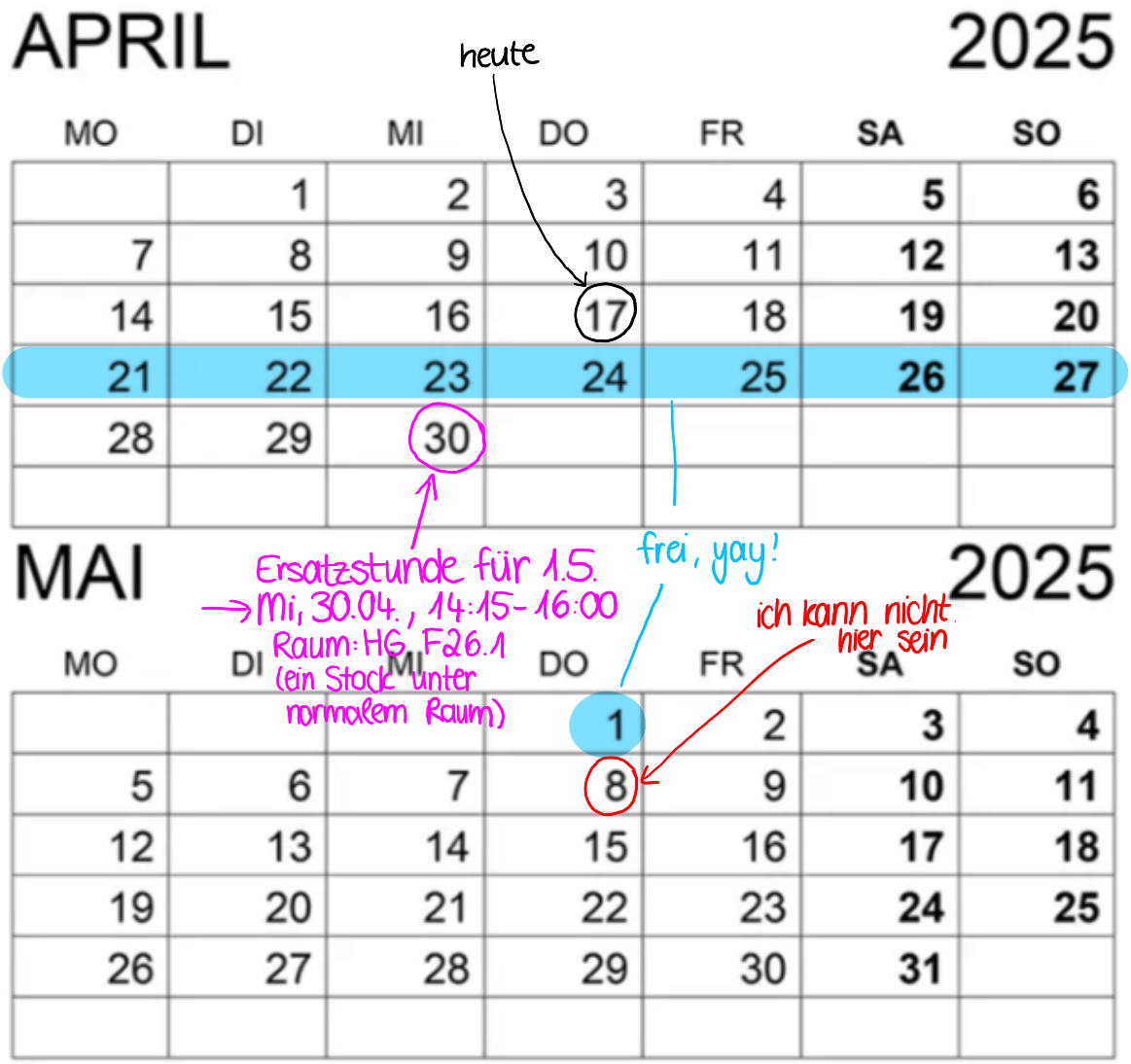
APRIL
2025
heute
MO DI MI DO FR SA SO
1 2 3 4 5 6
7 8 9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30
MAI
2025
Ersatzstunde für 1.5.
→ Mi, 30.04., 14:15-16:00
Raum: HG F26.1
(ein Stock unter normalem Raum)
frei, yay!
ich kann nicht hier sein
MO DI MI DO FR SA SO
1 2 3 4
5 6 7 8 9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31

TARGET SHOOTING

Gegeben: endliche Mengen S, U so dass $S \subseteq U$, $I_S : U \to \{0,1\}$ $I_S(u) = 1 \iff u \in S$

Gesucht: $\frac{|S|}{|U|}$

TARGET-SHOOTING

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

2: **return** $N^{-1} \cdot \sum_{i=1}^{N} I_S(u_i) = Y$

Sei $\varepsilon > 0$ beliebig klein. Wie groß muss N sein, damit der Algorithmus mit Wahrscheinlichkeit $\geq 1-\delta$ eine Antwort im Intervall $[(1-\varepsilon)\frac{|S|}{|U|}, (1+\varepsilon)\frac{|S|}{|U|}]$ ausgibt?

Satz 2.79. Seien $\delta, \varepsilon > 0$. Falls $N \geq 3\frac{|U|}{|S|} \cdot \varepsilon^{-2} \cdot \ln(2/\delta)$, 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: $Y_i = 1 \iff u_i \in S$

$Y_i := I_S(u_i)$ $\quad \Pr[Y_i = 1] = \frac{|S|}{|U|}$

$Y := \frac{1}{N}\sum_{i=1}^{n} Y_i$ $\quad E[Y] = \frac{|S|}{|U|}$

$Z := \sum_{i=1}^{n} Y_i = NY$

$\Pr[|Y - E[Y]| \geq \varepsilon \cdot E[Y]] \overset{!}{\leq} \delta$

$N \cdot |Y - E[Y]| = |NY - N \cdot E[Y]| = |Z - E[Z]|$

$\iff \Pr[|Z - E[Z]| \geq \varepsilon \cdot E[Z]] \overset{!}{\leq} \delta$

$Z \geq (1+\varepsilon)E[Z]$

$= \Pr[Z - E[Z] \geq \varepsilon \cdot E[Z]] + \Pr[Z - E[Z] \leq -\varepsilon \cdot E[Z]]$

$= \Pr[Z \geq (1+\varepsilon)E[Z]] + \Pr[Z \leq (1-\varepsilon)E[Z]]$

Chernoff: $\leq e^{-\frac{1}{3}\varepsilon^2 E[Z]} + e^{-\frac{1}{2}\varepsilon^2 E[Z]} \leq 2e^{-\frac{1}{3}\varepsilon^2 E[Z]} = 2e^{-\frac{1}{3}\varepsilon^2 N \cdot \frac{|S|}{|U|}} \overset{\text{Einsetzen von } N}{\leq} \delta$

PRIMZAHLTEST

übliches Finden von Primzahlen: Aussuchen einer random Zahl n gegebener Länge, dann testen, ob sie eine Primzahl ist

1) naive Option: alle Teiler bis $\sqrt{n}$ testen → ineffizient

2) random Zahl $a \in \{2, \dots, \sqrt{n}\}$ auswählen und schauen ob $\gcd(a,n) > 1$

$n = p \cdot q$, p und q sind zwei gleich große Primzahlen

Kleiner fermatscher Satz: Ist $n \in \mathbb{N}$ prim, so gilt für alle Zahlen $0 < a < n$ $a^{n-1} \equiv_n 1$.

$\Pr[\text{"prim"} \mid \text{nicht prim}] = \frac{|\{a \in [n-1] \mid a^{n-1} \equiv_n 1\}|}{n-1}$ n nicht prim

Carmichael-Zahl n: für alle teilerfremden Zahlen a gilt $a^{n-1} \equiv_n 1$, n nicht prim
kleinste Carmichael-Zahl: $561 = 3 \cdot 11 \cdot 17$

Miller-Rabin-Primzahltest

Idee: wenn n prim, dann ist $(\mathbb{Z}_n, +, *)$ ein Körper
$\rightarrow x^2 \equiv_n 1$ hat Lösungen $x \equiv_n 1$ und $x \equiv_n -1 \equiv_n n-1$

$n - 1 = d \cdot 2^k$, d ungerade

n prim $\Leftrightarrow \forall a \in \{1, \dots, n-1\}$ gilt $a^{n-1} = (a^d)^{2^k} \equiv_n 1$

$\rightarrow (a^d)^{2^{k-1}} \equiv_n 1 \quad \vee \quad (a^d)^{2^{k-1}} \equiv_n n-1$

- entweder gilt für jedes i mit $0 \le i \le k$ $(a^d)^{2^i} \equiv_n 1$
  oder es gibt ein i: $0 \le i \le k-1$ $(a^d)^{2^i} \equiv_n n-1$

MILLER-RABIN-PRIMZAHLTEST(n)

1: **if** $n = 2$ **then**
2: **return** 'Primzahl'
3: **else if** n gerade oder $n = 1$ **then**
4: **return** 'keine Primzahl'
5: Wähle $a \in \{2, 3, \dots, n-1\}$ zufällig und
6: berechne $k, d \in \mathbb{Z}$ mit $n - 1 = d2^k$ und d ungerade.
7: $x \leftarrow a^d \pmod n$
8: **if** $x = 1$ or $x = n - 1$ **then**
9: **return** 'Primzahl'
10: **repeat** $k - 1$ mal
11: $x \leftarrow x^2 \pmod n$
12: **if** $x = 1$ **then**
13: **return** 'keine Primzahl'
14: **if** $x = n - 1$ **then**
15: **return** 'Primzahl'
16: **return** 'keine Primzahl'

Laufzeit $O(\text{en}(n))$

$\Pr[\text{"prim"} \mid \text{nicht prim}] \le \frac{1}{4}$

HS17 Aufgabe 2c

4/80 Punkten ⇒ ca. 9 Minuten Zeit

(c) Beim Miller-Rabin-Primzahltest nutzt man einen Test $T : \mathbb{N} \times \mathbb{N} \to \{0, 1\}$ (eine Subroutine) mit den folgenden beiden Eigenschaften aus:

- $T(a, p) = 1$, falls $p > 2$ eine Primzahl ist und $a \in \{1, \ldots, p-1\}$.
- $\Pr[T(a, p) = 1] \leq 1/4$, falls $p > 2$ *keine* Primzahl ist und a uniform zufällig aus $\{1, \ldots, p-1\}$ ist.

Geben Sie einen möglichst effizienten Monte-Carlo-Algorithmus an, der entscheidet, ob ein $p > 2$ eine Primzahl ist, und dessen Fehlerwahrscheinlichkeit höchstens 10^{-4} ist. (Sie brauchen *nicht* darauf eingehen, wie der Test T aussieht, oder wie er implementiert wird.)

(4 Punkte)

Fehlerreduktionen:

- **Wiederholung MC:** Eine N-fache Wiederholung mit $N = 4\varepsilon^{-2} \ln \delta^{-1}$ steigert die Erfolgswahrscheinlichkeit eines Monte-Carlo-Algorithmus von $\frac{1}{2} + \varepsilon$ auf $\geq 1 - \delta$.
- **Wiederholung MC mit einseitigem Fehler:** Eine N-fache Wiederholung mit $N = \varepsilon^{-1} \ln \delta^{-1}$ steigert für einen Monte-Carlo-Algorithmus mit einseitigem Fehler die Erfolgswahrscheinlichkeit von ε auf $\geq 1 - \delta$.
- **Target Shooting:** Bestimmt der Target-Shooting-Algorithmus eine Menge $S \subseteq U$ mit $N \geq 3\frac{|U|}{|S|}\varepsilon^{-2} \ln(2/\delta)$ Versuchen, so ist die Ausgabe mit Wahrscheinlichkeit $\geq 1 - \delta$ im Intervall $\left[(1-\varepsilon)\frac{|S|}{|U|}, (1+\varepsilon)\frac{|S|}{|U|}\right]$.

N := 16
from i=1 ... N do
  wähle $a \in \{1, \ldots, p-1\}$ uniform zufällig
  if $T(a,p) = 0$ then return "p ist keine Primzahl"
return "p ist eine Primzahl"

$$N = \frac{1}{\varepsilon} \ln\left(\frac{1}{\delta}\right) = \frac{4}{3} \ln(10^4) = \frac{16}{3} \cdot \ln(10) \leq 16$$

Hashing

Hashfunktion $h: U \to [m]$ ($[m] = \{1,\dots,m\}$, m = #verfügbarer Speicherzellen), gegeben Datensatz $S := \{s_1,\dots,s_n\}$

→ sollte effizient berechenbar sein
→ optimalerweise $Pr[h(u)=i] = \frac{1}{m}$

$s_i = s_j \iff h(s_i) = h(s_j)$

#Kollisionen = #Duplikate + #unbeabsichtigte Kollisionen (= x)

Indikatorvariable X_{ij}:
$X_{ij} = 1 \iff$ es gibt Kollision mit Daten s_i und s_j, $s_i \neq s_j$

$$Pr[X_{ij}=1] = \begin{cases} \frac{1}{m} & \text{falls } s_i \neq s_j \\ 0 & \text{sonst} \end{cases}$$

$E[X_{ij}] \leq \frac{1}{m}$

$E[X] = \sum_{1 \leq i < j \leq n} E[X_{ij}] \leq \binom{n}{2} \frac{1}{m}$

Bloom-Filter

Wiederholung in $D = (s_1,\dots,s_n)$ ist ein $j \in [n]$, sodass $s_i = s_j$ für ein $i \in \{1,\dots,j-1\}$

gesucht: alle Wiederholungen in D

wähle $m,k \in \mathbb{N}$ und definiere k Hashfunktionen $h_i: U \to [m]$, $\forall i \in [k]$
Bitstring M der Länge m, zunächst $M=0$

Hashvektor $(x_1^{(i)},\dots,x_k^{(i)}) := (h_1(s_i),\dots,h_k(s_i))$

Beispiel: $D = (A, C, B, Z, F, H, C)$ (1, 2, 3, 4, 5, 6, 7)

$M = ($ 1 2 3 4 5 6 7 8 9 $)$
$M = ($ 1 1 0 1 1 1 1 0 1 $)$

Liste aller "Wiederholungen": $\mathcal{L} = \{Z, C\}$

Hash-Vektoren
$m = 9, k = 3$
$A \mapsto (5,4,1)$
$B \mapsto (4,6,1)$
$C \mapsto (5,1,2)$
$F \mapsto (1,9,4)$
$H \mapsto (4,7,5)$
$Z \mapsto (4,2,5)$

Analyse E[#falsche $\mathcal{L}$-Einträge]

IV X_i für i falsch $\mathcal{L}$, also $X_i = 1 \Leftrightarrow s_i$ nicht $(s_1, \dots, s_{i-1})$ und $m[x_j^{(i)}] = 1 \ \forall j \in [k]$ vor Bearbeiten von s_i

für ein j:

$$\Pr[m[x_j^{(i)}] = 0] = \left(1 - \frac{1}{m}\right)^{ke} \geq \left(1 - \frac{1}{m}\right)^{k(i-1)}$$

($ke \leftarrow$ #nicht voll)

$$\Pr[X_i = 1] = \Pr[\forall j : m[x_j^{(i)}] = 1] \leq \left(1 - \left(1 - \frac{1}{m}\right)^{k(i-1)}\right)^k$$

$$E[\#\text{Einträge in } \mathcal{L}] = \sum_{i=1}^{n} E[X_i] \leq n \cdot \left(1 - \left(1 - \frac{1}{m}\right)^{k(n-1)}\right)^k$$

$$\overset{1-x \leq e^{-x}}{\leq} n \left(1 - e^{-\frac{k(n-1)}{m}}\right)^k$$

Aufgabe 1 – *Hash tables* [Zusatzaufgabe]

Das folgende Problem ist als 'Element-Uniqueness-Problem' bekannt: Gegeben ist ein Array von Zahlen $a_1, \ldots, a_n$, zum Beispiel vom Typ `double`. Wir wollen herausfinden, ob zwei der Elemente des Arrays identisch sind, also ob es $1 \leq i < j \leq n$ gibt, so dass $a_i = a_j$.

Deterministisch lässt sich dieses Problem mithilfe von zwei for-loops in Zeit $O(n^2)$ lösen. Ein etwas geschickterer Ansatz, der einen schnellen Sortieralgorithmus verwendet, benötigt Zeit $O(n \ln n)$. In der Praxis lässt sich aber mit Hash-Tables ein noch schnellerer Ansatz finden.

Wir nehmen an, dass eine Hashfunktion h gegeben ist, die als Input Zahlen vom Typ `double` verwendet und als Ausgabe eine ganze Zahl zwischen (einschliesslich) 1 und n generiert. Die Idee ist, zunächst n Listen $L_1, \ldots, L_n$ zu generieren, wobei die Liste L_k alle Indizes $i \in \{1, \ldots n\}$ enthält sodass $h(a_i) = k$, und dann jede Liste L_k genauer zu untersuchen.

(a) Seien n, $a_1, \ldots, a_n$ und die Hashfunktion h gegeben. Sei

$$C = \{(i, j) : a_i \neq a_j \text{ und } h(a_i) = h(a_j)\}.$$

Beschreiben Sie einen Algorithmus der das Element-Uniqueness-Problem in Zeit $O(n + |C|)$ löst und $O(n)$ Speicherplatz benötigt.

Wir nehmen hierbei an, dass $h(x)$ in Zeit $O(1)$ berechnet werden kann.

(b) Angenommen, h wird zufällig gewählt, sodass für alle $a \neq b$ gilt

$$\Pr[h(a) = h(b)] = \frac{1}{n}.$$

Zeigen Sie, dass die erwartete Laufzeit des in (a) konstruierten Algorithmus $O(n)$ ist.

MINIQUIZFRAGEN

1) Wir können jeden Las Vegas Algorithmus mit erwarteter Laufzeit T in einen randomisierten Algorithmus mit Laufzeit höchstens $10T$ und Erfolgsw'keit mind. 0,9 umwandeln.

→ Richtig, $T' :=$ Laufzeit von $\mathcal{A}$, $T' \geq 0$, $\mathcal{A}'$ bricht nach $10T$ ab

$$Pr[T' \geq 10T] \leq \frac{E[T']}{10T} = \frac{T}{10T} = \frac{1}{10}$$

nur bis hier geschafft, den Rest besprechen wir vielleicht noch nächste Woche

2) Wir wollen überprüfen, ob ein Graph G auf $n \geq 3$ Knoten die Eigenschaft hat, dass alle Knoten höchstens Grad 10 haben. Wir schlagen folgenden Algorithmus $\mathcal{A}$ vor:

Mit Input G, wähle einen Knoten $v \in G$ zufällig gleichverteilt. Falls $d(v) \leq 10$ geben wir "Wahr" aus, ansonsten "Falsch".

(a) Falls G mind. einen Knoten mit $d(v) > 10$ hat, gilt $Pr[\mathcal{A}(G) = \text{"Falsch"}] \geq$

(b) Dies ist ein Algorithmus

(c) Falls G keinen Knoten v mit $d(v) > 10$ enthält, dann $Pr[\mathcal{A}(G) = \text{"Wahr"}] = 1$

3) Wenn wir wissen, dass 52633 eine Carmichael Zahl ist, folgt daraus, dass $11^{52632} \equiv_{52632} 1$ ist?

4) Betrachte den Fermat-Primzahltest

(a) Die Ausgabe "keine Primzahl" ist immer richtig

(b) Die Ausgabe "Primzahl" ist immer richtig

5) Wir wissen, dass $ggT(9973, 1001) = 1$ gilt. Folgt daraus, dass 9973 prim ist?