

# **Algorithmen und Wahrscheinlichkeit**

**Woche 8**

**Ilya Maier**

# Minitest

# Randomisierte Algorithmen

Randomisierter Algorithmus : Eingabe  $I \rightarrow$  Algorithmus  $A$  mit Zufallszahlen  $R \rightarrow$  Ausgabe  $A(I, R)$

- deterministisch: selbe Eingabe, selber Output
- nicht-deterministisch: selbe Eingabe, nicht unbedingt selber Output

**Monte Carlo Algorithmus:** Primzahltest, Target-shooting  
→ Korrektheit ist die Zufallsvariable

- immer gleiche Laufzeit
- manchmal falsches Ergebnis

**Las Vegas Algorithmus:** Quicksort, Duplikate finden  
→ Laufzeit ist die Zufallsvariable  
→ Geometrisch verteilt mit  $p = \Pr[A(I) \neq "???"]$

- immer korrekte Antwort
- manchmal dauert zu lange /  
gibt nach einer bestimmter Zeit “???” aus

# Quicksort/Quickselect

**Quicksort:** sortiert den Array  
erwartete Laufzeit:  $\mathcal{O}(n \log n)$

---

QUICKSORT( $A, \ell, r$ )

---

```
1: if  $\ell < r$  then
2:    $p \leftarrow \text{Uniform}(\{\ell, \ell + 1, \dots, r\})$       ▷ wähle Pivotelement zufällig
3:    $t \leftarrow \text{PARTITION}(A, \ell, r, p)$ 
4:   QUICKSORT( $A, \ell, t - 1$ )
5:   QUICKSORT( $A, t + 1, r$ )
```

---

**Quickselect:** gibt das  $k$ -kleinste Element aus  
erwartete Laufzeit:  $\mathcal{O}(n)$

---

QUICKSELECT( $A, \ell, r, k$ )

---

```
1:  $p \leftarrow \text{Uniform}(\{\ell, \ell + 1, \dots, r\})$       ▷ wähle Pivotelement zufällig
2:  $t \leftarrow \text{PARTITION}(A, \ell, r, p)$ 
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
```

---

# Fehlerreduktion

**Las-Vegas:**

Sei  $A$  ein Las-Vegas-Algorithmus mit  $\Pr[A(I) \text{ ist korrekt}] \geq \epsilon$

Sei  $A_\delta$  für  $\delta > 0$  ein Algorithmus, der entweder die erste Ausgabe verschieden von ??? ausgibt  
oder der nach  $N = \lceil \epsilon^{-1} \cdot \ln(\delta^{-1}) \rceil$  Versuchen ??? ausgibt

dann gilt  $\Pr[A_\delta(I) \text{ ist falsch}] \leq \delta$

$\epsilon$	$\delta$	$N$
0.1	0.01	47
0.5	0.01	10
0.5	$10^{-80}$	369
0.9	$10^{-30}$	77

# Fehlerreduktion

## Monte-Carlo - Einseitiger Fehler:

$\Pr[A(I) = \text{Ja}] = 1$  für alle Ja-Instanzen  $I$   $\implies$  Wenn  $A(I) = \text{Ja}$ , dann könnte die Ausgabe falsch sein  
 $\Pr[A(I) = \text{Nein}] \geq \epsilon$  für alle Nein-Instanzen  $I$   $\implies$  Wenn  $A(I) = \text{Nein}$ , dann ist die Ausgabe immer korrekt

Sei  $A_\delta$  für  $\delta > 0$  ein Algorithmus, der entweder Nein ausgibt, sobald das erste Mal Nein vorkommt,  
oder der nach  $N = \lceil \epsilon^{-1} \cdot \ln(\delta^{-1}) \rceil$  Versuchen Ja ausgibt

dann gilt:

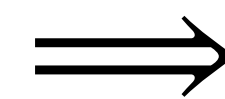
$\Pr[A_\delta(I) = \text{Ja}] = 1$  für alle Ja-Instanzen  $I$

$\Pr[A_\delta(I) = \text{Nein}] \geq 1 - \delta$  für alle Nein-Instanzen  $I$

## Monte-Carlo - Zweiseitiger Fehler:

$\Pr[A(I) \text{ ist korrekt}] \geq 0.5 + \epsilon$  für alle Instanzen  $I$

$A_\delta$  gibt die meiste Antwort aus nach  $N$  Wiederholungen



Für  $\delta > 0$ , wenn  $N = \lceil 4 \cdot \epsilon^{-2} \cdot \ln(\delta^{-1}) \rceil$

$\Pr[A_\delta(I) \text{ ist falsch}] \leq \delta$  für alle Instanzen  $I$

# Aufgaben