

Hash-Funktionen

- Funktion, die eine (beliebig große) Eingabemenge auf eine endliche Menge abbildet
- $h : K \rightarrow H$
 - K : Menge der Eingabewerte/Schlüssel
 - H : Menge der Hashwerte

- K : Beliebig lange Buchstabenfolgen/Wörter
- H : Zahlen von 0 bis 99 (willkürliche Festlegung)
- Funktion:
 - Handle jeden Buchstaben als Zahl (A=1, B=2 usw.)
 - Summiere alle Buchstaben des Eingabeworts
 - Rechne das Ergebnis modulo 100 (um nur noch Zahlen zwischen 0 und 99 zu erhalten)
- $h("HALLO") = (8 + 1 + 12 + 12 + 15) \bmod 100 = 48$

- Implementiere die zuvor beschriebene Hashfunktion als Methode in Java
 - Die Zeichenkette, deren Hashwert berechnet werden soll, soll als Parameter übergeben werden
 - Der berechnete Hashwert soll zurückgegeben werden
 - Eine Unterscheidung zwischen Groß- und Kleinschreibung soll nicht erfolgen

- Beispiel: Wörterbuch
 - Eingabe eines Wortes k
 - Bestimmung des Hashwerts $h(k)$
 - Am Index $h(k)$ befindet sich die Übersetzung des Wortes k
- Durch die Hashfunktion spart man sich das Durchsuchen aller Daten
- Problem: Was, wenn mehrere Eingabewörter den gleichen Hashwert haben?

- $h("HALLO") = (8 + 1 + 12 + 12 + 15) \bmod 100 = 48$
- $h("AUGEN") = (1 + 21 + 7 + 5 + 14) \bmod 100 = 48$
- Unterschiedliche Schlüssel können den gleichen Hashwert haben
 - Dies bezeichnet man als **Kollision**
- “Gute” Hashfunktionen versuchen, Kollisionen für erwartete Eingabewerte zu vermeiden
 - Gleichverteilung der Hashwerte wünschenswert

- Verwende Dein Programm aus Aufgabe 1, um für alle im Kurs vorkommenden **Vornamen** zu prüfen, ob Kollisionen auftreten

- Hashfunktion h mit zusätzlichen Bedingungen:
 - Zu einem gegebenen Hashwert x ist es praktisch unmöglich, den Eingabewert k zu finden, für den $h(k) = x$ gilt
 - Es ist praktisch unmöglich, zu einem gegebenen Eingabewert k einen weiteren Eingabewert k' zu finden, so dass $h(k) = h(k')$ gilt
 - Verschärfung: Es ist praktisch unmöglich, zwei Werte k, k' zu finden, so dass $h(k) = h(k')$ gilt

Verwendung kryptographischer Hashfunktionen

- Prüfsummen/Echtheitsnachweis
 - Eine Datei hat einen bestimmten Hashwert
 - Jede Veränderung an der Datei führt zu einem geänderten Hashwert
 - Kennt man den korrekten Hashwert, so kann man prüfen, ob man die richtige Datei erhalten hat
- Passwortspeicherung
 - Statt des Passworts im Klartext wird nur der Hashwert des Passworts gespeichert
 - Gibt jemand sein Passwort, so wird der Hashwert davon gebildet und mit dem gespeicherten Hashwert verglichen
 - Erlangt jemand unberechtigt Zugriff auf die Datenbank, so kann er dennoch nicht die gespeicherten Passwörter rekonstruieren
- Beispielfunktionen:
 - md5, sha