

# Übung 5 – Algorithmen II

Moritz Laupichler, Nikolai Maas – {moritz.laupichler, nikolai.maas}@kit.edu  
[https://algo2.iti.kit.edu/AlgorithmenII\\_WS23.php](https://algo2.iti.kit.edu/AlgorithmenII_WS23.php)

Institut für Theoretische Informatik - Algorithmik II

```
    result = current_weight;
    return true;
}

for( EdgeID eid = graph.edgeBegin( current ); eid != graph.edgeEnd( current ); ++eid ){
    const Edge & edge = graph.getEdge( eid );
    COUNTING( statistic_data.inc( DijkstraStatisticData::TOUCHED_EDGES ); )
    if( edge.forward ){
        COUNTING( statistic_data.inc( DijkstraStatisticData::RELAXED_EDGES ); )
        weight new_weight = edge.weight + current_weight;
        GUARANTEE( new_weight >= current_weight, std::runtime_error, "Weight overflow detected." );
        if( !priority_queue.isReached( edge.target ) ){
            COUNTING( statistic_data.inc( DijkstraStatisticData::SUCCESSFULLY_RELAXED_EDGES ); )
            COUNTING( statistic_data.inc( DijkstraStatisticData::REACHED_NODES ); )
            priority_queue.push( edge.target, new_weight );
        } else {
            if( priority_queue.getCurrentKey( edge.target ) > new_weight ){
                COUNTING( statistic_data.inc( DijkstraStatisticData::SUCCESSFULLY_RELAXED_NODES ); )
                priority_queue.decreaseKey( edge.target, new_weight );
            }
        }
    }
}
```

- Randomisierte Algorithmen
- Besprechung Übungsblatt 2

# Randomisierte Algorithmen

### ■ *Las Vegas Algorithmus*

- immer korrekte/optimale Lösung
- Laufzeit ist Zufallsvariable  
→ erwartete Laufzeit  $\mathbb{E}[T]$
- *Bsp.:* Quicksort

### ■ *Monte Carlo Algorithmus*

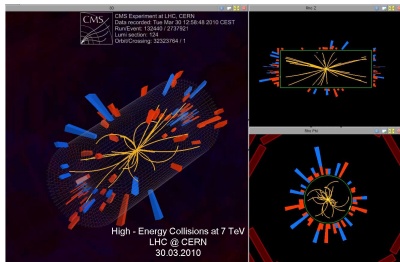
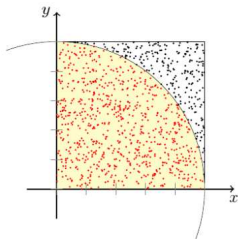
- falsche/suboptimale Lösung möglich  
→ mit Wahrscheinlichkeit  $p$
- deterministische Laufzeit
- *Bsp.:* Miller-Rabin Primzahltest

# Randomisierte Algorithmen

## Monte Carlo Simulation

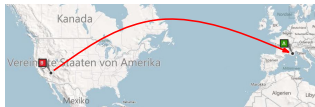
### ■ Monte Carlo Simulation

- nicht zu verwechseln mit *Monte Carlo Algorithmus!*
- Zufallsexperimente sehr oft ausführen Gesetz der großen Zahlen
- je länger / öfter ausgeführt, desto bessere Ergebnisse
  - Alternative zur analytischen Lösung
  - 3D-Rendering
  - Nachbildung komplexer Prozesse



# Randomisierte Algorithmen

Las Vegas  $\rightarrow$  Monte Carlo



**geg:** Las Vegas Algorithmus mit erwarteter Laufzeit  $\mathbb{E}[T] = f(n)$

**ges:** Monte Carlo Algorithmus mit Laufzeit  $\mathcal{O}(f(n))$ , Fehlerrate  $p$

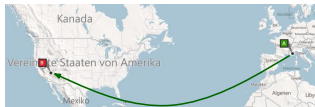
**Idee:** Abbruch nach Zeit  $\alpha f(n)$

- Ausgabe FALSCH, wenn Algorithmus abgebrochen wurde
- $\mathbb{P}[T > \alpha f(n)] \leq 1/\alpha$  Markov Ungleichung

$\rightarrow$  Monte Carlo Algorithmus mit Laufzeit  $\alpha f(n)$  und Fehlerrate  $p = 1/\alpha$

# Randomisierte Algorithmen

Monte Carlo → Las Vegas



**geg:** Monte Carlo Algorithmus mit Laufzeit  $\mathcal{O}(f(n))$ , Fehlerrate  $p$ ,  
Korrektheit in  $\mathcal{O}(g(n))$  prüfbar

**ges:** Las Vegas Algorithmus mit erwarteter Laufzeit  $\mathbb{E}[T]$

**Idee:** Wiederhole MC bis korrektes Ergebnis gefunden

■ Laufzeit  $T \leq i \cdot \mathcal{O}(f(n) + g(n))$   $i$  Schritte benötigt

■  $\mathbb{E}[T] \leq \mathbb{E}[i] \cdot \mathcal{O}(f(n) + g(n))$

$$\begin{aligned} \blacksquare \mathbb{E}[i] &= \sum_{k=1}^{\infty} k \cdot p^{k-1} \cdot (1-p) = \frac{1}{1-p} \\ &\text{nach } \sum_{k=1}^{\infty} kx^k = x/(1-x)^2, x < 1 \end{aligned}$$

→ Las Vegas Algorithmus mit erwarteter Laufzeit  $\mathbb{E}[T] \leq \frac{\mathcal{O}(f(n)+g(n))}{1-p}$

# Randomisierte Algorithmen

## Matrix-Matrix Multiplikation

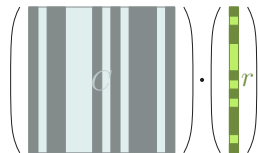
**Aufgabe:** Überprüfe, ob  $X \cdot Y = Z$  für Matrizen  $X, Y, Z$

■ deterministisch:

- berechne  $X \cdot Y$  und vergleiche mit  $Z$   
→ Laufzeit  $\mathcal{O}(n^3)$  (naiv),  $\mathcal{O}(n^{2.37})$  (best)

■ randomisiert:

- Wähle 0-1 Vektor  $r = (r_1, \dots, r_n)$  zufällig
- Wenn  $X(Yr) = Zr$  dann KORREKT, sonst FALSCH  
→ Laufzeit  $\mathcal{O}(n^2)$



**Behauptung:** Wenn  $XY \neq Z$  dann  $\mathbb{P}[XYr = Zr] \leq 0.5$



# Randomisierte Algorithmen

## Matrix-Matrix Multiplikation

**Behauptung:** Wenn  $XY \neq Z$  dann  $\mathbb{P}[XYr = Zr] \leq 0.5$

Definitionen:

- $A := XY, B := Z$

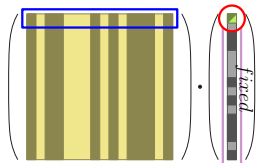
Annahme:  $A \neq B$ ; Wann ist dann  $Ar = Br$ ?

- $\exists i, j : A_{i,j} \neq B_{i,j}$
- Sei  $a := A_i$  und  $b := B_i$ ;  $i$ 'te Zeile von  $A$

$$\alpha := \sum_{k \neq j} a_k r_k \text{ und } \beta := \sum_{k \neq j} b_k r_k$$
$$\Rightarrow ar = \alpha + a_j r_j \text{ und } br = \beta + b_j r_j$$
$$\Rightarrow ar - br = (\alpha - \beta) + (a_j - b_j) r_j$$

- Annahme: Werte für  $r_{1 \dots j-1, j+1 \dots n}$  bereits festgelegt  
→ noch 2 Möglichkeiten für  $r_j$ , Gleichheit bei maximal einer  
 $\Rightarrow \mathbf{Pr}[ar - br = 0] = \mathbf{Pr}[r_j = \frac{\beta - \alpha}{a_j - b_j}] \leq \frac{1}{2}$

→ Fehlerrate  $p \leq 0.5$



# Randomisierte Algorithmen

## Matrix-Matrix Multiplikation

### Beschleunigung durch *probability boosting*

nur bei  $p \leq 0.5$  schnelle Konvergenz

- Wiederhole Test  $k$  mal mit unterschiedlicher Wahl von  $r$

- Ein Test liefert FALSCH

→  $AB \neq C$ , fertig

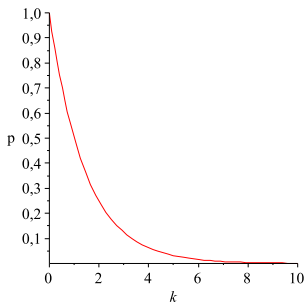
- Alle Tests liefern KORREKT

→ *false positive* mit Wahrscheinlichkeit

$$\mathbb{P}[ABr = Cr] \leq 0.5^k$$

→ Laufzeit  $\mathcal{O}(kn^2)$

linear längere Laufzeit bei exponentiell weniger Fehler



Müslipackungen enthalten jeweils eine von  $n$  verschiedenen Sammelkarten. Wie viele Packungen muss ich kaufen um alle Karten beisammenzuhaben?

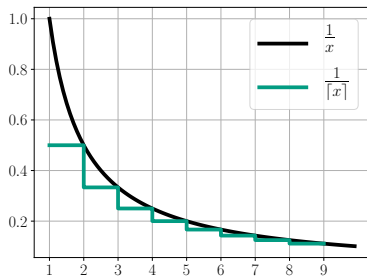
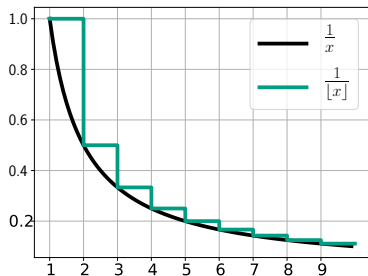
- $X = \#$  Packungen bis mind. eine von jeder Karte
- $X = \sum_{i=1}^n X_i$ , mit  $X_i = \#$  Packungen während ich  $i - 1$  Karten hatte
  - $X_i$  sind geometrische Zufallsvariablen mit  $p_i = 1 - \frac{i-1}{n}$
  - $\mathbb{E}[X_i] = \frac{1}{p_i} = \frac{n}{n-i+1}$
- $\mathbb{E}[X] = \mathbb{E}[\sum_{i=1}^n X_i] = \sum_{i=1}^n \mathbb{E}[X_i]$  Linearität des Erwartungswertes  
 $= \sum_{i=1}^n \frac{n}{n-i+1} = n \sum_{i=1}^n \frac{1}{i}$

# Harmonische Zahlen

$$\blacksquare H_n = \sum_{i=1}^n \frac{1}{i}$$

$$\ln n = \int_{x=1}^n \frac{1}{x} dx \leq \sum_{i=1}^n \frac{1}{i}$$

$$\sum_{i=2}^n \frac{1}{i} \leq \int_{x=1}^n \frac{1}{x} dx = \ln n$$



$$\blacksquare \ln n \leq H_n \leq \ln n + 1$$

Müslipackungen enthalten jeweils eine von  $n$  verschiedenen Sammelkarten. Wie viele Packungen muss ich kaufen um alle Karten beisammenzuhaben?

- $X = \#$  Packungen bis mind. eine von jeder Karte
- $X = \sum_{i=1}^n X_i$ , mit  $X_i = \#$  Packungen während ich  $i - 1$  Karten hatte
  - $X_i$  sind geometrische Zufallsvariablen mit  $p_i = 1 - \frac{i-1}{n}$
  - $\mathbb{E}[X_i] = \frac{1}{p_i} = \frac{n}{n-i+1}$
- $\mathbb{E}[X] = \mathbb{E}[\sum_{i=1}^n X_i] = \sum_{i=1}^n \mathbb{E}[X_i]$  Linearität des Erwartungswertes  
 $= \sum_{i=1}^n \frac{n}{n-i+1} = n \sum_{i=1}^n \frac{1}{i} = nH_n \leq n \ln n + n$

# Perfekte Hashfunktionen

- Repräsentation einer (statischen) Funktion
- Ausgabe  $r$ -bit Wert
- Kleine Menge an zulässigen Eingabewerten
- Rest undefiniert
- Lösung: Hashen und Gleichungssystem lösen  
Mit schlauer Wahl der Hashfunktion  $n \cdot (1 + o(1))$  Zeilen

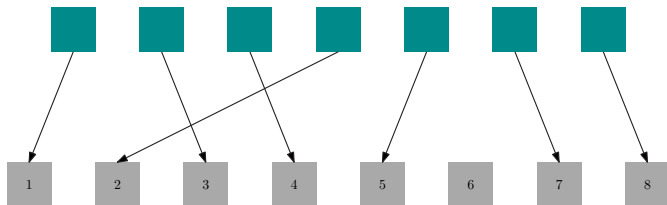
$x$	$f(x)$	$h(x)$	$f(x)$
Marvin	4 🎁 = 100	010001	100
Florian	2 🎁 = 010	100100	010
Tobias	4 🎁 = 100	111000	100
Daniel	0 🎁 = 000	000011	000
Sascha	6 🎁 = 101	111110	101

$$h(x) \cdot \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} = f(x)$$

# Perfekte Hashfunktionen

## Problemstellung

- Eingabe:  $n$  Objekte
- Gesucht: Zuordnung Objekt  $\rightarrow$  Zahl
- Abbildung soll injektiv sein  
z.B. Speicherplätze, Kurze Identifier





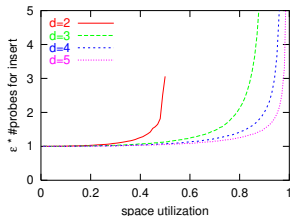
# Perfekte Hashfunktionen

## Cuckoo Hashing

- Cuckoo Hashing gibt jedem Objekt eine von 2 Positionen
- Baue Cuckoo-Hashtabelle
- Speichere je 1 Bit in Retrieval-Datenstruktur

4-äres Cuckoo Hashing:

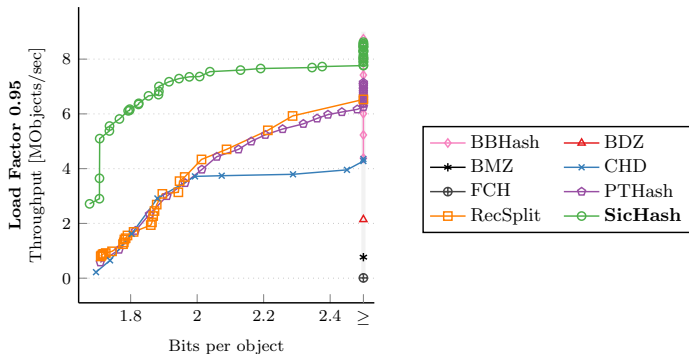
- ⇒ Retrieval-Datenstruktur mit 2 bits pro Eintrag
- ⇒ Perfekte Hashfunktion mit 2 bits pro Objekt



# Perfekte Hashfunktionen

## SicHash

- Irreguläres cuckoo hashing
  - Einige Objekte bekommen 2 Positionen, einige 4, und einige 8
- Mehrere Retrieval Datenstrukturen
  - 1-, 2- und 3-bit



# Übungsblatt 2

# Ende!



# Feierabend!