Algorithmen und Wahrscheinlichkeit

Kapitel 2.7

Die Ungleichung von Chernoff

Abschätzen von Wahrscheinlichkeiten

Markov:

$$\Pr[X \geq t] \leq \frac{\mathbb{E}[X]}{t} \qquad \forall\, X \geq 0, \quad \forall\, t > 0$$

Chebyshev:

$$\Pr[|X - \mathbb{E}[X]| \geq t] \leq \frac{\mathrm{Var}[X]}{t^2} \qquad \forall\, X, \quad \forall\, t > 0$$

Chernoff:

$$\Pr[X \geq (1+\delta)\mathbb{E}[X]] \leq e^{-\frac{1}{3}\delta^2 \mathbb{E}[X]} \qquad \forall\, X \sim \mathrm{Bin}(n,p), \quad \forall\, 0 < \delta < 1$$

Chernoff Schranken

$\Pr[X \geq (1+\delta)\ \mathbb{E}[X]]$ für $\delta = 0.1$:

n	Chebyshev	Chernoff
1000	0.1	0.270961
2000	0.05	0.0424119
5000	0.02	0.000244096
10000	0.01	$5.77914 \cdot 10^{-8}$
100000	0.001	$4.14559 \cdot 10^{-73}$

Kapitel 2.8

Randomisierte Algorithmen

Algorithmen

Klassischer Algorithmus:

Wir beweisen:

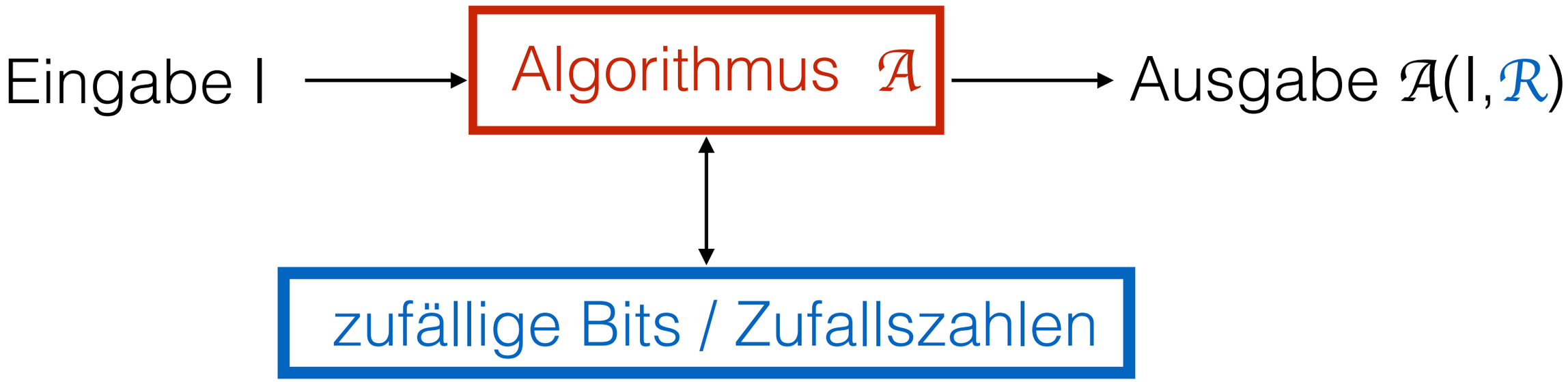
(1) Korrektheit:

für alle Eingaben I gilt: $\mathcal{A}(I)$ ist korrekt (d.h. was es sein soll)

(2) Laufzeit:

für alle Eingaben I mit Länge |I|=n: Laufzeit = O(f(n))

Randomisierte Algorithmen

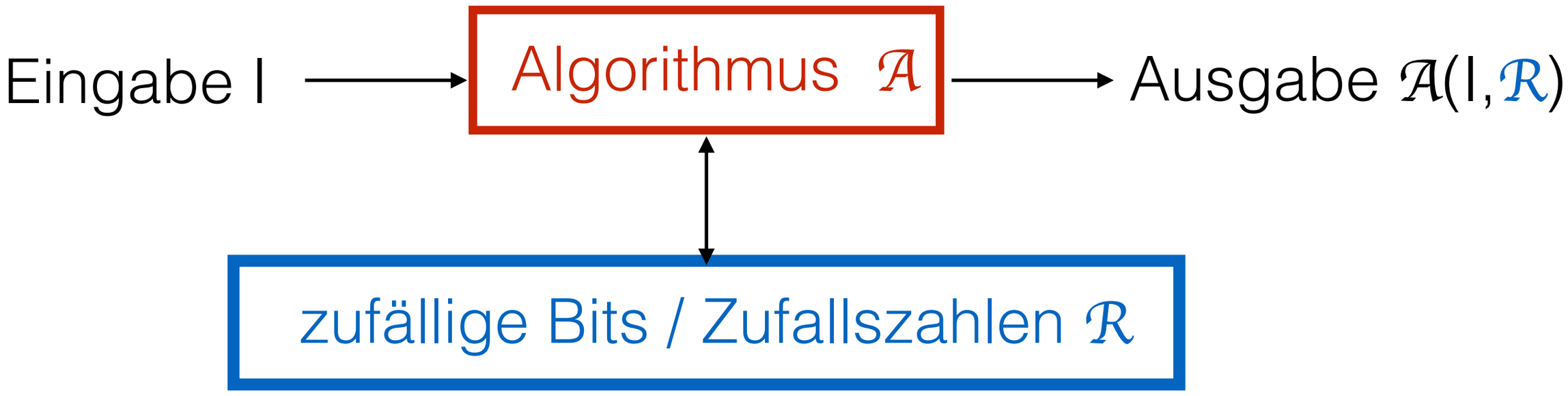


Eigenschaften:

Ausgabe $\mathcal{A}(I,\mathcal{R})$ hängt von Eingabe I *und* Zufallszahlen $\mathcal{R}$ ab.

Insbesondere: Ergebnis lässt sich i.A. nicht reproduzieren .. !

Randomisierte Algorithmen



Wir beweisen:

W'keit ist bzgl Wahl der Zufallszahlen $\mathcal{R}$

(1) Korrektheit:

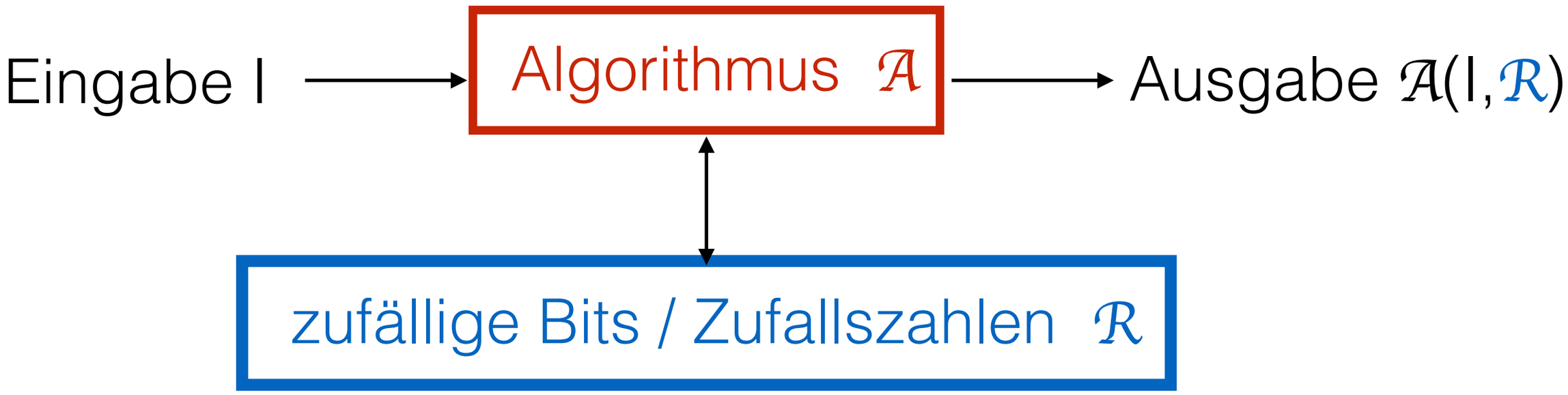
für alle Eingaben I gilt: $\Pr[\mathcal{A}(I,\mathcal{R})$ ist korrekt$] \geq \ldots$

(2) Laufzeit:

für alle Eingaben I mit Länge $|I|=n$:

$\mathbb{E}[\text{Laufzeit}] = O(f(n))$ und/oder $\Pr[\text{Laufzeit} \leqq f(n)] \geq \ldots$

Randomisierte Algorithmen



Wir beweisen:

(1) Korrektheit:

für alle Eingaben I gilt: $\Pr[\mathcal{A}(I,\mathcal{R})$ ist korrekt$] \geq \ldots$

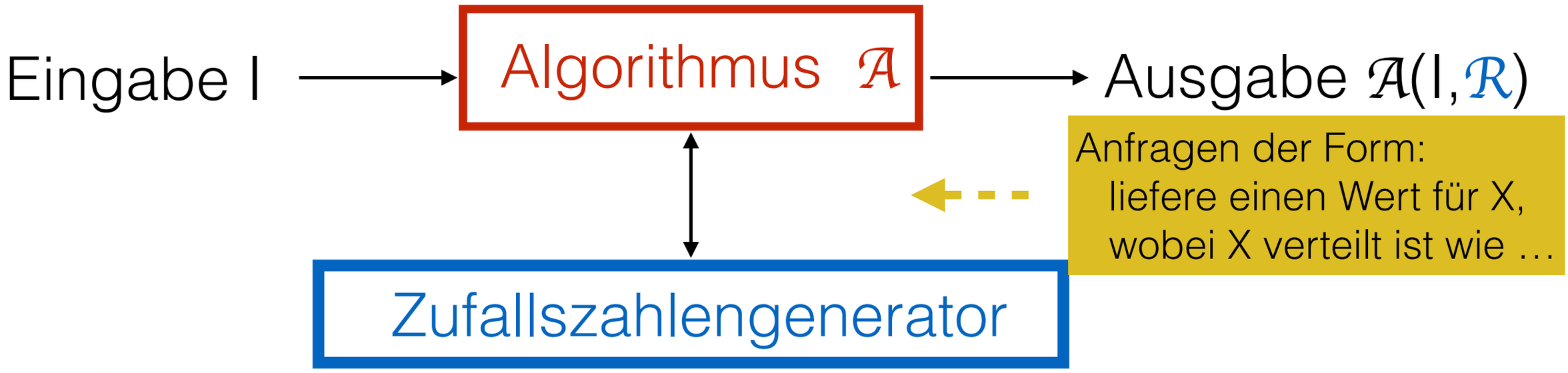
(2) Laufzeit:

für alle Eingaben I mit Länge $|I|=n$:

$\mathbb{E}[\text{Laufzeit}] = O(f(n))$ und/oder $\Pr[\text{Laufzeit} \leqq f(n)] \geq \ldots$

Idealer Weise:
W'keit „praktisch" Eins

Randomisierte Algorithmen



Annahme:
alle Werte, die der Zufallszahlengenerator erzeugt sind unabhängig

Wir beweisen:

(1) Korrektheit:

für alle Eingaben I gilt: $\Pr[\mathcal{A}(I,\mathcal{R}) \text{ ist korrekt}] \geq \ldots$

(2) Laufzeit:

für alle Eingaben I mit Länge $|I|=n$:

$\mathbb{E}[\text{Laufzeit}] = O(f(n))$ und/oder $\Pr[\text{Laufzeit} \leqq O(f(n))] \geq \ldots$

Idealer Weise:
W'keit „praktisch" Eins

Las-Vegas und Monte-Carlo Algorithmen

Las-Vegas Algorithmen:

- geben nie eine falsche Antwort, aber
- Laufzeit ist eine Zufallsvariable

Ziel: $\mathbb{E}$[Laufzeit] = „polynomiell“ (in Eingabelänge)

Monte-Carlo Algorithmen:

- Laufzeit immer polynomiell, aber
- geben zuweilen eine falsche Antwort

Ziel: Pr[Antwort falsch] = „winzig“

Beispiele

Las-Vegas Algorithmen:

- QuickSort

Monte-Carlo Algorithmen:

Aufgabe: sortiere Elemente

```
QuickSort(A, ℓ, r)
1: if ℓ < r then
2:     p ← Uniform({ℓ, ℓ + 1, ..., r})        ▷ wähle Pivotelement zufällig
3:     t ← Partition(A, ℓ, r, p)
4:     QuickSort(A, ℓ, t − 1)
5:     QuickSort(A, t + 1, r)
```

Satz:

- QuickSort bestimmt *immer* das richtige Ergebnis
- $\mathbf{E}$[Laufzeit] = O(n ln n)

Selektieren

Aufgabe: finde das *k*-te kleinste Element aus einem unsortierten Array

Select([12, 3, 22, 67, 8, 15, 19, 13] , 4): 13

Selektieren

QuickSelect(A, ℓ, r, k)

1: $p \leftarrow \text{Uniform}(\{\ell, \ell+1, \ldots, r\})$ ▷ wähle Pivotelement zufällig
2: $t \leftarrow$ **Partition**(A, ℓ, r, p)
3: **if** $t = \ell + k - 1$ **then**
4: **return** $A[t]$ ▷ gesuchtes Element ist gefunden
5: **else if** $t > \ell + k - 1$ **then**
6: **return** **QuickSelect**$(A, \ell, t-1, k)$ ▷ gesuchtes Element ist links
7: **else**
8: **return** **QuickSelect**$(A, t+1, r, k-t)$ ▷ gesuchtes Element ist rechts

Selektieren

```
QuickSelect(A, ℓ, r, k)
1: p ← Uniform({ℓ, ℓ + 1, ..., r})          ▷ wähle Pivotelement zufällig
2: t ← Partition(A, ℓ, r, p)
3: if t = ℓ + k − 1 then
4:     return A[t]                           ▷ gesuchtes Element ist gefunden
5: else if t > ℓ + k − 1 then
6:     return QuickSelect(A, ℓ, t − 1, k)    ▷ gesuchtes Element ist links
7: else
8:     return QuickSelect(A, t + 1, r, k − t)  ▷ gesuchtes Element ist rechts
```

Satz:

- QuickSelect bestimmt *immer* das richtige Ergebnis
- $\mathbf{E}$[Laufzeit] = O(n)

Selektieren:

erwartete Laufzeit: $O(n)$

Frage: Was bedeutet dies für die Praxis?

Gedankenexperiment

t_n := erwartete Laufzeit von Quickselect bei n Elementen

Wir definieren einen neuen Algorithmus wie folgt:

SuperQuickSelect(A,1,n,k)

rufe quickselect(A,1,n,k) auf, sobald Laufzeit grösser als $2t_n$:
breche Ausführung ab und gebe ??? aus

wg. Markov Ungleichung

Dies ist ein randomisierter Algorithmus mit
- Laufzeit ≦ $2t_n$
- Wahrscheinlichkeit für ??? ≦ 1/2

Gedankenexperiment

t_n := erwartete Laufzeit von Quickselect bei n Elementen

SuperQuickSelect(A,1,n,k)

rufe QuickSelect(A,1,n,k) auf,
sobald Laufzeit grösser als $2t_n$:
    breche Ausführung ab und gebe ??? aus

SuperSuperQuickSelect(A,1,n,k)

wiederhole maximal 100 mal:
    rufe SuperQuickSelect(A,1,n,k) auf,
    terminiere, falls dieser Ergebnis findet (also nicht ??? ausgibt)
gebe ??? aus

Dies ist ein randomisierter Algorithmus mit
- Laufzeit ≦ $200t_n$
- Wahrscheinlichkeit für ??? ≦ 2^{-100}

Las-Vegas Algorithmen:

- geben nie eine falsche Antwort, aber
- Laufzeit ist eine Zufallsvariable T

mit: $\mathbb{E}[T]$ = „polynomiell (in Eingabelänge)“

⇓ **stoppe Alg nach $2\mathbb{E}[T]$ Schritten**

… wdh 100 Mal

- Laufzeit immer polynomiell, aber
- zuweilen Antwort „???“

Pr[Antwort „???“] $\leq (1/2)^{100}$ (wg Markov Ungleichung)

Las-Vegas Algorithmen:

- geben nie eine falsche Antwort, aber
- Laufzeit ist eine Zufallsvariable T

$$\mathbb{E}[T] = \frac{1}{1-\delta} \cdot \mathit{poly}$$

⇑ **while Antwort „???“: repeat**

(Anzahl Versuche: Geo(1-δ))

- Laufzeit immer polynomiell, aber
- zuweilen Antwort „???“

mit: Pr[Antwort „???“] = δ

Las-Vegas Algorithmen:

- geben nie eine falsche Antwort, aber
- Laufzeit ist eine Zufallsvariable T

Ziel: $\mathbb{E}[T]$ = „polynomiell“ (in Eingabelänge)

alternative Definition:

- Laufzeit immer polynomiell, aber
- geben zuweilen eine Antwort „???“

Ziel: Pr[Antwort „???“] = „winzig“

Beispiele

Las-Vegas Algorithmen:

- QuickSort, QuickSelect

Monte-Carlo Algorithmen:

- Testen einer Münze: fair (Kopf/Zahl) vs. fake (Kopf/Kopf)

Algorithmus: werfe Münze ein Mal, falls Zahl: return „fair“
falls Kopf: return „fake“

Fehlerw’lichkeit:

0, falls Münze fake

1/2, falls Münze fair

Beispiele

Las-Vegas Algorithmen:

- QuickSort

Monte-Carlo Algorithmen:

- Testen einer Münze: fair (Kopf/Zahl) vs. fake (Kopf/Kopf)

Algorithmus: werfe Münze ~~ein~~ **100** Mal, falls $\geq$ **1 mal** Zahl: return „fair“
ansonsten: return „fake“

Fehlerw’lichkeit:

$$\begin{cases} 0, & \text{falls Münze fake} \\ (1/2)^{100}, & \text{falls Münze fair} \end{cases}$$

Monte-Carlo Algorithmen für Entscheidungsprobleme

entscheide: Ist Antwort „ja“ oder „nein“

Angenommen wir haben einen Algorithmus mit einseitigem Fehler:

Pr[Alg antwortet „nein“] = 0 ∀ Ja-Instanzen

Pr[Alg antwortet „ja“] ≦ 1-ε ∀ Nein-Instanzen

Dann gilt: $\varepsilon^{-1} \ln \delta^{-1}$ Wiederholungen reduzieren Fehler auf δ

(Antwort „nein“: wenn mind. ein Aufruf „nein“ ausgibt,
Antwort „ja“: wenn alle Wdh “ja“ ausgeben)

Pr[Alg gibt falsche Antwort] ≦ $(1-\varepsilon)^{\epsilon^{-1} \ln \delta^{-1}}$ ≦.. ≦ δ

Monte-Carlo Algorithmen für Entscheidungsprobleme

Einseitiger Fehler:

Pr[Alg antwortet „nein“] = 0 ∀ Ja-Instanzen

Pr[Alg antwortet „ja“] ≦ 1-ε ∀ Nein-Instanzen

⇒ $\varepsilon^{-1} \ln \delta^{-1}$ Wiederholungen reduzieren Fehler auf

Pr[Antwort falsch] ≦ δ

(Antwort „nein“: wenn mind. ein Aufruf „nein“ ausgibt,
Antwort „ja“: wenn alle Wdh “ja“ ausgeben)

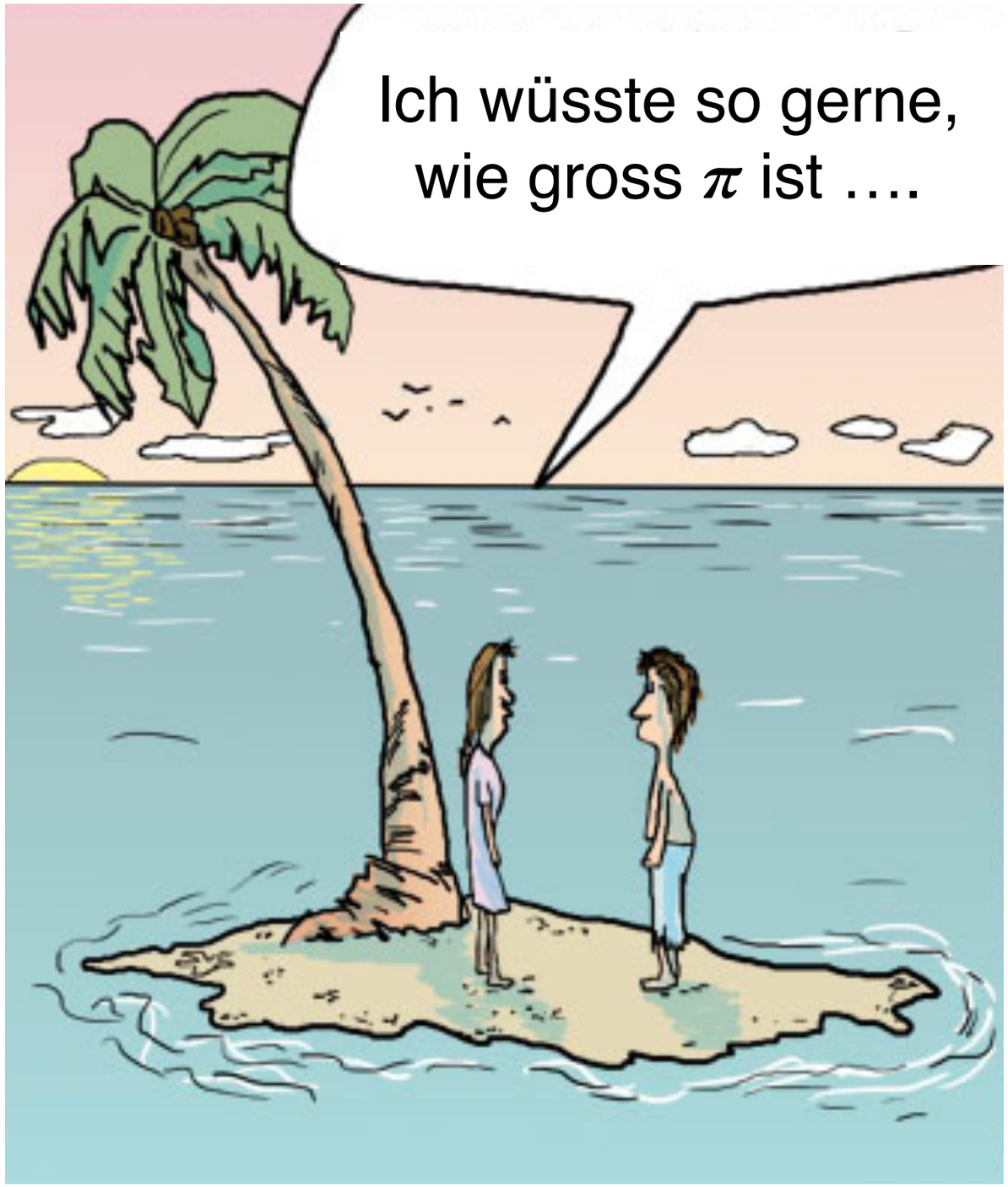
Zweiseitiger Fehler:

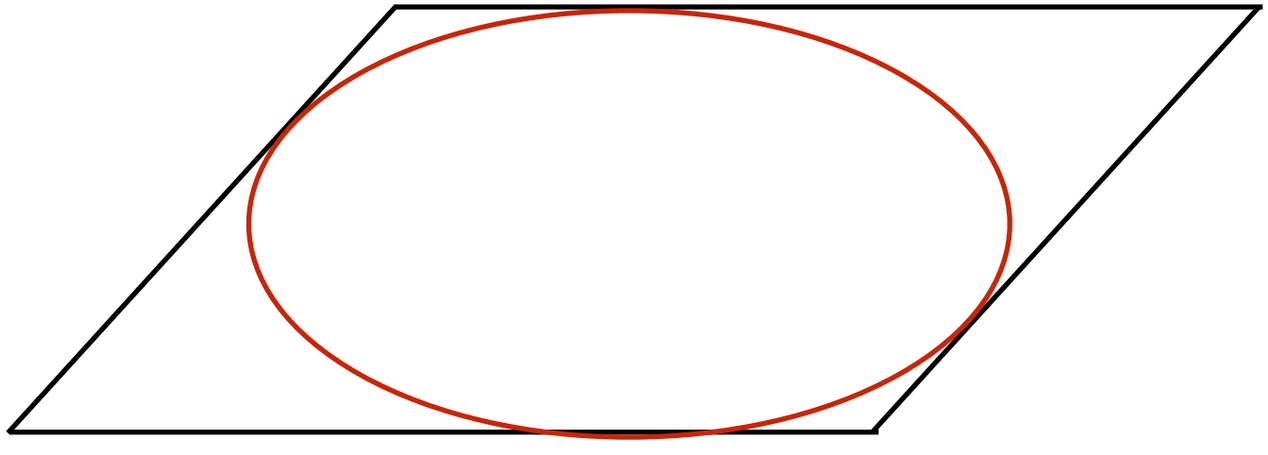
Pr[Antwort falsch] ≦ **1/2-ε** ∀ Instanzen

⇒ $4\,\varepsilon^{-2} \ln \delta^{-1}$ Wiederholungen reduzieren Fehler

Pr[Antwort falsch] ≦ δ

(Antwort: Mehrheit der gesehenen Antworten)


Ich wüsste so gerne,
wie gross π ist ….



Target-Shooting

Gegeben: zwei Mengen $S \subseteq U$

Aufgabe: bestimme $|S| / |U|$

Beispiel: $U = [-1,1] \times [-1,1]$

$S = \{(x,y) \in U : x^2 + y^2 \leqq 1\}$

$|S| / |U| = \boldsymbol{\pi} / 4$

Target-Shooting

Gegeben: zwei Mengen $S \subseteq U$

Aufgabe: bestimme $|S| / |U|$

Annahmen:
- *wir können ein Element aus U effizient zufällig gleichverteilt wählen*
- *es gibt eine effizient berechenbare Funktion*

$$\mathbb{I}_S(u) := \begin{cases} 1 & \text{falls } u \in S \\ 0 & \text{sonst} \end{cases}$$

Beispiel:

$U = [-1,1] \times [-1,1]$

$S = \{(x,y) \in U : x^2 + y^2 \leqq 1\}$

$|S| / |U| = \boldsymbol{\pi} / 4$

Target-Shooting

Gegeben: zwei Mengen $S \subseteq U$

Aufgabe: bestimme $|S| / |U|$

Annahmen:
- *wir können ein Element aus U effizient zufällig gleichverteilt wählen*
- *es gibt eine effizient berechenbare Funktion*

$$\mathbb{I}_S(u) := \begin{cases} 1 & \text{falls } u \in S \\ 0 & \text{sonst} \end{cases}$$

Target-Shooting

1: Wähle $u_1, \ldots, u_N \in U$ zufällig, gleichverteilt und unabhängig

2: **return** $N^{-1} \cdot \sum_{i=1}^{N} \mathbb{I}_S(u_i)$

Target-Shooting

TARGET-SHOOTING

1: Wähle $u_1, \ldots, u_N \in U$ zufällig, gleichverteilt und unabhängig

2: **return** $N^{-1} \cdot \sum_{i=1}^{N} \mathbb{I}_S(u_i)$

Notation: $Y_i := I_S(U_i)$ für alle i=1,…,N

$Y_1, \ldots, Y_N$ unabhängige Bernoulli-Variablen mit $\Pr[Y_i = 1] = |S|/|U|$

$$Y := \frac{1}{N} \sum_{i=1}^{N} Y_i = \frac{1}{N} \sum_{i=1}^{N} I_S(u_i)$$

Dann gilt: $\mathbb{E}[Y] = |S|/|U|$ … unabhängig von der Wahl von N

$$\mathrm{Var}[Y] = \frac{1}{N} \left(\frac{|S|}{|U|} - \left(\frac{|S|}{|U|} \right)^2 \right)$$

Target-Shooting

Target-Shooting

1: Wähle $u_1, \ldots, u_N \in U$ zufällig, gleichverteilt und unabhängig
2: **return** $N^{-1} \cdot \sum_{i=1}^{N} \mathbb{I}_S(u_i)$

Satz Seien $\delta, \varepsilon > 0$. Falls N „gross“ so ist die Ausgabe des Algorithmus Target-Shooting mit Wahrscheinlichkeit mindestens $1 - \delta$ im Intervall $\left[(1-\varepsilon)\frac{|S|}{|U|}, (1+\varepsilon)\frac{|S|}{|U|}\right]$.

Beweis: Chernoff-Schranke …

Analyse

Wir definieren Indikatorvariablen

$$Y_i := I_S(u_i) \quad \text{für} \quad i = 1, \ldots, N.$$

und die Zufallsvariable Y für die Ausgabe des Algorithmus:

$$Y := \frac{1}{N} \sum_{i=1}^{N} Y_i \ .$$

Y_i ...unabhängige Bernoulli-Variablen mit $\Pr[Y_i = 1] = \ p \ := \frac{|S|}{|U|}$.

$$\mathbb{E}[Y] = \frac{1}{N} \ Np \ = p \quad \text{und} \quad \mathrm{Var}[Y] = \frac{1}{N^2} \ Np(1-p) \ = \frac{1}{N}(p - p^2)$$

Analyse

$$\mathbb{E}[Y] = \frac{|S|}{|U|}$$

Wir wollen, dass

$$\left| Y - \frac{|S|}{|U|} \right| = |Y - \mathbb{E}[Y]|$$

mit grosser Wahrscheinlichkeit klein ist.

Analyse

$$\mathbb{E}[Y] = \frac{|S|}{|U|}$$

Wir wollen, dass

$$\left| Y - \frac{|S|}{|U|} \right| = |Y - \mathbb{E}[Y]|$$

mit grosser Wahrscheinlichkeit klein ist. Konkreter, für gegebene $\delta, \varepsilon > 0$, wollen wir

$$\Pr\left[\left| Y - \frac{|S|}{|U|} \right| \leq \varepsilon \frac{|S|}{|U|} \right] \geq 1 - \delta$$

Es ergibt sich eine Anforderung an N, abhängig vom δ, ε und $\frac{|S|}{|U|}$.

Analyse

Satz

Seien $\delta, \varepsilon > 0$. *Falls* $N \geq 3\frac{|U|}{|S|} \cdot \varepsilon^{-2} \cdot \ln(2/\delta)$, *ist die Ausgabe* Y *von* Target-Shooting *mit Wahrscheinlichkeit mindestens* $1 - \delta$ *im Intervall* $\left[\frac{|S|}{|U|} \pm \varepsilon\frac{|S|}{|U|}\right]$ *(multiplikativer Fehler von* $1 \pm \varepsilon$*).*

Oder, äquivalent,

$$\begin{aligned}\Pr\left[\left|Y - \frac{|S|}{|U|}\right| > \varepsilon\frac{|S|}{|U|}\right] &= \Pr\left[|Y - \mathbb{E}[Y]| > \varepsilon\mathbb{E}[Y]\right] \\ &= \Pr\left[|NY - \mathbb{E}[NY]| > \varepsilon\mathbb{E}[NY]\right] \leq \delta\end{aligned}$$

Da $NY = Y_1 + \cdots + Y_N$ eine Summe von N unabhängigen Bernoulli-Variablen ist, kann dies nun leicht mit Hilfe der uns bekannten Chernoff Schranken hergeleitet werden.

Herleitung der Schranke mittels Chernoff

Für welche N gilt

$$\Pr\left[|NY - \mathbb{E}[NY]| > \varepsilon \mathbb{E}[NY]\right] \leq \delta \ ?$$

$NY = \underbrace{Y_1 + \cdots + Y_N}_{:=Z}$, Summe von N unabh. Bernoulli-Variablen.

Nach bekannten Chernoff Schranken erhält man

$$\Pr\left[|Z - \mathbb{E}[Z]| > \varepsilon \mathbb{E}[Z]\right] \leq 2e^{-\varepsilon^2 \mathbb{E}[Z]/3} = 2e^{-\varepsilon^2 N|S|/(3|U|)}$$

Nun gilt $2e^{-\varepsilon^2 N|S|/(3|U|)} \leq \delta$ genau dann wenn

$$N \geq 3\frac{|U|}{|S|} \cdot \varepsilon^{-2} \cdot \ln(2/\delta).$$

Satz

Seien $\delta, \varepsilon > 0$. Falls $N \geq 3\frac{|U|}{|S|} \cdot \varepsilon^{-2} \cdot \ln(2/\delta)$, ist die Ausgabe Y von Target-Shooting *mit Wahrscheinlichkeit mindestens $1 - \delta$ im Intervall $\left[\frac{|S|}{|U|} \pm \varepsilon\frac{|S|}{|U|}\right]$ (multiplikativer Fehler von $1 \pm \varepsilon$).*

Randomisierte Algorithmen - Beispiele

Las-Vegas-Algorithmen

QuickSort, QuickSelect:

immer korrekt,
erwartete Laufzeit $O(n \log n)$ bzw $O(n)$

Monte-Carlo-Algorithmen

Primzahltest:

Fehlerw'keit „klein“,
Laufzeit $O(\text{poly}(\log n))$

Optimierungsalgorithmen

stabile Menge, Target-Shooting:

Ausgabe „nahe“ am Erwartungswert