Informatik II: Algorithmen und Datenstrukturen SS 2013

Vorlesung 5a, Dienstag, 14. Mai 2013

(Hashtabellen, Universelles Hashing)

Prof. Dr. Hannah Bast
Lehrstuhl für Algorithmen und Datenstrukturen
Institut für Informatik
Universität Freiburg

UNI FREIBURG

Blick über die Vorlesung heute

UNI FREIBURG

- Organisatorisches
  - Ihre Erfahrungen mit dem Ü4 (FreeBase, MapCountingSort)
  - Treffen mit Ihrem Tutor / Ihrer Tutorin
- Hashtabellen
  - Eine mögliche Realisierung von einer Map
  - Dabei zentral: **universelles Hashing**
  - Beispiele für universelle Klassen von Hashfunktionen
  - Ü5, Aufgabe 1: Mittels eines Programms nachprüfen, ob eine Klasse von Hashfunktionen universell ist

    Ü5, Aufgabe 2: Cuckoo Hashing … kommt morgen dran

Erfahrungen mit dem Ü4 (FreeBase / MapSort)

UNI FREIBURG

- Zusammenfassung / Auszüge Stand 14. Mai 16:00
  - Wieder zeitaufwändiger … besonders RTL2 schauen
  - Problem: "unkatholischer" Aufbau der Datei married-to.tsv
  - Wusste nicht, dass Queen mit Salman Rushdie verheiratet
  - MapCountingSort nur für kleine m schneller, wenn überhaupt
  - Array von Map.entrySet() ist schon sortiert … ist mir neu
  - Tests stören mehr als dass sie nützen … Übungssache!
  - Weniger Fummelarbeit in den Übungen bitte
  - SVN fehlerfeindlich / add arbeitet nicht sauber … hmm?
  - Schlimmste Blatt bisher, keine gute Hilfe / Anleitung

    Es gibt das Forum + bei größeren Problemen Treffen mit Tutor

Erfahrungen mit dem Ü4

UNI FREIBURG

- Fortsetzung …
  - Vorlesung setzt zu viel Programmierkenntnisse bzw. –erfahrung voraus

    Das sollte eigentlich nicht so sein

    Bitte erklären, wo genau das Problem liegt, zumal:

    1. Ich mache in der Vorlesung viel vor + stelle den Code dann auf den Wiki

    2. Wenn man Ideen braucht um weiter zu kommen, gibt es auf dem Forum immer schnell Antwort dazu

Treffen mit Ihrem Tutor / Ihrer Tutorin

UNI FREIBURG

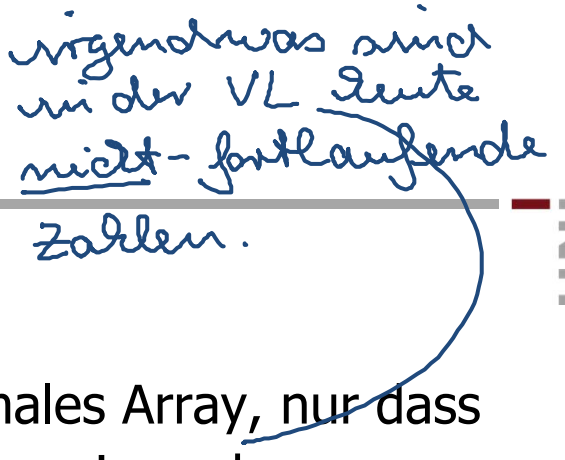
- Grund
  - Wir wollen Sie alle mal kennen lernen
  - Die meisten fanden das in der Vergangenheit gut
  - Gelegenheit für Fragen, die man sonst nicht stellt
  - Wir wollen auch schauen, dass es Sie wirklich gibt und Sie die Übungsblätter im Wesentlichen selber machen
- Vorgehen
  - Sie werden von Ihrem Tutor / Ihrer Tutorin angeschrieben
  - Treffen dauert ca. 30 Minuten
  - Ein Treffen pro Semester ist Pflicht, für alle !

Tipp für Windows-Benutzer

UNI FREIBURG

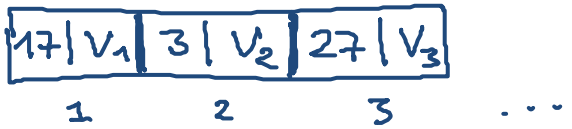
- Cygwin
  - Download unter www.cygwin.com
  - Dann haben Sie in Ihrer normalen DOS shell auch alle bekannten Unix/Linux Befehle
  - Insbesondere: cut, head, tail, less, more, sort, uniq, ...

Wie baut man eine Map?



- Zur Erinnerung
  - Ein assoziatives Array ist wie ein normales Array, nur dass die Indizes nicht 0, 1, 2, ... sind, sondern irgendwas
- Problem
  - Schnell ein Element mit einem bestimmten Schlüssel finden
  - Naive Lösung: Paare von Schlüsseln und Werten in einem normalen Feld (Java: ArrayList, C++: vector) speichern

    Array<KeyValuePair>
  - Bei n Schlüsseln kostet die Suche dann bis zu Θ(n) Zeit
  - Mit einer Hash Map geht es im günstigsten Fall in Zeit Θ(1) ... und zwar egal wie viele Elemente schon in der Map sind!



HashMap — Grundidee

UNI FREIBURG

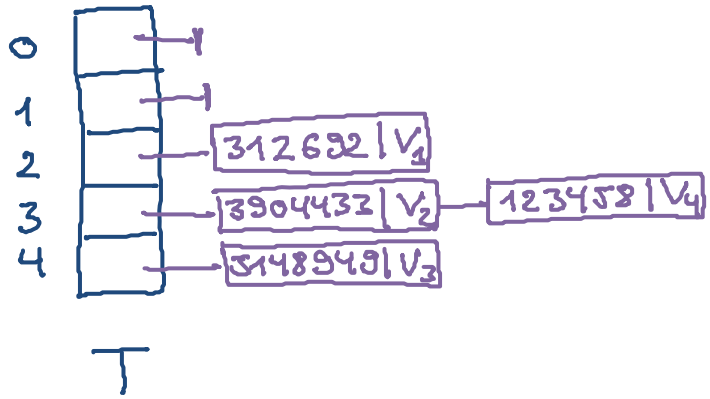
- Grundidee
  - Abbildung der Schlüssel auf die Indizes von einem normalen Feld, mit Hilfe einer sogenannten Hashfunktion
- Ein einfaches Beispiel
  - Schlüsselmenge { 312692, 3904433, 5148949 }
  - Hashfunktion $h(x) = x \bmod 5$, also Wertebereich [0..4]
    - $h(312692) = 2$, $h(3904433) = 3$, $h(5148949) = 4$
  - Ein gewöhnliches Feld T der Größe 5 (die Hashtabelle)
  - Wir speichern das Element mit Schlüssel x in $T[h(x)]$
  - In unseren Beispiel jetzt Zugriff in $\Theta(1)$ Zeit
  - Problem: zwei Schlüssel mit $x \neq y$ aber $h(x) = h(y)$
  - Das nennt man **Kollision**

HashMap — Kollisionen

$h_2(x) = x \bmod 5$

- Einfache Lösung
  - Jeder Eintrag der Hashtabelle kann nicht nur ein key-value Paar speichern, sondern eine Menge davon

    Array<Array<KeyValuePair>> hashTable;



insert (312692, V_1)
$h_2(312692) = 2$
insert (3904433, V_2)
$h_2(3904433) = 3$
insert (5148949, V_3)
$h_2(5148949) = 4$
lookup (3904433) = ?
$h_2(3904433) = 3$
→ V_2
insert (123458, V_4)
$h_2(123458) = 3$
lookup (854123) = ?
$h_2(854123) = 3$
→ gibt's nicht

HashMap — Kollisionen

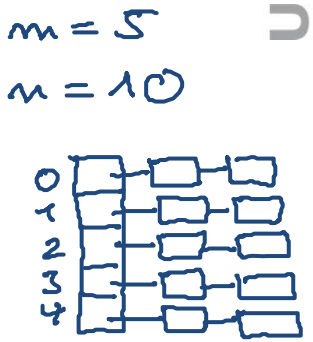
- Laufzeit für die Schlüsselsuche
  - Im besten Fall werden die Schlüssel gleichmäßig auf das Feld verteilt

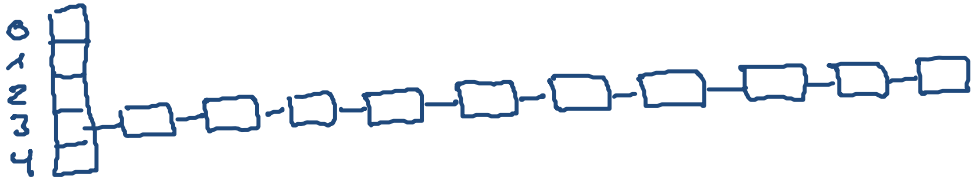
    Das sind ≈ n/m Schlüssel pro Eintrag

    n = #Schlüssel, m = Größe Hashtabelle

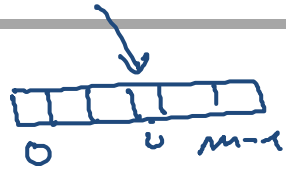
    Entsprechend Zeit Θ(n/m) pro Operation
  - Im schlechtesten Fall werden alle n Schlüssel auf denselben Eintrag der Hashtabelle abgebildet

    Dann Zeit Θ(n) pro Operation





Wahl der Hashfunktion 1/3



- Zufällige Schlüssel
  - Dann ist das einfache $h(x) = x \bmod m$ schon perfekt!
  - Denn für zwei zufällige Schlüssel x und y sind auch

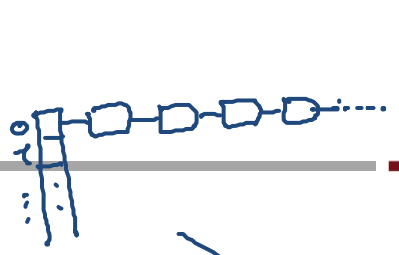
    $x \bmod m$ und $y \bmod m$ beide zufällig aus $\{0, \ldots, m-1\}$
  - Falls $x \neq y$ ist dann also $Pr(\, h(x) = h(y)\,) = 1/m$

$$Pr(\, h(x) = h(y)\,) = \sum_{i=0}^{m-1} Pr(\, h(x) = i \wedge h(y) = i\,)$$

$$= \sum_{i=0}^{m-1} \underbrace{Pr(\, h(x) = i\,)}_{1/m} \cdot \underbrace{Pr(\, h(y) = i\,)}_{1/m}$$

$$= m \cdot \frac{1}{m} \cdot \frac{1}{m} = \frac{1}{m} \quad \blacksquare$$

Wahl der Hashfunktion 2/3



UNI FREIBURG

- Nicht-zufällige Schlüssel
  - Dann kann h(x) = x mod m beliebig schlecht sein
  - Beispiel: m = 10 und 0, 10, 20, 30, 40, 50, 60, 70
  - Für alle x aus dieser Menge ist h(x) = x mod 10 = 0
  - Welche Hashfunktion soll man dann nehmen?
  - Für die Menge oben wäre h(x) = x mod 9 perfekt

    Aber für die Hashfunktion wäre 0, 9, 18, 27, … schlecht
  - Gibt es eine Hashfunktion, die für alle Schlüsselmengen gut ist?

Wahl der Hashfunktion 3/3

UNI FREIBURG

- Es kann nicht die eine Hashfunktion geben …

  … die für **alle** Schlüsselmengen gut ist !

  – Einfach weil jede Hashfunktion mit Wertebereich {0,…,m-1} unendlich viele Zahlen auf dasselbe abbilden muss

  – Umgekehrt gibt es aber für jede Schlüsselmenge viele Funktionen, die gut sind

  – Deshalb arbeitet man bei nicht-zufälligen Schlüsselmengen mit einer **Klasse von Hashfunktionen**, aus der man dann eine zufällig auswählt

  – Das nennt man **universelles Hashing**

Universelles Hashing 1/3

UNI FREIBURG

- Definition
  - Sei U die Menge der möglichen Schlüssel (Universum) und sei m die Größe der Hashtabelle
  - Sei H eine Menge von Hashfunktionen $U \rightarrow \{0, \ldots, m-1\}$
  - H ist c-universell wenn für alle $x, y \in U$ mit $x \neq y$ gilt:

    KOLLISION

    $$| \{h \in H : h(x) = h(y)\} | \leq c \cdot |H| / m$$
  - Mit anderen Worten, wenn $h \in H$ zufällig gewählt, dann

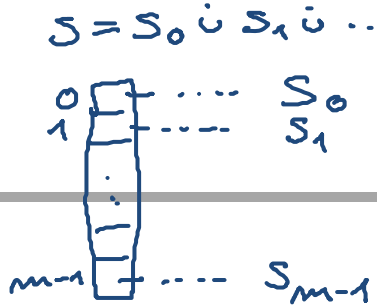
    $$\text{Prob}(h(x) = h(y)) \leq c \cdot 1 / m$$

    $c = 1$ wäre optimal (viele zufällige Schlüssel) aber schwierig
  - Bemerkung: wenn man x und y jede **zufällig** in eine der m "slots" der Hashtabelle schmeißt, dann

    $$\text{Prob}(\text{Kollision}) = 1 / m$$

    siehe vorherige Folie

Universelles Hashing 2/3



- Satz
  - Sei H eine c-universelle Klasse von Hashfunktionen
  - Sei S eine Menge von Schlüsseln und $h \in H$ zufällig gewählt
  - Sei S_i die Menge der Schlüssel x mit $h(x) = i$
  - Dann ist $E(|S_i|) \leq 1 + c \cdot |S| / m$ für alle i
  - Insbesondere: Falls $m = \Omega(|S|)$ gilt $E(|S_i|) = O(1)$

idealerweise $|S_i| = \frac{|S|}{m}$

Bevor wir das beweisen, ein kleiner Auffrisch- bzw. Crash- Kurs in Wahrscheinl.keitsrechnung

Beispiel dafür:
$m = 100$, $n = |S| = 100$
Ideal: $|S_i| = 100/100 = 1$
Nach Satz mit c=2: $E(|S_i|) \leq 1 + 2 \cdot \frac{100}{100} = 3$

z.B. $m = 10$
$n = |S| = 100$
Ideal wäre dann $|S_i| = \frac{100}{10} = 10$
Nach Satz z.B. mit c=2
$E(|S_i|) \leq 1 + 2 \cdot \frac{100}{10}$
$= 21.$

Einschub: Wahrscheinlichkeitsrechnung 1/3

UNI FREIBURG

- Wahrscheinlichkeitsraum / Ereignisse
  - Wir beschränken uns hier auf den diskreten Fall
  - Wahrscheinlichkeitsraum Ω von sog. Elementarereignissen
  - Die haben Wahrscheinlichkeiten ... Bedingung $\Sigma_{e \in \Omega} \Pr(e) = 1$
  - Ereignis E = Teilmenge von Ω, Wahrsch. $\Pr(E) = \Sigma_{e \in E} \Pr(e)$
  - Zum Beispiel: zweimal würfeln, dann $\Omega = \{1,...,6\}^2$

    (1,1) (1,2) ... (5,6) (6,6)

    Jedes e aus Ω hat dann Wahrscheinlichkeit $\Pr(e) = 1/36$

    E = beide Augenzahlen sind gerade, dann $\Pr(E) =$ $9/36 = 1/4$

    (2,2) (4,2) (6,2)
    (2,4) (4,4) (6,4)
    (2,6) (4,6) (6,6)

    9 Möglichkeiten

Einschub: Wahrscheinlichkeitsrechnung 2/3

UNI FREIBURG

- Zufallsvariable
  - ... weist einem Ausgang des Zufallsexperiments eine Zahl zu
  - Zum Beispiel: X = Summe Augenzahlen bei zweimal Würfeln
  - Sowas wie $X = 12$ oder $X \geq 7$ sind dann einfach Ereignisse
  - Beispiel 1: $\text{Prob}(X = 2) = \frac{1}{36}$ weil: (1,1)
  - Beispiel 2: $\text{Prob}(X = 4) = \frac{3}{36} = \frac{1}{12}$ weil: (1,3) (3,1) (2,2)
  - **Erwartungswert** ist definiert als $\mathbf{E}(X) = \Sigma\, k \cdot \Pr(X = k)$

    Intuitiv: gewichtetes Mittel der möglichen Werte von X, wobei die Gewichte die Wahrscheinlichkeiten der entspr. Werte sind

für X = Summe der Augenzahlen:

$$E(X) = 2 \cdot \underbrace{\Pr(X=2)}_{\frac{1}{36}} + 3 \cdot \underbrace{\Pr(X=3)}_{\frac{2}{36}} + \cdots + 12 \cdot \underbrace{\Pr(X=12)}_{\frac{1}{36}}$$

Einschub: Wahrscheinlichkeitsrechnung 3/3

- Summe von Erwartungswerten
  - Für beliebige (diskrete) Zufallsvariablen $X_1, \ldots, X_n$ gilt

    $\mathbf{E}(X_1 + \ldots + X_n) = \mathbf{E}(X_1) + \ldots + \mathbf{E}(X_n)$
  - Korollar: Bei einem Zufallsexperiment tritt das Ereignis E mit Wahrscheinlichkeit p auf. Sei X die Anzahl der Auftreten von E bei n Ausführungen dieses Experimentes, dann ist $\mathbf{E}(X) = n \cdot p$
  - Beispiel: $\mathbf{E}$(Anzahl Sechser bei 60 mal Würfeln) = 10
  - Beweis Korollar:

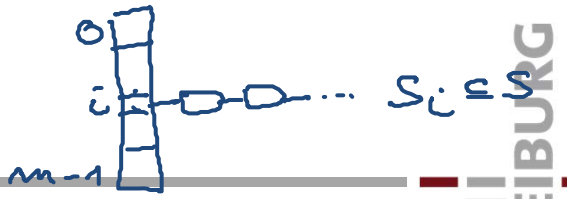
$X_i = 1$ gdw. bei der i-ten Ausführung hatte E ein
$X_i = 0$ sonst
X_i heißt INDIKATORVARIABLE

$X :=$ Anzahl Sechser $= \sum_{i=1}^{n} X_i$

$$E(X) = E\left(\sum_{i=1}^{n} X_i\right) = \sum_{i=1}^{n} E(X_i) = \sum_{i=1}^{n} \left(0 \cdot \underbrace{Pr(X_i = 0)}_{1-p} + 1 \cdot \underbrace{Pr(X_i = 1)}_{p}\right)$$

$$= n \cdot p \quad \blacksquare$$

Universelles Hashing 3/3



- Beweis von $E(|S_i|) \leq 1 + c \cdot |S| / m$ für alle i

Sei x ein beliebiges $i \in \{0, \ldots, m-1\}$ an

Fall 1: $S_i = \emptyset \Rightarrow |S_i| = 0$ prima $\Rightarrow |S_i| \leq 1 + c \cdot |S|/m$

Fall 2: $S_i \neq \emptyset \Rightarrow \exists x \in S_i$ d.h. $h(x) = i$

Für alle $y \in S \setminus \{x\}$ definiere $I_y = 1$ gdw. $h(y) = i$ (wegen $x \in S_i$)
$I_y = 0$ sonst

Dann $|S_i| = 1 + \sum_{y \in S \setminus \{x\}} I_y$

Also $E(|S_i|) = E(1 + \sum_{y \in S \setminus \{x\}} I_y) = 1 + \sum_{y \in S \setminus \{x\}} E(I_y)$

$\leq 1 + \sum_{y \in S \setminus \{x\}} c \cdot \frac{1}{m}$

$\leq 1 + c \cdot \frac{|S|}{m}$ ∎

$E(I_y) = 0 \cdot Pr(I_y = 0) + 1 \cdot Pr(I_y = 1)$
$= Pr(I_y = 1)$
$= Pr(h(x) = h(y))$
$\leq c \cdot \frac{1}{m}$ wegen c-universell

Klassen von Hashfunktionen 1/5

UNI FREIBURG

- Negativbeispiel 1
  - Die Menge aller h mit $h(x) = a \cdot x + b \bmod m$

    für $a, b \in U$

    Das heißt: um eine zufällige solche Hashfunktion zu wählen, wählt man einfach zufällig a und b aus U
  - Das sind $|U|^2$ mögliche Hashfunktionen, also viele
  - Aber trotzdem **nicht** universell

Klassen von Hashfunktionen 2/5

UNI FREIBURG

- Negativbeispiel 2
  - Die Menge aller Funktionen von $U \rightarrow \{1,\ldots,m\}$
  - Ist 1-universell
  - Aber als Klasse von Hashfunktionen ungeeignet

Klassen von Hashfunktionen 3/5

UNI FREIBURG

- Positivbeispiel 1
  - Sei p eine Primzahl mit $p > m$ und $p \geq u$, $U = \{0, \ldots, u - 1\}$
  - Sei H die Menge aller h mit $h(x) = (a \cdot x + b) \bmod p \bmod m$

    wobei $a, b \in U$
  - Die ist $\approx\mathbf{1}$-universell

    Siehe Exercise 4.11 in Mehlhorn/Sanders

Klassen von Hashfunktionen 4/5

UNI FREIBURG

- Positivbeispiel 2
  - Die Menge aller h mit $h(x) = a \bullet x \bmod m$, für ein $a \in U$
    - Schreibe $a = \Sigma_{i=0..k-1}\, a_i \cdot m^i$, wobei $k = \text{ceil}(\log_m |U|)$
    - Entsprechend $x = \Sigma_{i=0..k-1}\, x_i \cdot m^i$
    - Dann $a \bullet x := \Sigma_{i=0..k-1}\, a_i \cdot x_i$
    - Intuitiv: das "Skalarprodukt" der Darstellung zur Basis m
  - Die ist **1**-universell, siehe Theorem 4.4 in Mehlhorn/Sanders

Klassen von Hashfunktionen 5/5

UNI FREIBURG

- Positivbeispiel 3
  - Die Menge aller h mit $h(x) = a \cdot x \bmod 2^k \operatorname{div} 2^{k-\ell}$ für $a \in U$

    ... wobei $|U| = 2^k$, $m = 2^\ell$ ... in der Regel $k >> \ell$

    Das $\cdot$ ist hier wieder das normale Produkt

    Das heißt $a \cdot x$ gibt eine Zahl aus $0..|U|^2$

    Die lässt sich also in Binärdarstellung mit $2k$ Bits darstellen

    Eine Position in der Hashtabelle lässt sich mit ℓ Bits darstellen

    $h(x)$ ist dann einfach der Wert der Bits $k\text{-}l..k\text{-}1$ von $a \cdot x$
  - Diese Menge von Hashfunktionen ist **2**-universell

    Siehe Exercise 4.14 in Mehlhorn / Sanders

Histogramme 1/2

UNI FREIBURG

- Brauchen Sie für das Ü5, Aufgabe 1
  - Für jede der drei Klassen dort, berechnen Sie eine Liste von geschätzten Kollisions-Wahrscheinlichkeiten

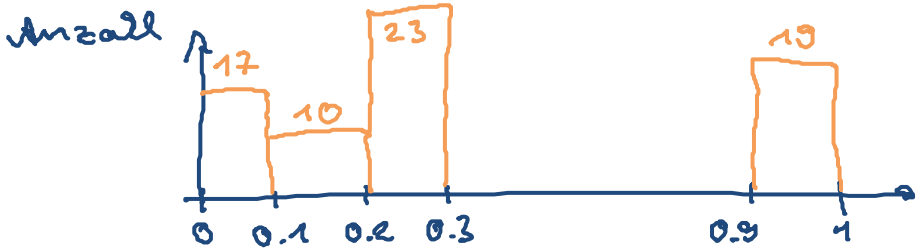
    Und zwar $u \cdot (u - 1) = 65\,280$ viele
  - Die visualisiert man am besten mit einem Histogramm:

    Werte $x_1, x_2, x_3, x_4, \ldots$ Wertebereich hier $[0,1]$

    Unterteile Wertebereich in n disjunkte Teil-Intervalle

    In unserem Fall hier kann man die gleich groß wählen

    Zähle für jedes Teil-Intervall I die Anzahl aller $x_i \in I$



Histogramme 2/2

UNI FREIBURG

- Wie malt man so ein Histogramm?
  - Anzahlen pro Bereich zeilenbasiert in eine Datei ausgeben

```
0.0     345
0.1     47
0.2     1234
...
```

  - Dann z.B. einfach mit gnuplot

```
set term png
set output "histogram.png"
plot "data.txt" using 1:2 with boxes
```

  - Geht aber auch mit R, S, Mathematica, Excel, …

Literatur / Links

UNI FREIBURG

- Universelles Hashing
  - In Mehlhorn / Sanders:

    4 Hash Tables and Associative Arrays
  - In Cormen / Leiserson / Rivest

    12 Hash Tables
  - In Wikipedia

    http://en.wikipedia.org/wiki/Universal_hashing