Informatik II: Algorithmen und Datenstrukturen SS 2015

Vorlesung 5b, Mittwoch, 20. Mai 2015

(Universelles Hashing Teil 2, Perfektes Hashing)

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

UNI FREIBURG

Blick über die Vorlesung heute

UNI FREIBURG

- Universelles Hashing, Teil 2
  - Universelle Klassen … zwei Negativ- und drei Positivbeispiele
  - Worst-Case Komplexität … gestern: average-case
  - Perfektes Hashing … noch eine Alternative
  - Histogramme … brauchen Sie für das Übungsblatt
  - Ü5 Aufgabe 1 – 3: Prüfen Sie für jede der fünf Klassen empirisch nach, ob sie universell sind

    Ü5 Aufgabe 4: Entwerfen Sie einen **eigenen** Algorithmus, der die kleinste Primzahl ≥ ein gegebenes m findet

    Diesmal mit **zwei** interessanten Zusatzaufgaben, und Sie haben **zwei** Wochen Zeit … das Blatt hat aber normalen Umfang

Klassen von Hashfunktionen 1/7

- Negativbeispiel 1

  z.B. $h(x) = (3 \cdot x + 7) \bmod 10$

  $m = 10$

  – Die Menge H aller h mit $h(x) = (a \cdot x + b) \bmod m$

    wobei a und b jeweils aus $\{0, \ldots, m-1\}$

    Das sind m^2 mögliche Hashfunktionen

  – Aber trotzdem **nicht** universell

Universell heißt:

$\forall x, y \in U \; x \neq y : |\{h \in H : h(x) = h(y)\}| \leq c \cdot \frac{|H|}{m}$

Gegenbeispiel: (ein Schlüsselpaar reicht)

$x = m, \; y = 2m$ (dann ist auch $x \neq y$)

$h(x) = (a \cdot m + b) \bmod m = b$

$h(y) = (a \, 2m + b) \bmod m = b$

$\Rightarrow \forall h \in H : h(x) = h(y) \Rightarrow |\{h : h(x) = h(y)\}| = |H|$

Klassen von Hashfunktionen 2/7

- Negativbeispiel 2
  - Die Menge H **aller** Funktionen von $U \to \{0, \ldots, m-1\}$
  - Für eine zufällige Funktion $h \in H$ ist dann für jedes $x \in U$ $h(x)$ zufällig aus $\{0, \ldots, m-1\}$

    Eine zufällige Funktion h aus H wählen und eine Menge S damit abbilden ist also wie zufälliges Werfen
  - Für den Fall haben wir schon gesehen:

    $\Pr(h(x) = h(y)) = 1/m$
  - Die Klasse ist also 1-universell

    Besser geht es nicht, warum dann Negativbeispiel ?

Klassen von Hashfunktionen 3/7

- Negativbeispiel 2, Fortsetzung
  - Als Klasse von Hashfunktionen trotzdem ungeeignet
  - Die Funktionen haben keine "kompakte" Form

    So wie zum Beispiel $(a \cdot x + b) \bmod m$
  - Um eine Funktion h aus H zu repräsentieren, müsste man für jedes x aus U speichern, was h(x) ist

    Das braucht $\Theta(|U|)$ Platz

    Dann kann man auch gleich die triviale Realisierung aus Vorlesung 4a (Folie 13) nehmen

Klassen von Hashfunktionen 4/7

- Positivbeispiel 1

  in der Regel $p \gg m$

  – Sei p eine Primzahl mit $p > m$ und $U = \{0, \ldots, p - 1\}$

  – Sei H die Menge aller h mit $h(x) = (a \cdot x + b) \bmod p \bmod m$

  wobei $a, b \in \{0, \ldots, p - 1\}$

  Es ist entscheidend, dass $ggT(p, m) = 1$

  – Diese Menge von Hashfunktionen ist **≈1**-universell

  Der Beweis würde hier zu weit führen, siehe Exercise 59 in Mehlhorn/Sanders

  Man kann sich aber einfach klarmachen, wie das mod p vor dem mod m die Probleme vom Negativbeispiel 1 verhindert

  z.B $m = 10$, $p = 101$ (wäre schlecht für N-Beispiel 1)

  $x = 80$, $y = 30$

  $a = 2$, $b = 17$

  $h(x) = (2 \cdot 80 \overset{\equiv 177}{+ 17}) \bmod 101 \bmod 10 = 6$

  $h(y) = (2 \cdot 30 \underset{\equiv 77}{+ 17}) \bmod 101 \bmod 10 = 7$

  $\neq$ (gut!)

Klassen von Hashfunktionen 5/7

- Positivbeispiel 2
  - Die Menge aller h mit $h(x) = a \bullet x \bmod m$

    wobei $a \in U$ und m eine **Primzahl** sein muss

    Ü5, Aufgabe 4 beschäftigt sich damit, wie man für eine gewünschte Größe m eine Primzahl in der Nähe bekommt
  - Die Operation $\bullet$ ist dabei wie folgt definiert:

    Schreibe $a = \Sigma_{i=0..k-1}\, a_i \cdot m^i$, wobei $k = \lceil \log_m |U| \rceil$

    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

Klassen von Hashfunktionen 6/7

- Positivbeispiel 2, Fortsetzung
  - Diese Klasse ist sogar **1**-universell

    Beweis siehe Theorem 12 in Mehlhorn/Sanders …
  - Beispielrechnung: $m = 11$, $U = \{0, \ldots, \text{was groß}\}$

    $a = 274$, $h_a(x) = (215 \cdot x) \bmod 11$

    $x = 47$

    wir schreiben jetzt a und x als Zahlen zur Basis $m = 11$

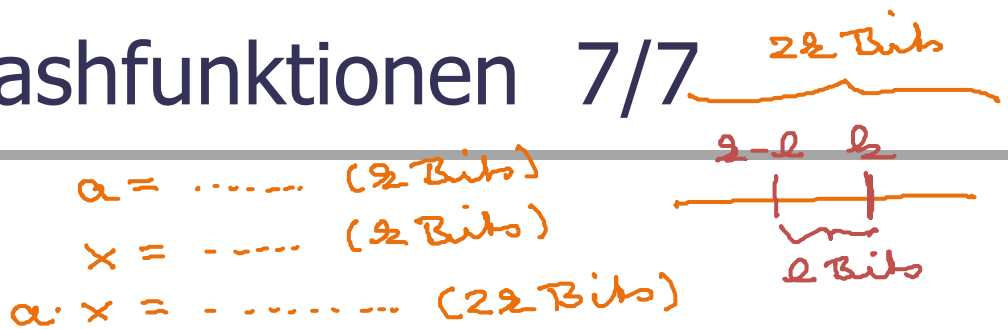
    $a = 2 \cdot 11^2 + 2 \cdot 11^1 + 10 \cdot 11^0 = (2\ 2\ 10)_{11}$ (242 = 121, 22 = 11, 10 = 1; 242 zur Basis 11)

    $x = 0 \cdot 11^2 + 4 \cdot 11^1 + 3 \cdot 11^0 = (0\ 4\ 3)_{11}$ (0 = 121, 44 = 11, 3 = 1)

    $h_a(x) = a \cdot x \bmod m = (2 \cdot 0 + 2 \cdot 4 + 10 \cdot 3) \bmod 11$ (38)

    $= 5$

Klassen von Hashfunktionen 7/7



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

    ↑ Integer Division = ohne Rest

    wobei $U = \{0, \ldots, 2^k - 1\}$, $a \in U$ **ungerade** und $m = 2^\ell$

    Das · ist hier wieder das normale Produkt

    Das heißt a · 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-ℓ..k-1 von a · x
  - Diese Menge von Hashfunktionen ist **2**-universell

    Siehe Exercise 62 in Mehlhorn / Sanders

    << und >>

    Im Code Bitverschiebeoperationen verwenden

Worst-Case Komplexität 1/3

- Bisher: Average-Case Komplexität
  - Wenn h aus einer universellen Klasse gewählt wird, dann ist

    $E(|S_i|) \leq 1 + c \cdot |S| / m$

    Insbesondere, wenn $m = \Theta(|S|)$

    $E(|S_i|) = O(1)$
  - Sei i_{max} die Position, auf die die meisten Schlüssel abgebildet werden, gilt dann auch

    $E(|S_{imax}|) = O(1)$?
  - Im Allgemeinen: $E(\max\{X_1, \ldots, X_n\}) \neq \max\{E(X_1), \ldots, E(X_n)\}$

    E und Summe kann man vertauschen, E und max nicht

Worst-Case Komplexität 2/3

- Zufälliges Werfen
  - Nehmen wir an, wir werfen zufällig n Bälle in n Kisten

    Besser kann eine zufällige Hashfunktion nicht verteilen
  - Sei S_i die Menge der Bälle in der i-ten Kiste und sei i_{max} der Index der Kiste mit den meisten Bällen
  - Dann ist $E(|S_i|) = 1$
  - Aber auch in dem Fall **nicht** $E(|S_{imax}|) = 1$

    Dazu schreiben wir gerade ein kleines Programm …

Worst-Case Komplexität 3/3

- Satz
  - Nehmen wir an, wir werfen zufällig n Bälle in n Kisten und seien S_i und i_{max} wie auf der Folie vorher

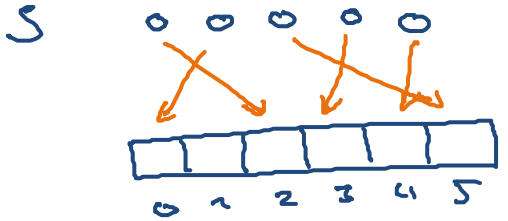
    Dann ist $|S_{imax}| = O(\log n / \log \log n)$ mit hoher W-keit
  - Wenn man n Bälle in m Kisten wirft, mit $n \geq m \cdot \log m$

    Dann ist $|S_{imax}| = \Theta(n / m)$ mit hoher W-keit

    Der Beweis würde hier zu weit führen ... nur kurz, wie der Term log n / log log n überhaupt zustande kommt:

    $k^k = n \Rightarrow k \approx \log n / \log \log n$

Perfektes Hashing 1/4



- Vorbetrachtung
  - Gegeben eine Schlüsselmenge S
  - Die Größe der Hashtabelle m sei ≥ |S|
  - Dann gibt es (sogar viele) Hashfunktionen h, die S **injektiv** auf {0, …, m – 1} abbilden

    Also ohne irgendeine Kollision (= perfekt)

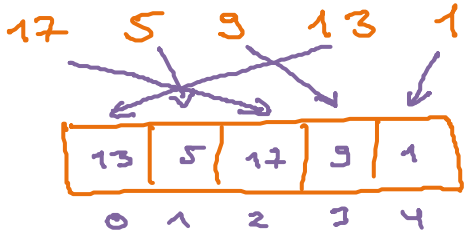
    Allerdings kann man diese Funktionen nicht unbedingt in wenig Platz speichern … siehe Negativbeispiel 2

Perfektes Hashing 2/4

UNI FREIBURG

- Definition
  - Gegeben eine Schlüsselmenge S
  - Die Größe der Hashtabelle m sei $\geq |S|$
  - Eine Hashfunktion h heißt **perfekt** für S, wenn gilt:
    - h bildet S injektiv auf $\{0, \ldots, m-1\}$ ab
    - h kann in $O(m)$ Platz gespeichert werden
    - $h(x)$ kann für alle x in $O(1)$ ausgewertet werden

Perfektes Hashing 3/4



- Beispiel
  - Sei $S = \{17, 5, 9, 13, 1\}$ und $m = 5$
  - Dann wäre folgende Hashfunktion perfekt:

$$h_2(x) = (3 \cdot x + 1) \bmod 5$$

Kann man immer so eine einfache Funktion finden ?

So einfach nicht immer, aber einfach genug !

Perfektes Hashing 4/4

- Satz
  - Sei S eine beliebige Schlüsselmenge und $m \geq 2 \cdot |S|$

    Dann gibt es eine perfekte Hashfunktion $S \rightarrow \{0, \ldots, m-1\}$

    Und man kann sie in $O(|S|)$ Zeit finden

    Der Beweis würde hier zu weit führen, siehe:

    Storing a Spare Table with O(1) Worst Case Access Time
    Fredman, Komlós, Szemerédi
    Journal of the ACM, Vol 31, No 3, **1984**

Histogramme 1/3

- Brauchen Sie für das Übungsblatt
  - Für jede der fünf Klassen H von heute, berechnen Sie eine Liste von geschätzten Kollisions-Wahrscheinlichkeiten

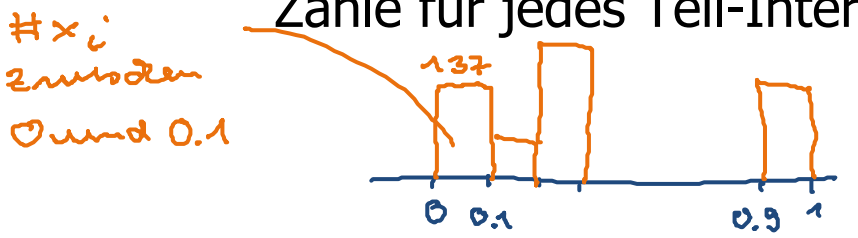
    Und zwar $u \cdot (u - 1) = 65\,280$ viele
  - Die visualisiert man am besten mit einem Histogramm:

    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 2/3

- Erstellen der Daten
  - Zeilenbasiert in eine Datei ausgeben

```
0.0     120
0.1     47
0.2     88
...
```

Erste Spalte = x-Achse, zweite Spalte = y-Achse

Histogramme 3/3

- Wie malt man so ein Histogramm?
  - Zum Beispiel mit **gnuplot**

```
set term pdf                              Ausgabe als PDF
set output "data.pdf"                     Ausgabedatei
set style fill solid 0.4                  Gefüllte Boxen
plot [] [0:200] "data.txt" with boxes     Malen

plot … lt rgb "orange"                    Andere Farbe
```

  - Das geht aber auch mit vielen anderen Programmen, z.B.

S … oder die open-source Variante R
Matlab … oder die open-source Variante Octave
Mathematica
Excel

Literatur / Links

- Universelle Klassen von Hashfunktionen
  - In Mehlhorn / Sanders:

    4 Hash Tables and Associative Arrays