Randomisierte Algorithmen

13. Hashing

Thomas Worsch

Fakultät für Informatik
Karlsruher Institut für Technologie

Wintersemester 2018/2019

Überblick

Ausgangspunkt

- Universum M der Kardinalität $|M| = m$
- Untermenge $S \subseteq M$ von Schlüsseln
- Hashtabelle T, adressiert mit Indizes aus N mit $|N| = n < m$
- Hashfunktion $h : M \to N$
- Information zu Schlüssel $s \in S$ in Eintrag $T[h(s)] = (s, \ldots)$

Definition

- Hashfunktion *perfekt* für Schlüsselmenge S, falls für alle $x, y \in S$ gilt: $x \neq y \implies h(x) \neq h(y)$.
- Kollision:
  wenn $|S| > |N|$, dann gibt es $x \neq y$ mit $h(x) = h(y)$,
  perfekte Hashfunktion unmöglich
- $|M| > |N|$: keine Hashfunktion kann für alle Schlüsselmengen perfekt sein
- Kollisionsauflösung z. B.:
  - speichere alle Schlüssel mit gleichem Hashwert in verketteter Liste
  - bilde kollidierende Schlüssel mit einer weiteren Hashfunktion auf jeweils eine (kleinere) zweite Hashtabelle ab
  - … oder eine von vielen anderen Möglichkeiten …

Anwendungen universeller Familien von Hashfunktionen

- randomisierte Algorithmen: *zufällige Wahl von $h \in H$*
  $\leadsto$ evtl. etwa Gleichverteilung der Schlüssel auf Tabelleneinträge
- Einsparung von Zufallsbits: *Derandomisierung*

Überblick

Definition

- Familie H von Hashfunktionen $h : M \longrightarrow N$ heißt *2-universell*, wenn für alle $x, y \in M$ mit $x \neq y$ gilt: Für zufällig gleichverteilt aus H gewähltes h ist $\mathbf{P}(h(x) = h(y)) \leq 1/n$.

M. a. W. ist für $x \neq y$

$$\frac{|\{h \mid h(x) = h(y)\}|}{|H|} \leq \frac{1}{n}$$

NB: $1/n$ auch die Wahrscheinlichkeit für $h(x) = h(y)$, falls Funktionswerte unabhängig zufällig gewählt.

Beispiel

- Die Menge $H = N^M$ *aller* Hashfunktionen ist 2-universell.
- Universalität: $\mathbf{P}(h(x) = h(y)) = 1/n$, denn
  zu jedem h mit $h(x) = h(y)$ gibt es $n - 1$ Hashfunktionen,
  die sich von h nur an der Stelle y unterscheiden.
- Nachteile:

Beispiel

- Die Menge $H = N^M$ *aller* Hashfunktionen ist 2-universell.
- Universalität: $\mathbf{P}(h(x) = h(y)) = 1/n$, denn
  zu jedem h mit $h(x) = h(y)$ gibt es $n - 1$ Hashfunktionen,
  die sich von h nur an der Stelle y unterscheiden.
- Nachteile:
  - Man benötigt „viele“ Zufallsbits: $\Theta(m \log n)$,
    um zufällig gleichverteilt ein $h \in H$ auszuwählen
  - unklar, ob jedes h „leicht“ berechenbar.

Definition

bequeme Notation zum Zählen von Kollisionen:
definiere für $X, Y \subseteq M$:

$$\begin{aligned}
\delta(x, y, h) &= \begin{cases} 1 & \text{falls } x \neq y \wedge h(x) = h(y) \\ 0 & \text{sonst} \end{cases} \\
\delta(x, y, H) &= \sum_{h \in H} \delta(x, y, h) \\
\delta(x, Y, h) &= \sum_{y \in Y} \delta(x, y, h) \\
\delta(X, Y, h) &= \sum_{x \in X} \delta(x, Y, h) \\
\delta(X, Y, H) &= \sum_{h \in H} \delta(X, Y, h)
\end{aligned}$$

13.6 Beobachtung

- wenn H 2-universell,
  dann für $x \neq y$ stets $\delta(x, y, H) \leq |H| / n$
  - andernfalls bei zufälliger Wahl eines $h \in H$
    $\mathbf{P}(h(x) = h(y)) > 1/n$

13.7 Satz

Für jede Familie H von Funktionen $h : M \longrightarrow N$ existieren $x, y \in M$ so, dass $\delta(x, y, H) \geq |H| / n - |H| / m$.

m sehr groß: obere und untere Schranken nahe beieinander

Rechnung

Rechnung

- Es seien k und g zwei positive ganze Zahlen
- $A(k,g) = k(k-1) + g(g-1)$.
- Für $k \leq g-1$ gilt dann $A(k,g) \geq A(k+1, g-1)$, denn:

$$
\begin{aligned}
& A(k,g) - A(k+1,g-1) \\
= \;& k(k-1) + g(g-1) - \big((k+1)k + (g-1)(g-2)\big) \\
= \;& k^2 - k + g^2 - g - \big(k^2 + k + g^2 - 3g + 2\big) \\
= \;& -k - g - k + 3g - 2 \\
= \;& 2(g-1-k) \\
\geq \;& 0\ .
\end{aligned}
$$

Beweis von Satz 13.7

- zunächst $h \in H$ beliebig
- für $z \in N$ sei $A_z = \{x \in M \mid h(x) = z\}$.
- für $w, z \in N$ gilt:

$$\delta(A_w, A_z, h) = \begin{cases} 0 & \text{falls } w \neq z \\ |A_z|\,(|A_z| - 1) & \text{falls } w = z \end{cases},$$

da alle Paare (x, y) mit $x \neq y$ genau 1 zur Summe beitragen.

Beweis (2)

- Gesamtzahl der Kollisionen $\delta(M, M, h) = \sum_{z \in N} |A_z| \, (|A_z| - 1)$
- minimal, wenn die A_z gleich groß
- also:

$$\delta(M, M, h) \geq \sum_{z \in N} \frac{m}{n} \left(\frac{m}{n} - 1 \right) = n \frac{m}{n} \left(\frac{m}{n} - 1 \right) = m^2 \left(\frac{1}{n} - \frac{1}{m} \right)$$

- also $\delta(M, M, H) \geq |H| \, m^2 \, (1/n - 1/m)$
- Schubfachprinzip: es gibt $x, y \in M$ mit

$$\delta(x, y, H) \geq \delta(M, M, H)/m^2 \geq |H| \, (1/n - 1/m) \ .$$

Überblick

O. B. d. A.

- Es sei nun $M = \{0, \ldots, m-1\}$ und $N = \{0, \ldots, n-1\}$.
- Es wird auch noch eine Primzahl $p \geq m$ benötigt.
  Wie klein kann sie gewählt werden?
- „Bertrands Postulat“
  einfacher Beweis (2 Seiten) von Ramanujan:
  `www.imsc.res.in/~rao/ramanujan/CamUnivCpapers/Cpaper24/page1.htm`

Satz (Chebyshev)

Zu jedem $m > 1$ gibt es eine Primzahl p mit $m \leq p < 2m$.

Definition

- Es sei $m \geq n$ und $p \geq m$ eine Primzahl.
- Familie $H = \{h_{a,b} \mid a, b \in \mathbb{Z}_p \wedge a \neq 0\}$ von Hashfunktionen wie folgt definiert:

$$
\begin{aligned}
& & h_{a,b}(x) &= g(f_{a,b}(x)) \\
\text{mit} & & f_{a,b} : \mathbb{Z}_p &\to \mathbb{Z}_p \\
& & x &\mapsto ax + b \bmod p \\
& & g : \mathbb{Z}_p &\to N \\
& & x &\mapsto x \bmod n
\end{aligned}
$$

- benutzt: Einschränkung der $h_{a,b} : \mathbb{Z}_p \to N$ auf $M \subseteq \mathbb{Z}_p$
- dadurch kann Zahl der Kollisionen nicht größer werden

Satz

Die eben definierte Familie H ist 2-universell.

Beweis

- Zeige: bei zufällig gewähltem $h_{a,b} \in H$ ist $\mathbf{P}\big(h_{a,b}(x) = h_{a,b}(y)\big) \leq 1/n$.
- m. a. W. $\quad \delta(x, y, H) \leq |H| / n$.
- Beweis in zwei Schritten:
  1. Zeige, dass für $x \neq y$ gilt: $\delta(x, y, H) = \delta(\mathbb{Z}_p, \mathbb{Z}_p, g)$.
  2. Schätze $\delta(\mathbb{Z}_p, \mathbb{Z}_p, g)$ nach oben ab.

Beweis (2)

1. Schritt: Zeige, dass für $x \neq y$ gilt: $\delta(x, y, H) = \delta(\mathbb{Z}_p, \mathbb{Z}_p, g)$.
   - Es sei $h_{a,b}(x) = h_{a,b}(y)$, also $g(f_{a,b}(x)) = g(f_{a,b}(y))$.
   - $a \neq 0$ und p prim $\implies$ $f_{a,b}$ bijektiv $\implies$ $r = f_{a,b}(x) \neq f_{a,b}(y) = s$.
   - Kollision: wenn $g(r) = g(s)$, d. h. $r \equiv s \mod n$.

Beweis (2)

1. Schritt: Zeige, dass für $x \neq y$ gilt: $\delta(x, y, H) = \delta(\mathbb{Z}_p, \mathbb{Z}_p, g)$.
   - Es sei $h_{a,b}(x) = h_{a,b}(y)$, also $g(f_{a,b}(x)) = g(f_{a,b}(y))$.
   - $a \neq 0$ und p prim $\implies$ $f_{a,b}$ bijektiv $\implies$ $r = f_{a,b}(x) \neq f_{a,b}(y) = s$.
   - Kollision: wenn $g(r) = g(s)$, d. h. $r \equiv s \mod n$.
   - Umgekehrt legen Werte $x \neq y$ und $r \neq s$ *eindeutig* ein $h_{a,b}$ mit $f_{a,b}(x) = r$ und $f_{a,b}(y) = s$ fest, denn es muss das folgende LGS über $\mathbb{Z}_p$ erfüllt werden:

$$\begin{aligned} ax + b &\equiv r \pmod p \\ ay + b &\equiv s \pmod p \end{aligned}$$

   - Also ist die Zahl $\delta(x, y, H)$ der h mit Kollision $h(x) = h(y)$ gleich der Anzahl $\delta(\mathbb{Z}_p, \mathbb{Z}_p, g)$ von Paaren (r, s) mit $r \neq s$ und $r \equiv s \mod n$.

Beweis (3)

2. Schritt: Wieviele solche Paare gibt es höchstens?

- Für $z \in N$ sei $A_z = \{x \in \mathbb{Z}_p \mid g(x) = z\}$.
- Für jedes z ist $|A_z| \leq \lceil p/n \rceil$.
- Also gibt es zu jedem r höchstens $\lceil p/n \rceil$ Werte s so, so dass (r, s) obiges Gleichungssystem löst.
- Also ist (wegen $r \neq s$)

$$\delta(\mathbb{Z}_p, \mathbb{Z}_p, g) \leq p\left(\left\lceil \frac{p}{n} \right\rceil - 1\right) = p\left(\left\lfloor \frac{p-1}{n} \right\rfloor + 1 - 1\right) \leq \frac{p(p-1)}{n} = \frac{|H|}{n}\,.$$

Definition

noch eine 2-universelle Familie von Hashfunktionen:

- sei n prim (falls nicht, vergrößere Hashtabelle)
- Schlüssel x werde dargestellt als Folge $\langle x_0, x_1, \ldots, x_r \rangle$ von $r + 1$ Werten x_i mit $0 \leq x_i \leq n - 1$.
- Hashfunktion h_a durch $a = \langle a_0, a_1, \ldots, a_r \rangle$ festgelegt
- definiere

$$h_a(x) = \sum_{i=0}^{r} a_i x_i \bmod n$$

- Die interessierende Familie von Hashfunktionen ist die Menge aller n^{r+1} solchen h_a.

Satz

Die eben festgelegte Familie von Hashfunktionen ist 2-universell.

13.16 Beweis

- Es sei $h_a(x) = h_a(y)$ und $x \neq y$.
- o. B. d. A. etwa $x_0 \neq y_0$ ($x_i \neq y_i$ für ein $i > 0$ analog)
- Es seien $a_1, \ldots, a_r$ beliebig aber fest. Dann

$$a_0(x_0 - y_0) = -\sum_{i=1}^{r} a_i(x_i - y_i) \bmod n \ .$$

- Da n prim ist, ist $x_0 - y_0$ invertierbar modulo n. Also ist a_0 eindeutig bestimmt.
- Also kollidieren zwei Werte x und y für genau n^r der n^{r+1} möglichen a bzw. h_a.
- Also ist die Kollisionswahrscheinlichkeit gerade $1/n$.

Überblick

13.17 Aufgabenstellung

- bearbeite Folge von Anfragen FIND(x_1), FIND(x_2) und prüfe, ob x_i in Menge S von Schlüsseln eingetragen ist.
- *dynamisches Wörterbuchproblem:*
  S ergibt sich durch Operationen INSERT(x) und DELETE(x), die zwischen die FIND-Operationen eingestreut sind.
- *statisches Wörterbuchproblem:*
  S liegt von vorne herein fest.

13.18 Algorithmus für das dynamische Wörterbuchproblem

- Beginn:
  - wähle h zufällig aus 2-universeller Familie H
  - benutze h bei der Abarbeitung der Liste $R = R_1, R_2, \ldots, R_r$ von Operationen
- $R_j = \textsc{insert}(x)$:
  - füge x in der Hashtabelle an Stelle $h(x)$ ein
  - bei Kollision wird lineare verkette Liste aufgebaut
- $R_j = \textsc{find}(x)$:
  - suche in der Hashtabelle an Stelle $h(x)$ (bzw. ggf. in der dort angehängten linearen Liste) nach x

13.19

- Ist S die Menge der zu einem Zeitpunkt gespeicherten Schlüssel und
- soll dann ein weiterer Schlüssel x eingefügt werden,
- so ist $\delta(x, S, h)$ die Länge der linearen Liste, in die x eingefügt wird.

13.20 Lemma

Für alle $x \in M$ und $S \subseteq M$ und ein zufällig gewähltes h aus einer 2-universellen Familie H gilt:

$$\mathbf{E}\left[\delta(x, S, h)\right] \leq \frac{|S|}{n} \ .$$

13.21 Beweis

$$
\begin{aligned}
\mathbf{E}\left[\delta(x,S,h)\right] &= \sum_{h\in H} \frac{\delta(x,S,h)}{|H|} \\
&= \frac{1}{|H|} \sum_{h\in H} \sum_{y\in S} \delta(x,y,h) \\
&= \frac{1}{|H|} \sum_{y\in S} \delta(x,y,H) \\
&\leq \frac{1}{|H|} \sum_{y\in S} \frac{|H|}{n} \\
&= \frac{|S|}{n}\,.
\end{aligned}
$$

Dabei rührt das „$\leq$“ von der Überlegung in Punkt 13.6 her.

13.22 Satz

Der Erwartungswert für den Gesamtzeitaufwand $t(R, h)$ zur Abarbeitung einer Liste R mit s INSERT-Operationen bei zufällig gewähltem $h \in H$ ist

$$\mathbf{E}\left[t(R,h)\right] \leq r\left(1+\frac{s}{n}\right) \ .$$

13.23 Beweis

- $\mathbf{E}[t(R,h)]$ ist kleiner oder gleich dem r-fachen des maximal zu erwartenden Zeitbedarfs für die Bearbeitung einer Operation.
- Dieser ist im wesentlichen beschränkt durch den Erwartungswert für die Länge der jeweils zu untersuchenden linearen Liste, der nach Lemma 13.21 maximal s/n ist.

13.24 Bemerkung

- Falls man $n \geq s$ wählen kann, dann ist obiger Erwartungswert durch 2 beschränkt.
- Aber u.U. immer noch Aufwand proportional zu s pro FIND-Operation

Vorbereitungen für das statische Wörterbuchproblem

- $h = h_{a,b} = ax + b$ zufällig gewählt (siehe Def. 13.11).
- Erwartungswert der Anzahl Kollisionen bei Schlüsselmenge S und Tabellengröße n:

$$E = \sum_{x \neq y \in S} \mathbf{P}(h(x) = h(y)) \leq \binom{|S|}{2} \cdot \frac{1}{n}$$

- Falls $n = |S| : E \leq \frac{n}{2}$, also
  Wahrscheinlichkeit für $\geq |S|$ Kollisionen ist $\leq 1/2$
- Falls $n = |S|^2 : E \leq \frac{1}{2}$, also
  h mit Wahrscheinlichkeit $\geq 1/2$ injektiv.

Datenstrukturen für nachfolgenden Algorithmus

zweistufiges Hashen:

- primäre Hashtabelle N
  - $N[i]$ speichert Parameter für h_i und
  - Verweis auf sekundäre Hashtabelle N_i
- sekundäre Hashtabellen N_i geeigneter Größe k_i^2

man wird sehen:

- Gesamtspeicherbedarf proportional zur Anzahl Schlüssel

Algorithmus (1)

$N \leftarrow \langle$*Hashtabelle der Größe* $n = |S|\rangle$
$H \leftarrow \langle$*2-universelle Hashfamilie für* $N\rangle$
$\langle$*1. Phase: suche primäre Hashfunktion*$\rangle$
repeat
    $h \leftarrow \langle$*zufällig aus* H *gewählt*$\rangle$
    for $i \leftarrow 0$ **to** $n - 1$ **do**
        $S_i \leftarrow \emptyset$
        $k_i \leftarrow 0$
    od
    for each $s \in S$ **do**
        $\langle$*add* s *to* $S_{h(s)}\rangle$
        $k_{h(s)} \leftarrow k_{h(s)} + 1$
    od
until $\sum_{i<n} \binom{k_i}{2} < n$ $\langle$*Anzahl Kollisionen kleiner* $n\rangle$

Algorithmus (2)

⟨*2. Phase: suche sekundäre injektive Hashfunktionen*⟩
for $i \leftarrow 0$ **to** $n-1$ **do**
    $N_i \leftarrow$ ⟨*Hashtabelle der Größe* k_i^2⟩
    $H_i \leftarrow$ ⟨*Hashfamilie für Tabelle mit* k_i^2 *Einträgen*⟩
    repeat
        $h_i \leftarrow$ ⟨*zufällig aus* H_i *gewählt*⟩
    until ⟨h_i *ist auf* S_i *injektiv*⟩
od

Analyse des Algorithmus

1. Phase:
   - Erwartungswert für Anzahl Durchläufe durch **repeat**-Schleife ist 2 (13.26)
   - Erwartungswert für Zeitbedarf in $O(n)$
2. Phase:
   - einzelne **repeat**-Schleife:
     - Erwartungswert für Anzahl Durchläufe jeweils konstant (13.26)
     - Zeitbedarf (durch Test dominiert), also
     - proportional zu k_i^2.
   - Gesamtzeitbedarf
     - im wesentlichen durch $\sum k_i^2$ bestimmt,
     - i. e. durch die Gesamtzahl Kollisionen beschränkt
     - also wegen 1. Phase durch $O(n)$ beschränkt

Überblick

Überblick

Idee

Berechne

- aus echten Zufallsbits
- mittels Hashfunktionen *zusätzliche* Bits,
- die zwar nicht mehr echt unabhängig voneinander sind,
- „aber noch so aussehen“ und so verwendet werden (können).

Definition

Zufallsvariablen $X_1, \ldots, X_k$ *paarweise unabhängig*, wenn für alle $i \neq j$ und alle x und y gilt:

$$\mathbf{P}\big(X_i = x \wedge X_j = y\big) = \mathbf{P}(X_i = x) \cdot \mathbf{P}\big(X_j = y\big) \ .$$

In diesem Fall ist $\mathbf{E}\left[X_i X_j\right] = \mathbf{E}\left[X_i\right] \mathbf{E}\left[X_j\right]$.

Anmerkung

- paarweise Unabhängigkeit ist schwächere Forderung als (totale) Unabhängigkeit
- Beispiel:
  - X und Y unabhängige $\{0, 1\}$-Zufallsvariablen
  - $Z = X \oplus Y$
  - dann sind X, Y, Z paarweise unabhängig
  - aber offensichtlich nicht unabhängig.

Lemma

Für paarweise unabhängige Zufallsvariablen $X_1, \ldots, X_k$ gilt:

$$\mathbf{var}\left[\sum_i X_i\right] = \sum_i \mathbf{var}\left[X_i\right] \ .$$

Beweis

$$
\begin{aligned}
\mathbf{var}\left[\textstyle\sum_i X_i\right] &= \mathbf{E}\left[\left(\sum_i X_i\right)^2\right] - \left(\mathbf{E}\left[\sum_i X_i\right]\right)^2 \\
&= \mathbf{E}\left[\sum_i X_i^2 + \sum_{i\neq j} X_i X_j\right] - \left(\sum_i \mathbf{E}\left[X_i\right]\right)^2 \\
&= \sum_i \mathbf{E}\left[X_i^2\right] + \sum_{i\neq j} \mathbf{E}\left[X_i X_j\right] - \sum_i \left(\mathbf{E}\left[X_i\right]\right)^2 - \sum_{i\neq j} \mathbf{E}\left[X_i\right]\mathbf{E}\left[X_j\right] \\
&= \sum_i \mathbf{E}\left[X_i^2\right] - \sum_i \left(\mathbf{E}\left[X_i\right]\right)^2 + \sum_{i\neq j} \underbrace{\mathbf{E}\left[X_i X_j\right] - \mathbf{E}\left[X_i\right]\mathbf{E}\left[X_j\right]}_{=0} \\
&= \sum_i \left(\mathbf{E}\left[X_i^2\right] - \left(\mathbf{E}\left[X_i\right]\right)^2\right) = \sum_i \mathbf{var}\left[X_i\right]
\end{aligned}
$$

Definition

- M und N wie bisher und $|M| = m \geq |N| = n$.
- Familie H von Hashfunktionen $h : M \longrightarrow N$ heißt *stark 2-universell*, wenn für alle $x, y \in M$ mit $x \neq y$ und für alle $x', y' \in N$ gilt:
  - Für ein zufällig gleichverteilt aus H ausgewähltes h ist $\mathbf{P}(h(x) = x' \wedge h(y) = y') = 1/n^2$.

Beobachtung

- stark 2-universelle Familie von Hashfunktionen sind 2-universell im Sinne von Definition 13.3,
- denn

$$\begin{aligned}\mathbf{P}(h(x) = h(y)) &= \sum_{x' \in N} \mathbf{P}(h(x) = x' \wedge h(y) = x') \\ &= |N| / n^2 = 1/n \ .\end{aligned}$$

- Für jedes $x' \in N$ gilt außerdem:

$$\mathbf{P}(h(x) = x') = \sum_{y' \in N} \mathbf{P}(h(x) = x' \wedge h(y) = y') = 1/n$$

Also ist für eine zufällig gleichverteilt gewählte Hashfunktion aus einer stark 2-universellen Familie

$$\mathbf{P}(h(x) = x' \wedge h(y) = y') = \mathbf{P}(h(x) = x') \cdot \mathbf{P}(h(y) = y')$$

Vereinbarung

Im folgenden:

- Wahrscheinlichkeitsraum Ω: stark 2-universelle Familie von Hashfunktionen (von M nach N)
- Für $i \in M$ sei H_i die Zufallsvariable auf Ω mit $H_i(h) = h(i)$.
- Zufallsvariablen H_i sind paarweise unabhängig, denn

$$\begin{array}{ccc} \mathbf{P}\big(H_x = x' \wedge H_y = y'\big) & & \mathbf{P}(H_x = x') \cdot \mathbf{P}\big(H_y = y'\big) \\ \| & & \| \\ \mathbf{P}(h(x) = x' \wedge h(y) = y') & = & \mathbf{P}(h(x) = x') \cdot \mathbf{P}(h(y) = y') \end{array}$$

Beispiel

Abwandlung der Familie aus Definition 13.11:

- $m = n = 2^r$
- r Bits lange Wörter, interpretiert als Elemente des Körpers $F = GF[2^r]$
- für $a, b \in F$ sei $h_{a,b} : F \to F$ die Hashfunktion mit $h_{a,b}(i) = ai + b$.
- Die Familie der 2^{2r} solchen Hashfunktionen ist stark 2-universell.

Vereinbarung

- $L \subseteq A^*$ formale Sprache und T eine randomisierte **BPP**- oder **RP**-Turingmaschine für L
- Fasse T so auf, dass es ein „echtes“ Wort $x \in A^n$ und eine passende Folge $y \in \{0,1\}^r$ von Zufallsbits als zusätzliche Eingabe erhält.
- $T(x,y) \in \{0,1\}$ für das Ergebnis der Berechnung.
- $W_x = \{y \mid T(x,y) = 1\}$ bezeichne die Menge der y, die „bezeugen“, dass $x \in L$.
- $\mu(W_x) = |W_x| / 2^r$

Vereinbarung (2)

- $W_x = \{y \mid T(x,y) = 1\}$
- $\mu(W_x) = |W_x| / 2^r$
- Also

bei **RP**:
$$x \in L \implies \mu(W_x) > 1/2$$
$$x \notin L \implies \mu(W_x) = 0$$

bei **BPP**:
$$x \in L \implies \mu(W_x) > 3/4$$
$$x \notin L \implies \mu(W_x) < 1/4$$

- Außerdem ist in beiden Fällen r polynomiell in n.
- insbesondere
  $\forall n \in \mathbb{N}_0 \; \forall x \in A^n \; \exists y \in \{0,1\}^{r(n)} : x \in L \iff T(x,y) = 1$

Überblick

Definition

Eine formale Sprache $L \subseteq A^*$ ist in $\mathbf{P/poly}$, wenn es eine deterministische Polynomialzeit-Turingmaschine $T(x, y)$ und ein Polynom $r(n)$ gibt mit der Eigenschaft:

$$\forall n \in \mathbb{N}_0 \; \exists y \in \{0,1\}^{r(n)} \; \forall x \in A^n \; : \begin{cases} x \in L \Rightarrow T(x,y) = 1 \\ x \notin L \Rightarrow T(x,y) = 0 \end{cases}$$

Ein solches y heißt auch *„advice string“*.

Satz (Adleman, 1978)

$$\mathbf{RP} \subseteq \mathbf{P/poly} \qquad \text{und} \qquad \mathbf{BPP} \subseteq \mathbf{P/poly}$$

Anmerkungen:

- Für jede Wortlänge n gibt es *ein einzelnes Wort* y, das für alle Eingaben der Länge n so viel „Hilfestellung“ gibt, dass man deterministisch in Polynomialzeit das Wortproblem entscheiden kann.
  - im folgenden Beweis für den Fall **RP**

Satz (Adleman, 1978)

$$\mathbf{RP} \subseteq \mathbf{P/poly} \qquad \text{und} \qquad \mathbf{BPP} \subseteq \mathbf{P/poly}$$

Anmerkungen:

- Für jede Wortlänge n gibt es *ein einzelnes Wort* y, das für alle Eingaben der Länge n so viel „Hilfestellung“ gibt, dass man deterministisch in Polynomialzeit das Wortproblem entscheiden kann.
  - im folgenden Beweis für den Fall **RP**
- Wären die „advice strings“ in Polynomialzeit berechenbar
  - dann wäre $\mathbf{P/poly} = \mathbf{P}$,
  - und man hätte vollständige Derandomisierung erreicht.

Beweis (1)

- T eine **RP**-Turingmaschine, die L akzeptiert.
- Es sei n eine Wortlänge und $r = r(n)$ für Polynom $r(n)$.
- Betrachte die Matrix M,
  - Zeilen: mit den $|A|^n$ Wörtern $x \in A^n$ indiziert
  - Spalten: mit den 2^r Folgen von r (Zufalls-)Bits indiziert
  - Eintrag

$$M_{xy} = \begin{cases} 1 & \text{falls } y \in W_x \\ 0 & \text{falls } y \notin W_x \end{cases}$$

Beweis (2)

- Entferne aus M Zeilen für Wörter $w \notin L$
- neue Matrix: M_0.

Beweis (2)

- Entferne aus M Zeilen für Wörter $w \notin L$
- neue Matrix: M_0.
- in jeder Zeile mindestens die Hälfte aller Einträge 1

Beweis (3)

- Nun: iteratives Berechnen von Matrizen $M_1, M_2, \ldots$
- wenn in M_i in jeder Zeile mindestens die Hälfte der Bits 1 ist,

Beweis (3)

- Nun: iteratives Berechnen von Matrizen $M_1, M_2, \ldots$
- wenn in M_i in jeder Zeile mindestens die Hälfte der Bits 1 ist,
- dann ist insgesamt mindestens die Hälfte aller Bits 1,

Beweis (3)

- Nun: iteratives Berechnen von Matrizen $M_1, M_2, \ldots$
- wenn in M_i in jeder Zeile mindestens die Hälfte der Bits 1 ist,
- dann ist insgesamt mindestens die Hälfte aller Bits 1,
- dann in mindestens einer Spalte mindestens die Hälfte der Bits 1

Beweis (3)

- Nun: iteratives Berechnen von Matrizen $M_1, M_2, \ldots$
- wenn in M_i in jeder Zeile mindestens die Hälfte der Bits 1 ist,
- dann ist insgesamt mindestens die Hälfte aller Bits 1,
- dann in mindestens einer Spalte mindestens die Hälfte der Bits 1
- sei y_i die zu dieser Spalte zugehörige Bitfolge
- Zu M_i und y_i sei
  M_{i+1} die Matrix, die entsteht, wenn man
  - Spalte y_i streicht und
  - alle Zeilen, die in der gestrichenen Spalte eine 1 hatten.

Beweis (4)

- Zu M_i und y_i sei
  M_{i+1} die Matrix, die entsteht, wenn man
  - Spalte y_i streicht und
  - alle Zeilen, die in der gestrichenen Spalte eine 1 hatten.
- da in y_i mindestens die Hälfte aller Bits 1 war,
  - wird mindestens die Hälfte aller Zeilen von M_i gestrichen
  - in den verbleibenden Zeilen wird eine 0 gestrichen
  - in jeder Zeile von M_{i+1} mindestens die Hälfte der Bits 1
- nach spätestens $\log_2 |A|^n = cn \in \Theta(n)$ Iterationen alles gestrichen

Beweis (5)

Eine deterministische TM D mit Advice String kann wie folgt jedes Wort der Länge n bearbeiten:

- D bekommt $y = y_0 \cdots y_{cn}$ als Hilfestellung.
- zu Eingabe x arbeitet D erst mal so
  - bestimmt zunächst $n = |x|$
  - bestimmt $r = r(n)$
  - zerlegt y in Teilwörter y_i der Länge r.
- dann simuliert D alle $T(x, y_i)$
  - bis einmal akzeptiert wurde und akzeptiert dann auch,
  - oder lehnt x ab, falls kein $T(x, y_i)$ akzeptiert.
- Länge von y polynomiell und mit T auch D in Polynomialzeit

Überblick

13.43 Algorithmus (Chor/Goldreich, 1989)

- T eine **BPP**-Turingmaschine für L;
  brauche für Eingabelänge n polynomiell viele $r = r(n)$ Zufallsbits
- $H = \{h_{a,b} : \{0,1\}^r \to \{0,1\}^r\}$ stark 2-universell (13.37)
- $2r$ Zufallsbits (a, b) um ein $h \in H$ auszuwählen

Der neue Algorithmus arbeitet dann für eine Konstante k so:

- wähle zufällig ein $h_{a,b}$ aus H aus.
- berechne für $i = 1, \ldots, k$ jeweils $y_i = h_{a,b}(i)$.
- zähle, für wieviele i gilt: $y_i \in W_x$:
  - falls mindestens $k/2$ mal: x akzeptieren
  - andernfalls x ablehnen

Satz

Die Fehlerwahrscheinlichkeit von Algorithmus 13.43 ist $O(1/k)$.

Beweis (1)

Betrachte die Zufallsvariablen

$$X_i = \begin{cases} 1 & \text{falls } y_i \in W_x \\ 0 & \text{sonst} \end{cases} .$$

Beweis (2)

Die X_i sind identisch verteilt.
Zeige: Sie sind auch paarweise unabhängig:

$$
\begin{aligned}
& \mathbf{P}\big(X_i = a \wedge X_j = b\big) = \mathbf{P}\big([y_i \in W_x] = a \wedge [y_j \in W_x] = b\big) \\
= \quad & \mathbf{P}([h(i) \in W_x] = a \wedge [h(j) \in W_x] = b) \\
= \quad & \sum_{z \in W_x} \sum_{z' \in W_x} \mathbf{P}([h(i) \in \{z\}] = a \wedge [h(j) \in \{z'\}] = b) \\
= \quad & \sum_{z \in W_x} \sum_{z' \in W_x} \mathbf{P}([h(i) \in \{z\}] = a) \cdot \mathbf{P}([h(j) \in \{z'\}] = b) \\
= \quad & \sum_{z \in W_x} \mathbf{P}([h(i) \in \{z\}] = a) \cdot \sum_{z' \in W_x} \mathbf{P}([h(j) \in \{z'\}] = b) \\
= \quad & \mathbf{P}([h(i) \in W_x] = a) \cdot \mathbf{P}([h(j) \in W_x] = b) \\
= \quad & \mathbf{P}(X_i = a) \cdot \mathbf{P}\big(X_j = b\big)
\end{aligned}
$$

Beweis (3)

- Der Erwartungswert ist $\mu = \mathbf{E}[X_i] = \mu(W_x)$.
- Die Varianz ist
  $\mathbf{var}[X_i] = \mathbf{E}\left[X_i^2\right] - (\mathbf{E}[X_i])^2 = \mathbf{E}[X_i] - (\mathbf{E}[X_i])^2 = \mu(1-\mu)$.
- Da $0 \leq \mu \leq 1$, ist $\mathbf{var}[X_i] \leq 1/4$.
- Also ist $\mathbf{var}[\sum X_i] \leq k/4$.

Beweis (4)

- Der Algorithmus liefert das falsche Ergebnis, falls
  - $x \in L$ und $\sum X_i < k/2$ oder $x \notin L$ und $\sum X_i \geq k/2$, d. h.
  - $\mu(W_x) > 3/4$ und $\sum X_i < k/2$ oder $\mu(W_x) < 1/4$ und $\sum X_i \geq k/2$, d. h.
  - $k\mu(W_x) > 3k/4$ und $\sum X_i < k/2$ oder $k\mu(W_x) < k/4$ und $\sum X_i \geq k/2$.
- beide Fälle zusammengefasst: $\left|\sum X_i - \mathbf{E}\left[\sum X_i\right]\right| > k/4$
- Ungleichung von Chebyshev liefert

$$\mathbf{P}\left(\left|\sum X_i - \mathbf{E}\left[\sum X_i\right]\right| > k/4\right) \leq \frac{16\,\mathbf{var}\left[\sum X_i\right]}{k^2} \leq \frac{4}{k}\,.$$

Bemerkung

Beachte: für Fehlerwahrscheinlichkeit $1/k$ benötigen

- Chor/Goldreich nur $2r$ Zufallsbits
- unabhängige Wiederholungen aber $r \log k$ Zufallsbits

13.46 Hash Mixing Lemma

Sei H eine stark 2-universelle Familie von Hashfunktionen von $\{0,1\}^r$ nach $\{0,1\}^r$ und $\varepsilon = 2^{-r/3}$.

Dann gilt für alle Teilmengen $A, B \subseteq \{0,1\}^r$ und für fast alle Hashfunktionen h außer einem Bruchteil ε von ihnen:

$$\left|\mathbf{P}_y(y \in A \wedge h(y) \in B) - \mathbf{P}_{y,z}(y \in A \wedge z \in B)\right| \leq \varepsilon \ .$$

Schreibweise $\mathbf{P}_y(\ldots)$:
Die Wahrscheinlichkeit des Ereignisses ist gemeint für den Fall, dass y zufällig gleichverteilt aus dem zugrunde liegenden Universum gewählt ist, und analog für $\mathbf{P}_{y,z}(\ldots)$.

13.47 Beweis (1)

- Schätze Anzahl der „schlechten“ h mit der Eigenschaft

$$\left|\mathbf{P}_y(y \in A \wedge h(y) \in B) - \mathbf{P}_{y,z}(y \in A \wedge z \in B)\right| \geq \varepsilon \ .$$

  nach oben ab.
- Das ist gleich der Anzahl der h mit

$$\left|\mathbf{P}_y(h(y) \in B \mid y \in A) - \mu(B)\right| \geq \frac{\varepsilon}{\mu(A)}$$

$$\left||A|\mathbf{P}_y(h(y) \in B \mid y \in A) - |A|\mu(B)\right| \geq \frac{\varepsilon|A|}{\mu(A)}$$

- für festes h paarweise unabhängige Zufallsvariablen

$$Z_y^h = \begin{cases} 1 & \text{falls } h(y) \in B \\ 0 & \text{sonst} \end{cases}$$

13.47 Beweis (2)

- Zufallsvariablen

$$Z_y^h = \begin{cases} 1 & \text{falls } h(y) \in B \\ 0 & \text{sonst} \end{cases}$$

- Eine Hashfunktion h ist schlecht, falls

$$\left| \sum_{y \in A} Z_y^h - |A|\mu(B) \right| \geq \frac{\varepsilon |A|}{\mu(A)} = \varepsilon 2^r \ .$$

13.47 Beweis (3)

- Eine Hashfunktion h ist schlecht, falls

$$\left| \sum_{y \in A} Z_y^h - |A|\mu(B) \right| \geq \frac{\varepsilon |A|}{\mu(A)} = \varepsilon 2^r \ .$$

- Ungleichung von Chebyshev:

$$\mathbf{P}_h\left(\left| \sum_{y \in A} Z_y^h - |A|\mu(B) \right| \geq \varepsilon 2^r \right)$$
$$\leq \frac{\mathbf{var}\left[\sum_{y \in A} Z_y^h\right]}{\varepsilon^2 2^{2r}} = \frac{|A|\mathbf{var}\left[Z_y^h\right]}{\varepsilon^2 2^{2r}} \leq \frac{2^r \cdot 1}{2^{-2r/3} \cdot 2^{2r}} = 2^{-r/3} = \varepsilon$$

13.48 Algorithmus (Nisan, 1994) (1)

- Gegeben sei eine **RP**-Turingmaschine T mit Fehlerwahrscheinlichkeit $p = \sup_x \left|\overline{W}_x\right| < 1/2$.
- Sei x eine Eingabe der Länge n.
- Zur Verringerung der Fehlerwahrscheinlichkeit wird
  - eine (in n) polynomiale Anzahl k von Läufen $T(x, y_i)$
  - für (nicht total unabhängige) y_i durchgeführt,
  - und x genau dann akzeptiert, wenn in mindestens einem Fall $T(x, y_i)$ akzeptiert.
- Die y_i werden mit Hilfe von nur $O(r \log k)$ echter Zufallsbits erzeugt.

13.48 Algorithmus (2)

Die y_i werden mit Hilfe von nur $O(r \log k)$ echter Zufallsbits erzeugt:

- Es werden $\ell = \lceil \log k \rceil$ Hashfunktionen $h_1, \ldots, h_\ell$ unabhängig voneinander aus der stark 2-universellen Familie von Hashfunktionen aus Beispiel 13.37 ausgewählt.
- Dafür sind jeweils $2r$ Zufallsbits nötig.
- Es wird ein zufälliger Startwert $y \in \{0,1\}^r$ gewählt.
- Werte $y_0, \ldots, y_{2^a-1}$ sind festgelegt als die $y_i = g_i(y)$.
- Dabei sind die Funktionen g_i rekursiv festgelegt.

13.48 Algorithmus (3)

Die y_i werden mit Hilfe von nur $O(r \log k)$ echter Zufallsbits erzeugt:

- $\ell = \lceil \log k \rceil$ Hashfunktionen $h_1, \ldots, h_\ell$ unabhängig voneinander
- Es wird ein zufälliger Startwert $y \in \{0,1\}^r$ gewählt.
- Werte $y_0, \ldots, y_{2^a-1}$ sind festgelegt als die $y_i = g_i(y)$.
- Funktionen g_i rekursiv wie folgt festgelegt.:
  - g_0 ist die Identität.
  - Sind $g_0, \ldots, g_{2^a-1}$ schon definiert,
  - dann sind die nächsten 2^a Funktionen $g_{i+2^a} = g_i \circ h_{a+1}$.
- Insgesamt also: $y, h_1(y), h_2(y), h_1(h_2(y)),$
  $h_3(y), h_1(h_3(y)), h_2(h_3(y)), h_1(h_2(h_3(y))), \ldots$

Bemerkung

- Abkürzung $M_j(y) = \{y = y_0, \ldots, y_{2^j-1}\}$
- für die Menge aller y_i in Abhängigkeit von Startwert y und Anzahl benutzter Hashfunktionen.
- Der Algorithmus liefert also genau dann die falsche Antwort, wenn $M(y) = M_\ell(y) \subseteq \overline{W}_x$ ist.

13.49 Lemma

Für Algorithmus 13.48 gilt: Ist $x \in L$, dann ist

$$\mathbf{P}\left(M(y) \subseteq \overline{W}_x\right) \leq \mu(\overline{W}_x)^{2^\ell} + (\ell + 2)\varepsilon$$

für $\varepsilon = 2^{-r/3}$.

ohne Beweis

13.51 Satz

Die Fehlerwahrscheinlichkeit von Algorithmus 13.48 kleiner als $2^{-O(k(n))}$.

13.52 Beweis

vorangegangenes Lemma:

$$\mathbf{P}\left(M(y) \subseteq \overline{W}_x\right) \leq \mu(\overline{W}_x)^{2^\ell} + (\ell+2)\varepsilon$$

$$\mathbf{P}\left(M(y) \subseteq \overline{W}_x\right) \leq \mu(\overline{W}_x)^{2^\ell} + (\ell+2)2^{-r/3}$$

Da k polynomiell in n ist, $2^\ell \in O(k)$ und $\mu(\overline{W}) < 1/2$, folgt die Behauptung.