Algorithmen und Wahrscheinlichkeit

Kapitel 2.4.2 + 2.7

Varianz und Konzentration

Zufallsvariable - Erwartungswert

Definition Zu einer Zufallsvariablen X definieren wir den *Erwartungswert* $\mathbb{E}[X]$ durch

$$\mathbb{E}[X] := \sum_{x \in W_X} x \cdot \Pr[X = x]$$

Welche Schlüsse können wir ziehen,

wenn wir den Erwartungswert einer Zufallsvariablen kennen?

Was wir gerne hätten:

$$\Pr[\ |X - \mathbb{E}[X]|\ \text{”gross”}\ ]\ =\ \text{”klein”}$$

Abweichung vom Erwartungswert

Was wir gerne hätten:

$$\Pr[\ |X - \mathbb{E}[X]|\ \text{"gross"}\ ]\ =\ \text{"klein"}$$

Beobachtung: stimmt sicher nicht immer … ☹

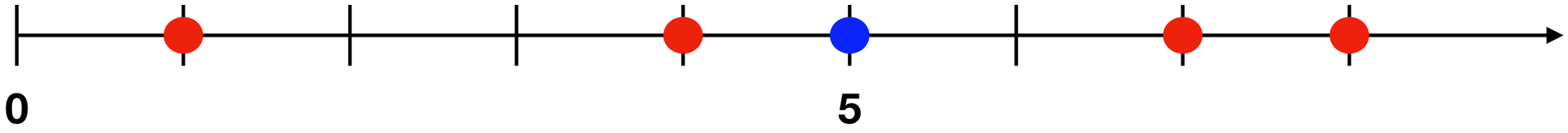
Beispiel:

$$\Pr[\ X\ =\ -10^{10}\ ]\ =\ \frac{1}{2}\ =\ \Pr[\ X\ =\ 10^{10}\ ]$$

Abweichung vom Erwartungswert

Was wir gerne hätten:

Pr[$|X - \mathbb{E}[X]|$ "gross"] = "klein"



Laplaceraum mit $\omega_1 = 1$, $\omega_2 = 4$, $\omega_3 = 7$, $\omega_4 = 8$ Erwartungswert $\mu = 5$

Was wir messen wollen ist die durchschnittliche Abweichung vom Erwartungswert

zum Beispiel $\frac{1}{4}\sum_i |\omega_i - \mu|$ oder $\frac{1}{4}\sum_i (\omega_i - \mu)^2$

Varianz $\mathbf{Var[X]} = \mathbb{E}[(X - \mu)^2]$, wobei $\mu := \mathbb{E}[X]$

Varianz

Definition Für eine Zufallsvariable X mit $\mu = \mathbb{E}[X]$ definieren wir die *Varianz* $\mathrm{Var}[X]$ durch

$$\mathrm{Var}[X] := \mathbb{E}[(X-\mu)^2] = \sum_{x \in W_X} (x-\mu)^2 \cdot \Pr[X = x].$$

Die Grösse $\sigma := \sqrt{\mathrm{Var}[X]}$ heisst *Standardabweichung* von X.

Satz Für eine beliebige Zufallsvariable X gilt

$$\mathrm{Var}[X] = \mathbb{E}[X^2] - \mathbb{E}[X]^2.$$

Beweis: $\mathbb{E}[(X-\mu)^2] = \mathbb{E}[X^2 - 2\mu X + \mu^2]$

$= \mathbb{E}[X^2] - 2\mu\mathbb{E}[X] + \mu^2 = \mathbb{E}[X^2] - \mu^2$

Varianz

Satz Für eine beliebige Zufallsvariable X und $a, b \in \mathbb{R}$ gilt

$$\mathrm{Var}[a \cdot X + b] = a^2 \cdot \mathrm{Var}[X].$$

Beweis: Definition der Varianz und Linearität des Erwartungswertes, siehe Skript

Rechenregeln für Momente

$$\mathbb{E}[X+Y] = \mathbb{E}[X] + \mathbb{E}[Y] \qquad \forall X, Y$$

$$\mathbb{E}[X \cdot Y] = \mathbb{E}[X] \cdot \mathbb{E}[Y] \qquad \forall X, Y \text{ unabhängig}$$

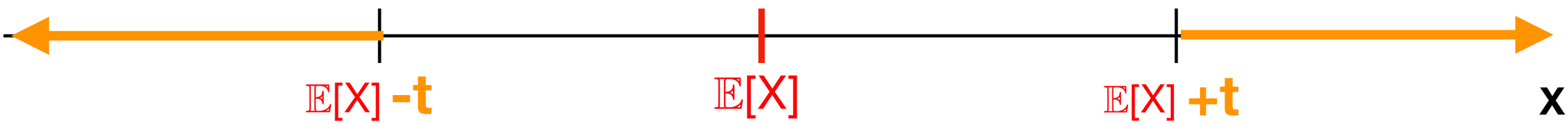
$$\mathrm{Var}[X+Y] = \mathrm{Var}[X] + \mathrm{Var}[Y] \qquad \forall X, Y \text{ unabhängig}$$

$$\mathrm{Var}[X \cdot Y] \neq \mathrm{Var}[\mathrm{X}] \cdot \mathrm{Var}[Y] \qquad \text{i.A.} \quad \text{(auch für unabhängige ZV)}$$

Tail Bounds

X Zufallsvariable

$$\Pr[|X - \mathbb{E}[X]| \geq t]$$



Chebyshev:

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

Markov Ungleichung

Satz *(Ungleichung von Markov)* Sei X eine Zufallsvariable, die nur nicht-negative Werte annimmt. Dann gilt für alle $t \in \mathbb{R}$ mit $t > 0$, dass

$$\Pr[X \geq t] \leq \frac{\mathbb{E}[X]}{t}.$$

Beweis:

$$\mathbb{E}[X] \overset{\text{Def}}{=} \sum_{x \in W_X} x \cdot \Pr[X = x]$$

$$\geq \cancel{\sum_{x \in W_X,\ x<t} x \cdot \Pr[X = x]} + \sum_{x \in W_X,\ x \geq t} \cancel{x} \; t \cdot \Pr[X = x]$$

$$\mathbb{E}[X] \geq t \cdot \sum_{x \in W_X,\ x \geq t} \Pr[X = x] = t \cdot \Pr[X \geq t]$$

Chebyshev Ungleichung

Satz *(Ungleichung von Chebyshev)* Sei X eine Zufallsvariable und $t \in \mathbb{R}$ mit $t > 0$. Dann gilt

$$\Pr[|X - \mathbb{E}[X]| \geq t] \leq \frac{\mathrm{Var}[X]}{t^2}.$$

Beweis:

$$|X - \mathbb{E}[X]| \geq t \iff (X - \mathbb{E}[X])^2 \geq t^2$$

$$Y := (X - \mathbb{E}[X])^2$$

dann gilt $Y \geq 0$ (da Y ein Quadrat ist)

daher können wir die Markov-Ungleichung anwenden:

$$\Pr[|X - \mathbb{E}[X]| \geq t] = \Pr[Y \geq t^2] \leq \frac{\mathbb{E}[Y]}{t^2} = \frac{\mathrm{Var}[X]}{t^2}$$

Markov & Chebyshev

Markov:

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

Chebyshev:

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

Insbesondere: ($\sigma := \sqrt{\mathrm{Var}[X]}$ Standardabweichung)

Für $t := \overset{C}{\cancel{10}}\sigma$

$$\Pr[|X - \mathbb{E}[X]| \geq \overset{C}{\cancel{10}}\sigma] \leq \frac{\mathrm{Var}[X]}{(\overset{C}{\cancel{10}}\sigma)^2} = \frac{1}{\overset{C^2}{\cancel{100}}} \qquad \forall\, X, \quad \forall\, C > 0$$

Kapitel 2.7

Abschätzen von Wahrscheinlichkeiten

Beispiele

Bernoulli-Verteilung

$$X \sim \text{Bernoulli}(p)$$

$$f_X(x) = \begin{cases} p & \text{für } x = 1, \\ 1 - p & \text{für } x = 0, \\ 0 & \text{sonst} \end{cases}$$

$$\mathbb{E}[X] = p$$

$$\text{Var}[X] = \mathbb{E}[(X - p)^2]$$

$$= p \cdot (1 - p)^2 + (1 - p) \cdot (0 - p)^2 = \ldots$$

$$= p(1 - p)$$

Beispiel: Werfen einer Münze, Indikator für Kopf

Binomialverteilung

$$X \sim \mathrm{Bin}(n, p).$$

$$f_X(x) = \begin{cases} \binom{n}{x} p^x (1-p)^{n-x}, & x \in \{0, 1, \ldots, n\} \\ 0, & \text{sonst.} \end{cases}$$

$$\mathbb{E}[X] = np \quad \text{und} \quad \mathrm{Var}[X] = np(1-p)$$

Beispiel: Werfen einer Münze n mal, X = Anzahl Kopf

$X \sim \mathrm{Bin}(n,p) \iff X = X_1 + \ldots + X_n$ mit $X_i \sim$ Bernoulli(p)
X_i unabhängig

Geometrische Verteilung

$$X \sim \mathrm{Geo}(p)$$

$$f_X(i) = \begin{cases} p(1-p)^{i-1} & \text{für } i \in \mathbb{N}, \\ 0 & \text{sonst.} \end{cases}$$

$$\mathbb{E}[X] = \frac{1}{p} \quad \text{und} \quad \mathrm{Var}[X] = \frac{1-p}{p^2}$$

Beispiel: Wiederholtes Werfen einer Münze,
X = Anzahl Würfe bis zum ersten Mal Kopf

Coupon Collector

Szenario: es gibt n verschiedene Bilder
in jeder Runde erhalten wir (gleichw'lich) eines der Bilder

X := Anzahl Runden bis wir alle n Bilder besitzen

X_i := Anzahl Runden in Phase i, $X_i \sim$ Geo((n-(i-1))/n)

$$\mathbb{E}[X] = \sum_{i=1}^{n} \mathbb{E}[X_i] = \sum_{i=1}^{n} \frac{n}{n-i+1} = n \cdot \sum_{i=1}^{n} \frac{1}{i} = n \cdot \ln n + O(n)$$

$$\mathrm{Var}[X] \overset{X_i \text{ unabh.}}{=} \sum_{i=1}^{n} \mathrm{Var}[X_i] \leq \ldots \leq n^2 \cdot \sum_{i=1}^{n} \frac{1}{i^2} \leq n^2 \cdot \frac{\pi^2}{6}$$

Coupon Collector

X_i := Anzahl Runden in Phase i, $X_i \sim$ Geo((n-(i-1))/n)

$$\mathbb{E}[X] = \sum_{i=1}^{n} \mathbb{E}[X_i] = \sum_{i=1}^{n} \frac{n}{n-i+1} = n \cdot \sum_{i=1}^{n} \frac{1}{i} = n \cdot \ln n + O(n)$$

$$\mathrm{Var}[X] \overset{X_i \text{ unabh.}}{=} \sum_{i=1}^{n} \mathrm{Var}[X_i] \leq \ldots \leq n^2 \cdot \sum_{i=1}^{n} \frac{1}{i^2} \leq n^2 \cdot \frac{\pi^2}{6}$$

Chebyshev:

$$\Pr[|X - \mathbb{E}[X]| \geq C \frac{\pi}{\sqrt{6}} n] \leq \frac{1}{C^2}$$

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	$4.14559 \cdot 10^{-73}$

Abschätzen von Wahrscheinlichkeiten

Chernoff:

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

Beispiel: $X \sim \mathrm{Bin}(n,1/2)$

dann gilt: $\mathbb{E}[X] = n/2$ $\mathrm{Var}[X] = n/4$ $\sigma = \sqrt{n}/2$

setze: $\delta := C/\sqrt{n}$ (d.h Abweichung $2C\sigma$)

Chebyshev: $\Pr[X \geq (1+\delta)\mathbb{E}[X]] \leq 1/C^2$

Chernoff: $\Pr[X \geq (1+\delta)\mathbb{E}[X]] \leq e^{-\frac{C^2}{6}}$

Chernoff Ungleichung

Satz (Chernoff-Schranke). Seien $X_1, \ldots, X_n$ unabhängige Bernoulli-verteilte Zufallsvariablen mit $\Pr[X_i = 1] = p_i$ and $\Pr[X_i = 0] = 1 - p_i$. Dann gilt für $X := \sum_{i=1}^{n} X_i$:

$$\Pr[X \geq (1+\delta)\mathbb{E}[X]] \leq e^{-\frac{1}{3}\delta^2 \mathbb{E}[X]} \quad \text{für alle } 0 < \delta \leq 1,$$

Beweisidee: $\forall t > 0$

$$\Pr[\, X \geq (1+\delta)\mathbb{E}[X] \,] \iff \Pr[\, e^{tX} \geq e^{t(1+\delta)\mathbb{E}[X]} \,]$$

$$\overset{\text{Markov Ungl.}}{\leq} \frac{\mathbb{E}[e^{tX}]}{e^{t(1+\delta)\mathbb{E}[X]}}$$

$$\leq \ \ldots \ \leq \ e^{-\frac{1}{3}\delta^2 \mathbb{E}[X]}$$

Rechnen und geschickte Wahl von t

Chernoff Ungleichung

Satz (Chernoff-Schranken). Seien $X_1, \ldots, X_n$ unabhängige Bernoulli-verteilte Zufallsvariablen mit $\Pr[X_i = 1] = p_i$ and $\Pr[X_i = 0] = 1 - p_i$. Dann gilt für $X := \sum_{i=1}^{n} X_i$:

(i) $\Pr[X \geq (1+\delta)\mathbb{E}[X]] \leq e^{-\frac{1}{3}\delta^2 \mathbb{E}[X]}$ für alle $0 < \delta \leq 1$,

(ii) $\Pr[X \leq (1-\delta)\mathbb{E}[X]] \leq e^{-\frac{1}{2}\delta^2 \mathbb{E}[X]}$ für alle $0 < \delta \leq 1$,

(iii) $\Pr[X \geq t] \leq 2^{-t}$ für $t \geq 2e\mathbb{E}[X]$.

Kapitel 2.8

Randomisierte Algorithmen

Algorithmen

Klassischer Algorithmus:

Wir beweisen:

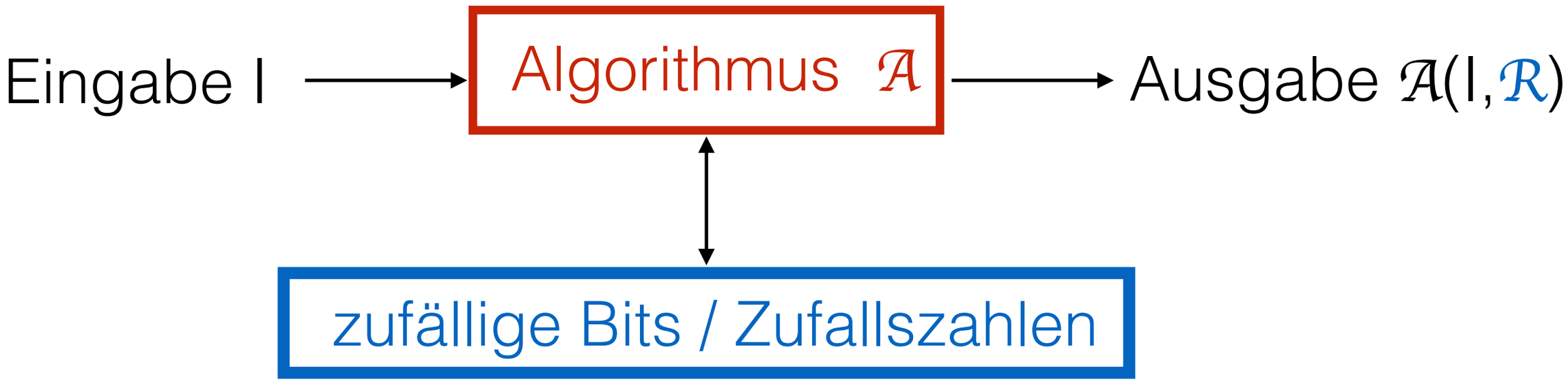
(1) Korrektheit:

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

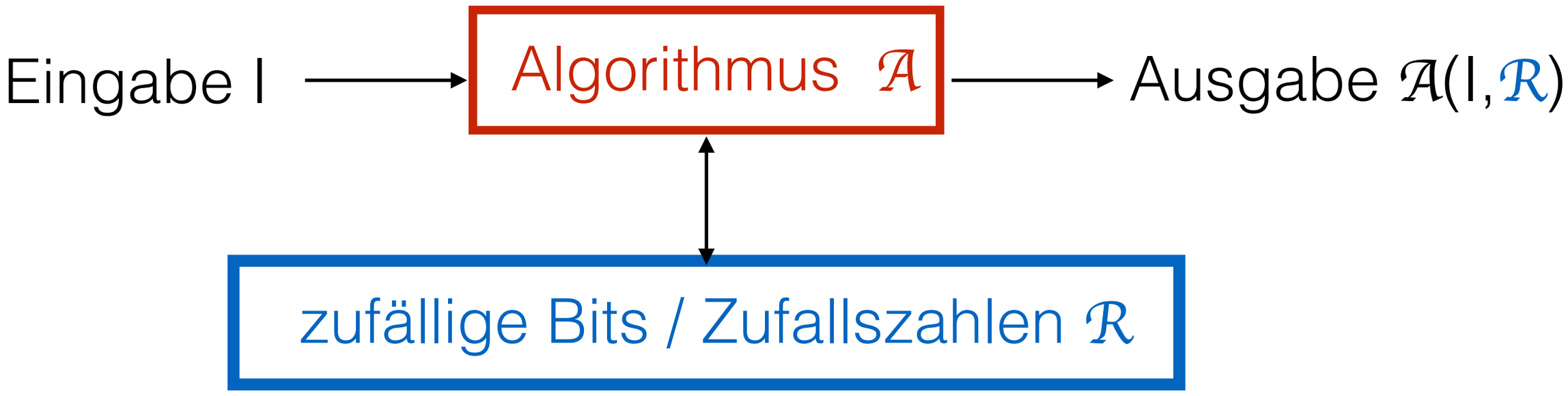


Eigenschaften:

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:

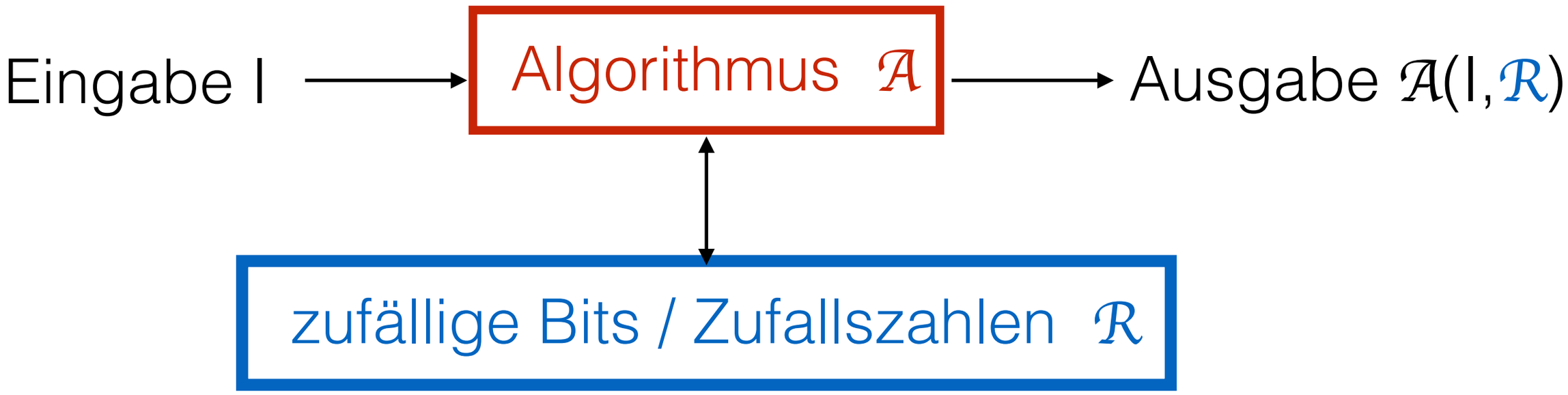
für alle Eingaben I gilt: $\Pr[\mathcal{A}(I,\mathcal{R})$ ist korrekt$] \geq \ldots$

(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$] \geq \ldots$

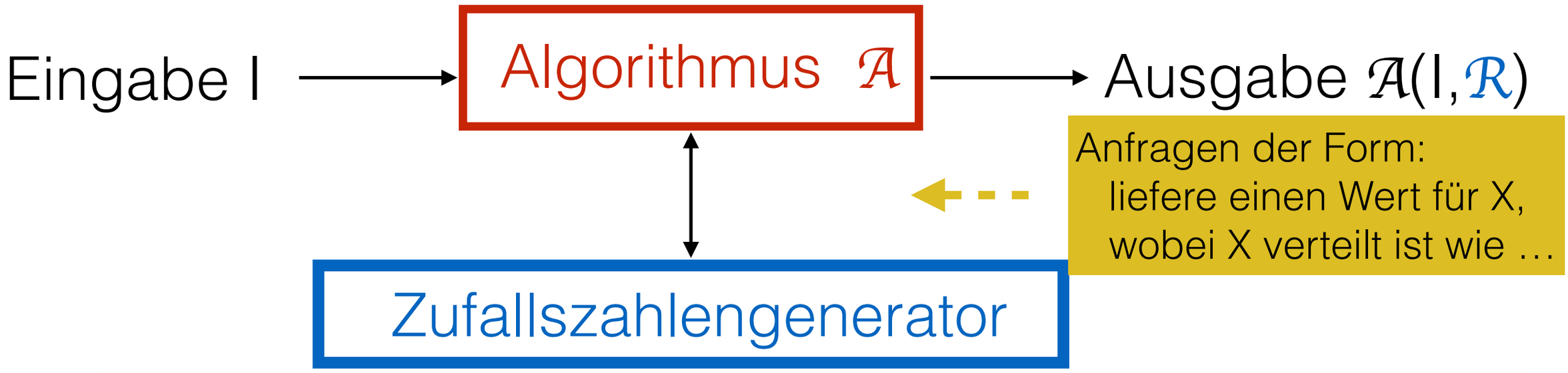
(2) Laufzeit:

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)

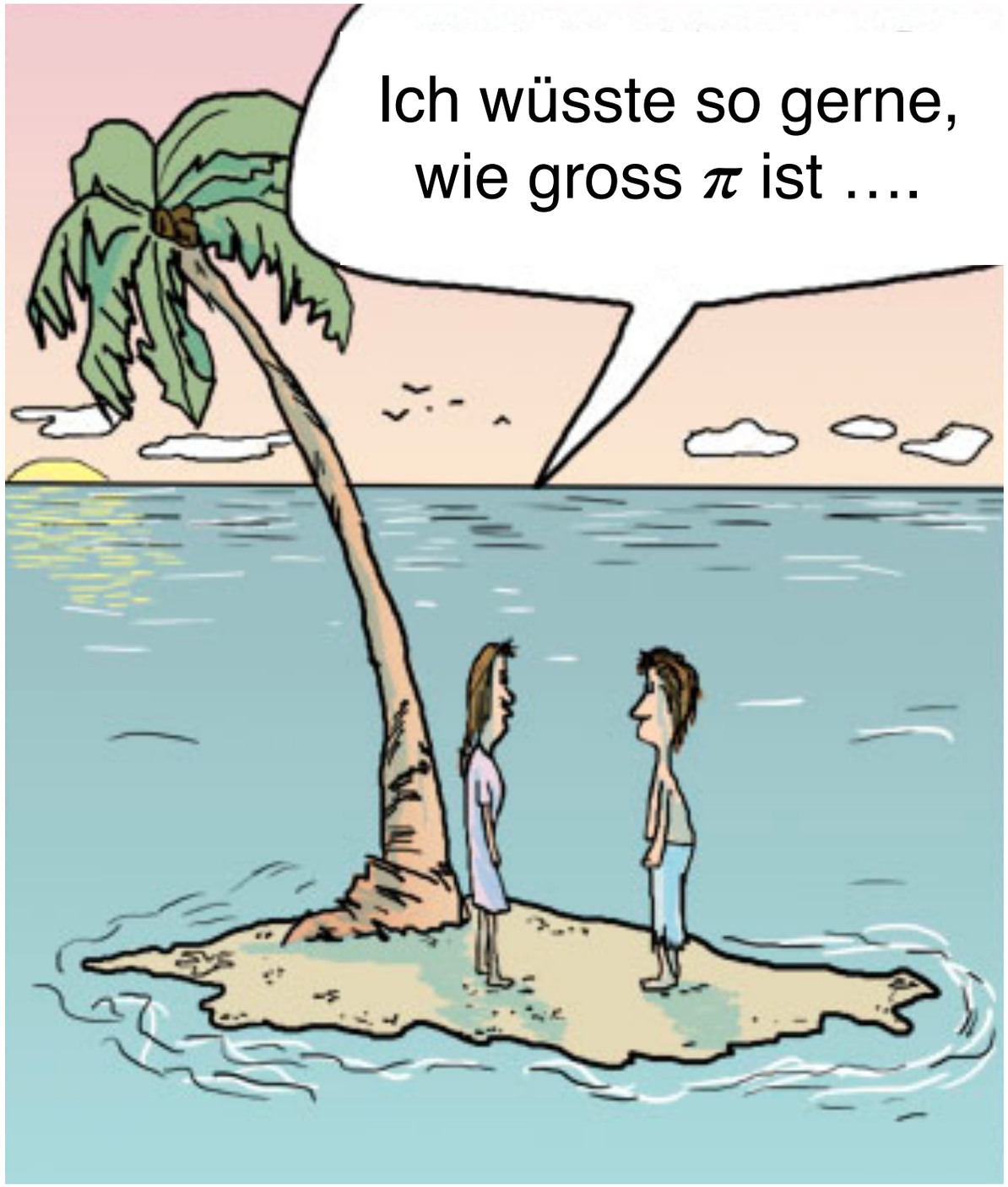
Zweiseitiger Fehler:

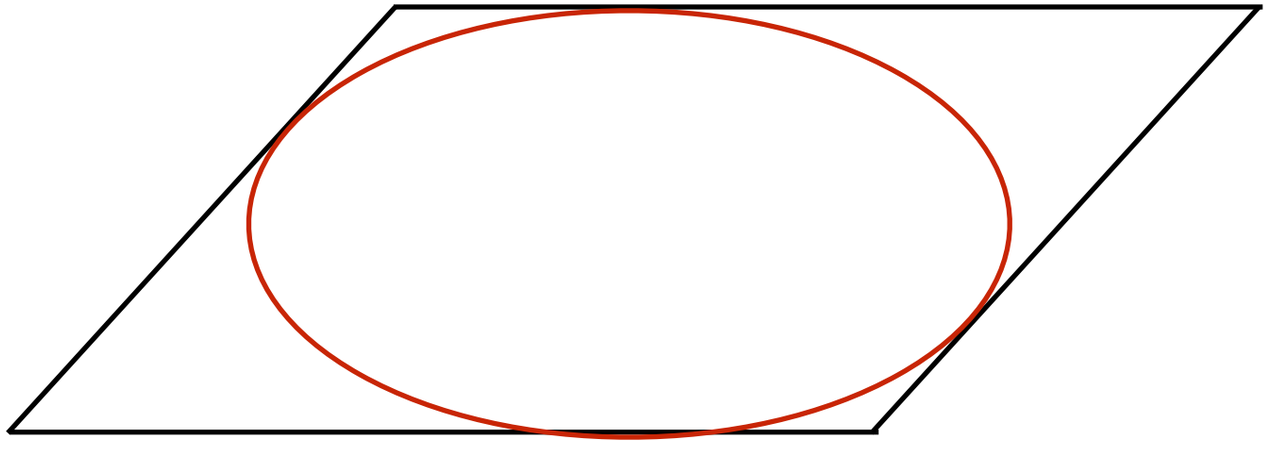
Pr[Antwort falsch] ≦ **1/2-ε** ∀ Instanzen

⇒ $4\,\varepsilon^{-2} \ln \delta^{-1}$ 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 …

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