Zufall oder Absicht?

Randomisierung und Derandomisierung

Markus Bläser

Universität des Saarlandes

17. Januar 2011

Algorithmus

Eingabe x

```
int main(int x) {
⋮
}
```

Ausgabe y

Algorithmus

Eingabe x

```
int main(int x) {
⋮
}
```

Ausgabe y

Jetzt: Algorithmus kann Münzen werfen,
d.h. er erhält Bits, die jeweils mit Wahrscheinlichkeit $1/2$ Null oder Eins sind.

Komplexitätstheorie

Komplexitätstheoretische Untersuchungen:

1. Abstraktion $\longrightarrow$ mathematische Modelle
2. Beweis exakter Aussagen über
   1. Laufzeit
   2. Speicherplatz
   3. Anzahl verwendeter Zufallsbits
   4. ...

"Beweis" und "exakt" heißt z.B.

- auf jede Eingabe x werden $\leq f(|x|)$ viele Rechenschritte gemacht
- und *nicht*: auf 100 Testeingaben wurde im Mittel 10ms gerechnet.

Komplexitätstheorie

Komplexitätstheoretische Untersuchungen:

1. Abstraktion $\longrightarrow$ mathematische Modelle
2. Beweis exakter Aussagen über
   1. Laufzeit
   2. Speicherplatz
   3. Anzahl verwendeter Zufallsbits
   4. ...

"Beweis" und "exakt" heißt z.B.

- auf jede Eingabe x werden $\leq f(|x|)$ viele Rechenschritte gemacht
- und *nicht*: auf 100 Testeingaben wurde im Mittel 10ms gerechnet.

Konkret bedeutet dies für heute:

- weniger bunte Bilder, mehr Mathematik

Komplexitätstheorie

Komplexitätstheoretische Untersuchungen:

1. Abstraktion $\longrightarrow$ mathematische Modelle
2. Beweis exakter Aussagen über
   1. Laufzeit
   2. Speicherplatz
   3. Anzahl verwendeter Zufallsbits
   4. ...

"Beweis" und "exakt" heißt z.B.

- auf jede Eingabe x werden $\leq f(|x|)$ viele Rechenschritte gemacht
- und *nicht*: auf 100 Testeingaben wurde im Mittel 10ms gerechnet.

Konkret bedeutet dies für heute:

- weniger bunte Bilder, mehr Mathematik
- weniger Verpackung, mehr Inhalt

Randomisierte Algorithmen (2)

Was ist die Ausgabe eines randomisierten Algorithmus?

Randomisierte Algorithmen (2)

Was ist die Ausgabe eines randomisierten Algorithmus?

Wahrscheinlichkeitsverteilung auf den möglichen Outputs, also

Ausgabe	0	1	2	3	...
Wahrscheinlichkeit	0.1	0	0.25	0.05	...

Randomisierte Algorithmen (2)

Was ist die Ausgabe eines randomisierten Algorithmus?

Wahrscheinlichkeitsverteilung auf den möglichen Outputs, also

Ausgabe	0	1	2	3	...
Wahrscheinlichkeit	0.1	0	0.25	0.05	...

Entscheidungsproblem

Ausgabe 1: Eingabe hat Eigenschaft
Ausgabe 0: Eingabe hat Eigenschaft nicht

Beispiel: Primzahltest

Eingabe: $n \in \mathbb{N}$

Ausgabe: $\begin{cases} 1 & n \text{ prim} \\ 0 & \text{sonst} \end{cases}$

Randomisierte Algorithmen (3)

Beispiel: Primzahltest

Eingabe n prim:	Ausgabe 1 mit Wahrscheinlichkeit $> 1/2$
	(damit Augabe 0 mit Wahrscheinlichkeit $< 1/2$)
n nicht prim:	Ausgabe 0 mit Wahrscheinlichekit $> 1/2$
	(und Ausgabe 1 mit Wahrscheinlichkeit $< 1/2$).

Randomisierte Algorithmen (3)

Beispiel: Primzahltest

Eingabe n prim:	Ausgabe 1 mit Wahrscheinlichkeit $> 1/2$
	(damit Augabe 0 mit Wahrscheinlichkeit $< 1/2$)
n nicht prim:	Ausgabe 0 mit Wahrscheinlichekit $> 1/2$
	(und Ausgabe 1 mit Wahrscheinlichkeit $< 1/2$).

Problem: $0.50000000000001 > 1/2$, aber die Ausgabe des Primzahltest wird sich kaum von einem Münzwurf unterscheiden!

Randomisierte Algorithmen (3)

Beispiel: Primzahltest

Eingabe n prim:	Ausgabe 1 mit Wahrscheinlichkeit $> 1/2$
	(damit Augabe 0 mit Wahrscheinlichkeit $< 1/2$)
n nicht prim:	Ausgabe 0 mit Wahrscheinlichekit $> 1/2$
	(und Ausgabe 1 mit Wahrscheinlichkeit $< 1/2$).

Problem: $0.50000000000001 > 1/2$, aber die Ausgabe des Primzahltest wird sich kaum von einem Münzwurf unterscheiden!

Lösung: *Beschränkte Fehlerwahrscheinlichkeit!*

Eingabe n prim:	Ausgabe 1 mit Wahrscheinlichkeit $\geq 2/3$
	(damit Ausgabe 0 mit Wahrscheinlichkeit $\leq 1/3$)
n nicht prim:	Ausgabe 0 mit Wahrscheinlichekit $\geq 2/3$
	(und Ausgabe 1 mit Wahrscheinlichkeit $\leq 1/3$).

Probability Amplification

Frage: Warum ist $\geq 2/3$ besser als $> 1/2$.

Probability Amplification

Frage: Warum ist $\geq 2/3$ besser als $> 1/2$.

Antwort: Probability Amplification!

Idee: Lasse Algorithmus *mehrmals* statt einmal laufen und treffe Mehrheitsentscheid.

Probability Amplification

Frage: Warum ist $\geq 2/3$ besser als $> 1/2$.

Antwort: Probability Amplification!

Idee: Lasse Algorithmus *mehrmals* statt einmal laufen und treffe Mehrheitsentscheid.

Wahrscheinlichkeiten:

einmal		$\frac{2}{3}$
dreimal	$\left(\frac{2}{3}\right)^3 + 3 \cdot \left(\frac{2}{3}\right)^2 \frac{1}{3} =$	$\frac{20}{27} \approx 0.74$
fünfmal	$\ldots$	$\frac{176}{243} \approx 0.79$
$\vdots$	$\vdots$	$\vdots$
n-mal	$\ldots$	$\approx 1 - 2^{-c \cdot n}$

Selbst für mäßig große n ist die Fehlerwahrscheinlichkeit z.B. kleiner als die Wahrscheinlichkeit eines Hardware-Fehlers.

Bedeutung der Randomisierung

Zentrale Frage: Hilft Randomisierung?

Bedeutung der Randomisierung

Zentrale Frage: Hilft Randomisierung?

Zwei Antworten:

1. Kommt darauf an.
2. Vielleicht, aber eher nicht.

Derandomisierung

Prinzipiell hilft Randomisierung nicht:

Theorem

Kann $f : \mathbb{N} \to \{0, 1\}$ *durch einen randomisierten Algorithmus* A *berechnet werden, dann kann* f *auch durch einen deterministischen Algorithmus* $\hat{A}$ *berechnet werden.*

[1] Länge von $n \in \mathbb{N}$: Anzahl Bits $\approx \log_2(n)$

Derandomisierung

Prinzipiell hilft Randomisierung nicht:

Theorem

Kann $f : \mathbb{N} \to \{0, 1\}$ durch einen randomisierten Algorithmus A berechnet werden, dann kann f auch durch einen deterministischen Algorithmus $\hat{A}$ berechnet werden.

Beweis:

- A verwende $\leq r(\ell)$ Zufallsbits auf Eingaben der Länge ℓ.[1]
- $\hat{A}$ zählt alle $\{0, 1\}$-Strings z der Länge $r(\ell)$ auf und
- simuliert A und verwendet jeweils z als Zufallsbits.
- $\hat{A}$ trifft Mehrheitsentscheid am Ende.

[1] Länge von $n \in \mathbb{N}$: Anzahl Bits $\approx \log_2(n)$

Derandomisierung

Prinzipiell hilft Randomisierung nicht:

Theorem

Kann $f : \mathbb{N} \to \{0, 1\}$ durch einen randomisierten Algorithmus A berechnet werden, dann kann f auch durch einen deterministischen Algorithmus $\hat{A}$ berechnet werden.

Beweis:

- A verwende $\leq r(\ell)$ Zufallsbits auf Eingaben der Länge ℓ.[1]
- $\hat{A}$ zählt alle $\{0, 1\}$-Strings z der Länge $r(\ell)$ auf und
- simuliert A und verwendet jeweils z als Zufallsbits.
- $\hat{A}$ trifft Mehrheitsentscheid am Ende.

Problem: exponentieller Blowup!

[1] Länge von $n \in \mathbb{N}$: Anzahl Bits $\approx \log_2(n)$

Polynom-Identitätstests

Eingabe: Polynom $p(X) = a_n X^n + a_{n-1} X^{n-1} + \cdots + a_1 X + a_0$
vom Grad $\leq n$ mit $a_\nu \in \mathbb{Z}$, $0 \leq \nu \leq n$.

Ausgabe: $\begin{cases} 1 & \text{falls } p \equiv 0 \\ 0 & \text{sonst} \end{cases}$

Polynom-Identitätstests

Eingabe: Polynom $p(X) = a_n X^n + a_{n-1} X^{n-1} + \cdots + a_1 X + a_0$
vom Grad $\leq n$ mit $a_\nu \in \mathbb{Z}$, $0 \leq \nu \leq n$.

Ausgabe: $\begin{cases} 1 & \text{falls } p \equiv 0 \\ 0 & \text{sonst} \end{cases}$

So nicht schwierig ...
weil $p \equiv 0 \iff a_n = a_{n-1} = \cdots = a_1 = a_0 = 0$

Polynom-Identitätstests (2)

Modell: *Blackbox-Modell*

Eingabe $x \in \mathbb{Z}$

$$\boxed{p(X) = a_n X^n + a_{n-1} X^{n-1} + \cdots + a_1 X + a_0}$$

Ausgabe: $p(x) \in \mathbb{Z}$

Polynom-Identitätstests (2)

Modell: *Blackbox-Modell*

Eingabe $x \in \mathbb{Z}$

$$p(X) = a_n X^n + a_{n-1} X^{n-1} + \cdots + a_1 X + a_0$$

Ausgabe: $p(x) \in \mathbb{Z}$

- Algorithmus kann p an beliebiger Stelle auswerten,
- hat aber keinen Zugriff auf die Koeffizienten.
- Algorithmus kennt n (obere Schranke für Grad).
- Ausgabe: $\begin{cases} 1 & \text{falls } p \text{ in Blackbox identisch } 0 \\ 0 & \text{sonst} \end{cases}$

Polynom-Identitätstests (3)

Theorem

1. *Es gibt einen randomisierten Algorithmus für das Polynom-Identitätsproblem mit Fehlerwahrscheinlichkeit* $\leq 1/3$*, der die Blackbox nur einmal fragt.*
2. *Jeder deterministische Algorithmus, der das Polynom-Identitätsproblem löst, muß die Blackbox mindestens* $(n+1)$*-mal fragen.*

Beweis: Tafel