Speicherplatz zugeordnet wurde, der schon besetzt ist. Es gibt zahlreiche Methoden, um solche Überläufer unterzubringen, d.h. Hashkollisionen zu beheben. Es ist klar, dass es letztendlich darum geht für Überläufer einen freien Platz zu finden. Wir betrachten hier zwei einfache Methoden wie folgt:

1. Methode: Offene Adressierung, lineares Sondieren

2. Methode: Verkettung, verkette Listen

Bei der ersten Methode, der offenen Adressierung mit linearem Sondieren, kann man sich das so vorstellen, dass man zunächst an der vom Hashwert angegebenen Speicherplatz nachschaut. Wenn dieser besetzt ist sucht man von diesem Speicherplatz aus weiter, d.h. man sucht einen freien Speicherplatz nach dem vom Hashwert angezeigten Speicherplatz. Den Datensatz (samt Schlüssel) speichert man in dem ersten freien Speicherplatz, den man findet. Es kann sein, dass man bei der Suche ans Ende der Tabelle gelangt. In diesem Fall fährt man fort mit der Suche am Anfang der Tabelle fort. Dabei werden die Datensätze, resp. Schlüssel in derselben Reihenfolge der Hashtabelle beigefügt, wie sie in den Proto-Hashtabellen erscheinen.

Bei der 2. Methode, der Verkettung, wird jeder Tabelleneintrag als Adresse für eine verkettete Liste betrachtet. Datensätze mit demselben Hashwert des Schlüssels werden in einer verketteten Liste zusammengefasst. Bei dieser Übung sollte die Reihenfolge in den verketten Listen die Reihenfolge in den Proto-Hashtabellen wiederspiegeln.

In den Proto-Hashtabellen sind die Hash-Kollisionen nicht behoben. In den eigentlichen Hashtabellen werden die Hashkollisionen mit den beiden hier beschriebenen Methoden behoben. Unten werden als "gelöste Beispiele" nur Hashtabellen der Länge 20, d.h. mit $L = 20$, gezeigt. Ihr müsst jedoch mit beiden Methoden zusätzlich Hashtabellen der Länge 25 erstellen. In den Hashtabellen wird auch eingetragen wie viele Schritte man benötigt, um, ausgehend vom Speicherplatz gemäss Hashwert des Schlüssels, den Datensatz zu lokalisieren.

©2020 mathepunk

2.2.1 Offene Adressierung, lineares Sondieren

Beispiel für Hashtabelle mit offener Adressierung, lineares Sondieren

j_S: Speicherplatz Hashtabelle

$h_{20}(s)$: Hashwert

$j_S - h_{20}(s) + 1$: Anzahl Schritte beim linearen Sondieren

j_S	Schlüssel (s)	$h_{20}(s)$	$j_S - h_{20}(s) + 1$
1	evanescent	1	1
2	imminent	1	2
3	breakable	3	1
4	eggs	2	3
5	quack	5	1
6			
7	substance	7	1
8			
9			
10	continue	10	1
11	pollution	11	1
12	invincible	10	3
13	soap	13	1
14	quarrelsome	10	5
15	committee	12	4
16	remarkable	16	1
17			
18	industrious	18	1
19			
20	jagged	20	1
		Summe	27

Mittlere Anzahl Schritte beim Sondieren: $\frac{27}{15} = \mathbf{\underline{1.8}}$

©2020 mathepunk

Hashtabellen mit offener Adressierung, lineares Sondieren

j_S	**Schlüssel (s)**	$h_{20}(s)$	$j_S - h_{20}(s) + 1$
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
		Summe	

Mittlere Anzahl Schritte beim Sondieren:

©2020 mathepunk

j_S	**Schlüssel (s)**	$h_{25}(s)$	$j_S - h_{25}(s) + 1$
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			
25			
		Summe	

Mittlere Anzahl Schritte beim Sondieren:

©2020 mathepunk

2.2.2 Verkettung, verkettete Listen

Beispiel für Hashtabelle mit Verkettung, verkettete Listen

j: Hashwert der Schlüssel in der verketteten Liste

n_{sj}: Anzahl Suchschritte in der verketteten Liste (für alle Datensätze der Liste)

j	**Schlüssel (s)**							**n_{sj}**
	S_{j1}		**S_{j2}**		**S_{j3}**		**S_{j4}**	
1	evanescent	→	imminent					3
2	eggs							1
3	breakable							1
4								
5	quack							1
6								
7	substance							1
8								
9								
10	continue	→	invincible	→	quarrelsome			6
11	pollution							1
12	committee							1
13	soap							1
14								
15								
16	remarkable							1
17								
18	industrious							1
19								
20	jagged							1
							Summe	19

Mittlere Anzahl Schritte beim Sondieren: $\frac{19}{15} = \underline{\mathbf{1.27}}$

©2020 mathepunk

Hashtabellen mit Verkettung, verkettete Listen

j: Hashwert der Schlüssel in der verketteten Liste

n_{sj}: Anzahl Suchschritte in der verketteten Liste (für alle Datensätze der Liste)

j	**Schlüssel (s)**								**n_{sj}**
	s_{j1}		**s_{j2}**		**s_{j3}**		**s_{j4}**		
1									
2									
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									
13									
14									
15									
16									
17									
18									
19									
20									
							Summe		

Mittlere Anzahl Schritte beim Sondieren:

©2020 mathepunk

j	Schlüssel (s)								n_{sj}
	S_{j1}		S_{j2}		S_{j3}		S_{j4}		
1									
2									
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									
13									
14									
15									
16									
17									
18									
19									
20									
21									
22									
23									
24									
25									
								Summe	

Mittlere Anzahl Schritte beim Sondieren:

©2020 mathepunk

Anhang

Englische Zufallswörter (k)

1	soap	41	habitual	81	closed	121	seat	161	tooth
2	continue	42	blow	82	jittery	122	post	162	flight
3	industrious	43	front	83	elderly	123	end	163	perpetual
4	pollution	44	dark	84	sincere	124	smelly	164	shop
5	breakable	45	houses	85	cemetery	125	nose	165	hammer
6	breakable	46	party	86	heavenly	126	thick	166	argument
7	pizzas	47	view	87	addition	127	moan	167	impolite
8	heartbreaking	48	amuck	88	baby	128	shock	168	puzzled
9	women	49	overflow	89	wink	129	square	169	pushy
10	substance	50	hypnotic	90	fearless	130	irritate	170	animal
11	remarkable	51	title	91	hole	131	quick	171	rabbits
12	invincible	52	scissors	92	reflect	132	exercise	172	slap
13	evanescent	53	lace	93	amount	133	mend	173	far
14	quack	54	legs	94	quarter	134	zephyr	174	pan
15	outstanding	55	whip	95	rejoice	135	lyrical	175	fireman
16	observant	56	zinc	96	airport	136	enter	176	stretch
17	far-flung	57	punish	97	muddle	137	wide	177	train
18	familiar	58	massive	98	small	138	channel	178	funny
19	elegant	59	thin	99	secret	139	alert	179	quiet
20	adjustment	60	drum	100	roasted	140	sparkling	180	tree
21	troubled	61	treat	101	terrific	141	gainful	181	fallacious
22	mellow	62	pot	102	rude	142	current	182	fence
23	whispering	63	saw	103	lame	143	crime	183	penitent
24	possessive	64	queue	104	able	144	happen	184	last
25	imminent	65	inject	105	quizzical	145	tip	185	well-to-do
26	quarrelsome	66	plucky	106	languid	146	close	186	name
27	eggs	67	start	107	ski	147	exchange	187	rich
28	jagged	68	slip	108	calculator	148	earth	188	unlock
29	committee	69	smart	109	crooked	149	decisive	189	reduce
30	present	70	copy	110	promise	150	grape	190	loving
31	guard	71	mine	111	swim	151	governor	191	frequent
32	uttermost	72	clean	112	dance	152	hanging	192	wrist
33	giraffe	73	tramp	113	eye	153	fluttering	193	hideous
34	belligerent	74	mint	114	guarded	154	glossy	194	vegetable
35	harmonious	75	spill	115	separate	155	bustling	195	twig
36	stupendous	76	hour	116	tedious	156	greasy	196	crook
37	development	77	milk	117	pig	157	volleyball	197	allow
38	damaged	78	practise	118	rat	158	entertain	198	powder
39	office	79	chance	119	scene	159	leather	199	train
40	fancy	80	anxious	120	trade	160	blush	200	drain

©2020 mathepunk