A&W 2026

G-13 Timo Stucki, LFW E 13

Week 8

Randomised Algorithms

Target-Shooting

Monte Carlo and Las Vegas Algorithms

Error Probability Reduction with Exercise

Infos

ETH zürich

Feel free to contact me:

- tistucki@student.ethz.ch
- Discord: timostucki

Material:

- timostucki.com

CodeX Recap?

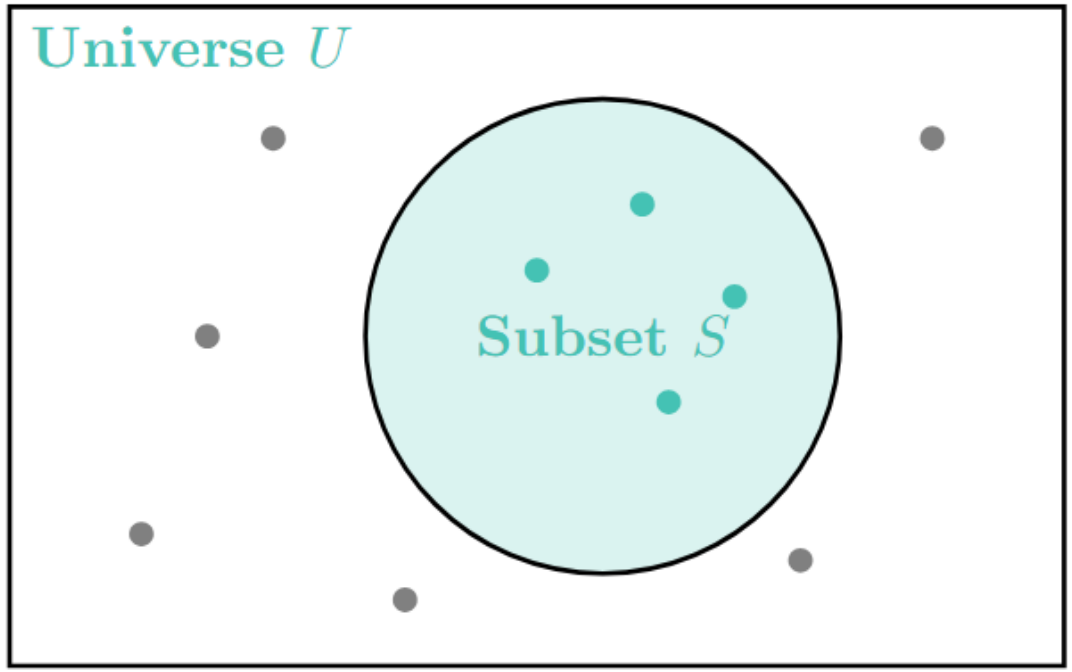
Randomised Algorithms

Target-Shooting

Target-Shooting

Example: How many people have a certain virus (i.e. are in S) ?

The goal is to estimate the ratio $\frac{|S|}{|U|}$ by sampling N points. Each point u_i is a "shot." If it hits the subset S, the indicator $I_S(u_i) = 1$; otherwise, it is 0.



Algorithm Output:

$$X = \frac{\text{Hits in } S}{\text{Total Shots } N}$$

Target-Shooting

Target-Shooting

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

Target-Shooting

Wir wollen den Target-Shooting-Algorithmus genauer analysieren. Sei dazu für $i \in [N]$

$$Y_i := I_S(u_i).$$

Nun sind wegen der unabhängigen und uniformen Wahl der u_i die Variablen $Y_1, \ldots, Y_N$ unabhängige Bernoulli-Variablen mit $\Pr[Y_i = 1] = |S|/|U|$ für jedes $i = 1, \ldots, N$. Für die Zufallsvariable

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

erhalten wir demnach $\mathbb{E}[Y] = |S|/|U|$, unabhängig von der Wahl von N.

Target-Shooting

Nun sind wegen der unabhängigen und uniformen Wahl der u_i die Variablen $Y_1, \ldots, Y_N$ unabhängige Bernoulli-Variablen mit $\Pr[Y_i = 1] = |S|/|U|$ für jedes $i = 1, \ldots, N$. Für die Zufallsvariable

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

erhalten wir demnach $\mathbb{E}[Y] = |S|/|U|$, unabhängig von der Wahl von N. Die Varianz

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

Target-Shooting

With some extra steps we can apply Chernoff's Inequalities (since the **n** Ys are independent Bernoulli distr. random variables):

Satz 2.70 (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 $t \geq 2e\mathbb{E}[X]$.

Target-Shooting

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 $\left[(1-\varepsilon)\frac{|S|}{|U|}, (1+\varepsilon)\frac{|S|}{|U|}\right]$.

Target-Shooting

On Cheatsheet! 🎉

Error Reduction

- **Monte Carlo Repetition:** An N-fold repetition of a Monte Carlo algorithm for $N = 4\varepsilon^{-2} \ln \delta^{-1}$ boosts the success probability from $\frac{1}{2} + \varepsilon$ to $\geq 1 - \delta$ (by outputing the majority answer).
- **One-sided MC Repetition:** An N-fold repetition for $N = \varepsilon^{-1} \ln \delta^{-1}$ boosts the success probability of a one-sided error Monte Carlo algorithm from ε to $\geq 1 - \delta$.
- **Target Shooting:** If the target-shooting algorithm identifies a set $S \subseteq U$ with $N \geq 3\frac{|U|}{|S|}\varepsilon^{-2} \ln(2/\delta)$ trials, then with probability $\geq 1 - \delta$ the output lies in the interval

$$\left[(1-\varepsilon)\frac{|S|}{|U|}, (1+\varepsilon)\frac{|S|}{|U|}\right].$$

Monte Carlo and Las Vegas Algorithms

Monte Carlo Algorithms

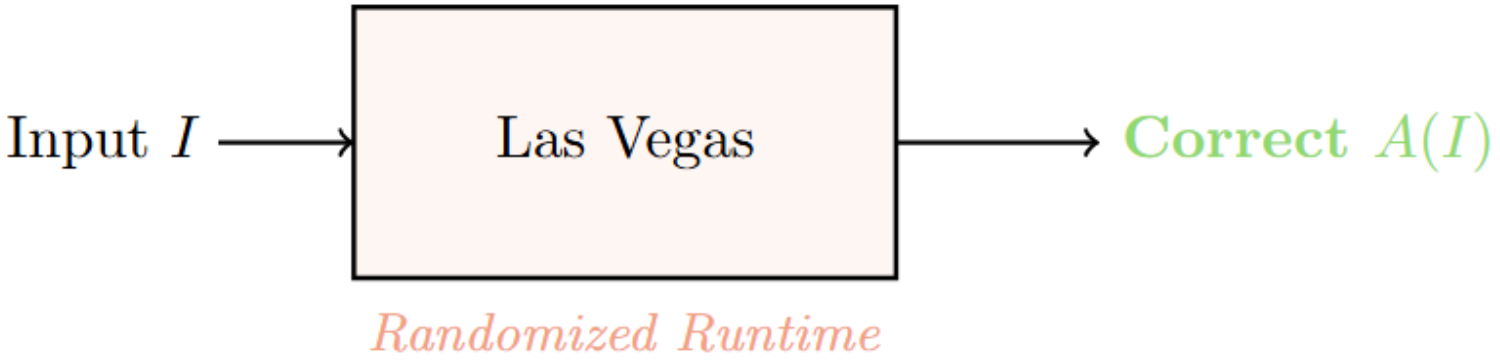
"Fast, but potentially flawed."

- The execution time is fixed and predictable (deterministic), but the result has a certain probability of being incorrect.
- **Intuition:** Think of a witness who always answers your question immediately but sometimes misremembers the facts.
- **Example: Primality Testing.** To check if a massive number is prime, the algorithm picks random witnesses. It always finishes quickly, but there is a tiny chance it calls a composite number "prime."

Las Vegas Algorithms

"Correct, but potentially slow."

- The result is **always correct** (or the algorithm admits it doesn't know with "???"), but the time it takes to find that result is a random variable.
- **Intuition:** Think of a researcher who refuses to give you an answer until they are 100% sure. They might find the answer in an hour, or it might take them a week.
- **Example: Randomized Quicksort.** By picking a random pivot, we avoid the worst-case $O(n^2)$ for almost all inputs. The result is always a perfectly sorted list, but the number of comparisons varies per run.



Monte Carlo and Las Vegas Algorithms

The Key Takeaway

- **Amplification:** We can turn a "weak" Monte Carlo algorithm into a "strong" one by running it multiple times to decrease the error probability.
- **Transformation:** A Las Vegas algorithm can be turned into a Monte Carlo algorithm by stopping it after a fixed time T and outputting "???" if it hasn't finished.

Error Probability Reduction

Error Probability Reduction

On Cheatsheet! 🎉

Error Reduction

- **Monte Carlo Repetition:** An N-fold repetition of a Monte Carlo algorithm for $N = 4\varepsilon^{-2} \ln \delta^{-1}$ boosts the success probability from $\frac{1}{2} + \varepsilon$ to $\geq 1 - \delta$ (by outputing the majority answer).
- **One-sided MC Repetition:** An N-fold repetition for $N = \varepsilon^{-1} \ln \delta^{-1}$ boosts the success probability of a one-sided error Monte Carlo algorithm from ε to $\geq 1 - \delta$.
- **Target Shooting:** If the target-shooting algorithm identifies a set $S \subseteq U$ with $N \geq 3\frac{|U|}{|S|}\varepsilon^{-2} \ln(2/\delta)$ trials, then with probability $\geq 1 - \delta$ the output lies in the interval

$$\left[(1-\varepsilon)\frac{|S|}{|U|}, (1+\varepsilon)\frac{|S|}{|U|}\right].$$

Error Probability Reduction

Satz 2.72. Sei $\mathcal{A}$ ein randomisierter Algorithmus, der nie eine falsche Antwort gibt, aber zuweilen '???' ausgibt, wobei

$$\Pr[\mathcal{A}(I) \text{ korrekt}] \geq \varepsilon.$$

Dann gilt für alle $\delta > 0$: bezeichnet man mit $\mathcal{A}_\delta$ den Algorithmus, der $\mathcal{A}$ solange aufruft, bis entweder ein Wert verschieden von '???' ausgegeben wird (und $\mathcal{A}_\delta$ diesen Wert dann ebenfalls ausgibt) oder bis $N = \varepsilon^{-1} \ln \delta^{-1}$-mal '???' ausgegeben wurde (und $\mathcal{A}_\delta$ dann ebenfalls '???' ausgibt), so gilt für den Algorithmus $\mathcal{A}_\delta$, dass

$$\Pr[\mathcal{A}_\delta(I) \text{ korrekt}] \geq 1 - \delta.$$

Error Probability Reduction

Satz 2.74. Sei $\mathcal{A}$ ein randomisierter Algorithmus, der immer eine der beiden Antworten JA oder NEIN ausgibt, wobei

$$\Pr[\mathcal{A}(I) = \text{JA}] = 1 \quad \text{falls } I \text{ eine JA-Instanz ist,}$$

und

$$\Pr[\mathcal{A}(I) = \text{NEIN}] \geq \varepsilon \quad \text{falls } I \text{ eine NEIN-Instanz ist.}$$

Dann gilt für alle $\delta > 0$: bezeichnet man mit $\mathcal{A}_\delta$ den Algorithmus, der $\mathcal{A}$ solange aufruft, bis entweder der Wert NEIN ausgegeben wird (und $\mathcal{A}_\delta$ dann ebenfalls NEIN ausgibt) oder bis $N = \varepsilon^{-1} \ln \delta^{-1}$-mal JA ausgegeben wurde (und $\mathcal{A}_\delta$ dann ebenfalls JA ausgibt), so gilt für alle Instanzen I

$$\Pr[\mathcal{A}_\delta(I) \text{ korrekt}] \geq 1 - \delta.$$

Error Probability Reduction

Satz 2.72. Sei $\mathcal{A}$ ein randomisierter Algorithmus, der nie eine falsche Antwort gibt, aber zuweilen '???' ausgibt, wobei

$$\Pr[\mathcal{A}(I) \text{ korrekt}] \geq \varepsilon.$$

Dann gilt für alle $\delta > 0$: bezeichnet man mit $\mathcal{A}_\delta$ den Algorithmus, der $\mathcal{A}$ solange aufruft, bis entweder ein Wert verschieden von '???' ausgegeben wird (und $\mathcal{A}_\delta$ diesen Wert dann ebenfalls ausgibt) oder bis $N = \varepsilon^{-1} \ln \delta^{-1}$-mal '???' ausgegeben wurde (und $\mathcal{A}_\delta$ dann ebenfalls '???' ausgibt), so gilt für den Algorithmus $\mathcal{A}_\delta$, dass

$$\Pr[\mathcal{A}_\delta(I) \text{ korrekt}] \geq 1 - \delta.$$

Satz 2.74. Sei $\mathcal{A}$ ein randomisierter Algorithmus, der immer eine der beiden Antworten JA oder NEIN ausgibt, wobei

$$\Pr[\mathcal{A}(I) = \text{JA}] = 1 \quad \text{falls } I \text{ eine JA-Instanz ist,}$$

und

$$\Pr[\mathcal{A}(I) = \text{NEIN}] \geq \varepsilon \quad \text{falls } I \text{ eine NEIN-Instanz ist.}$$

Dann gilt für alle $\delta > 0$: bezeichnet man mit $\mathcal{A}_\delta$ den Algorithmus, der $\mathcal{A}$ solange aufruft, bis entweder der Wert NEIN ausgegeben wird (und $\mathcal{A}_\delta$ dann ebenfalls NEIN ausgibt) oder bis $N = \varepsilon^{-1} \ln \delta^{-1}$-mal JA ausgegeben wurde (und $\mathcal{A}_\delta$ dann ebenfalls JA ausgibt), so gilt für alle Instanzen I

$$\Pr[\mathcal{A}_\delta(I) \text{ korrekt}] \geq 1 - \delta.$$

- **One-sided MC Repetition:** An N-fold repetition for $N = \varepsilon^{-1} \ln \delta^{-1}$ boosts the success probability of a one-sided error Monte Carlo algorithm from ε to $\geq 1 - \delta$.

Error Probability Reduction

On Cheatsheet! 🎉

Error Reduction

- **Monte Carlo Repetition:** An N-fold repetition of a Monte Carlo algorithm for $N = 4\varepsilon^{-2} \ln \delta^{-1}$ boosts the success probability from $\frac{1}{2} + \varepsilon$ to $\geq 1 - \delta$ (by outputing the majority answer).
- **One-sided MC Repetition:** An N-fold repetition for $N = \varepsilon^{-1} \ln \delta^{-1}$ boosts the success probability of a one-sided error Monte Carlo algorithm from ε to $\geq 1 - \delta$.
- **Target Shooting:** If the target-shooting algorithm identifies a set $S \subseteq U$ with $N \geq 3\frac{|U|}{|S|}\varepsilon^{-2} \ln(2/\delta)$ trials, then with probability $\geq 1 - \delta$ the output lies in the interval

$$\left[(1-\varepsilon)\frac{|S|}{|U|}, (1+\varepsilon)\frac{|S|}{|U|}\right].$$

Error Probability Reduction

Satz 2.75. Sei $\varepsilon > 0$ und $\mathcal{A}$ ein randomisierter Algorithmus, der immer eine der beiden Antworten JA oder NEIN ausgibt, wobei

$$\Pr[\mathcal{A}(\mathrm{I}) \text{ korrekt}] \geq 1/2 + \varepsilon.$$

Dann gilt für alle $\delta > 0$: bezeichnet man mit $\mathcal{A}_\delta$ den Algorithmus, der $\mathrm{N} = 4\varepsilon^{-2} \ln \delta^{-1}$ unabhängige Aufrufe von $\mathcal{A}$ macht und dann die Mehrheit der erhaltenen Antworten ausgibt, so gilt für den Algorithmus $\mathcal{A}_\delta$, dass

$$\Pr[\mathcal{A}_\delta(\mathrm{I}) \text{ korrekt}] \geq 1 - \delta.$$

Error Probability Reduction

On Cheatsheet! 🎉

Error Reduction

- **Monte Carlo Repetition:** An N-fold repetition of a Monte Carlo algorithm for $N = 4\varepsilon^{-2} \ln \delta^{-1}$ boosts the success probability from $\frac{1}{2} + \varepsilon$ to $\geq 1 - \delta$ (by outputing the majority answer).
- **One-sided MC Repetition:** An N-fold repetition for $N = \varepsilon^{-1} \ln \delta^{-1}$ boosts the success probability of a one-sided error Monte Carlo algorithm from ε to $\geq 1 - \delta$.
- **Target Shooting:** If the target-shooting algorithm identifies a set $S \subseteq U$ with $N \geq 3\frac{|U|}{|S|}\varepsilon^{-2} \ln(2/\delta)$ trials, then with probability $\geq 1 - \delta$ the output lies in the interval

$$\left[(1-\varepsilon)\frac{|S|}{|U|}, (1+\varepsilon)\frac{|S|}{|U|}\right].$$

Error Probability Reduction

Satz 2.76. Sei $\varepsilon > 0$ und $\mathcal{A}$ ein randomisierter Algorithmus für ein Maximierungsproblem, wobei gelte:

$$\Pr[\mathcal{A}(I) \geq f(I)] \geq \varepsilon.$$

Dann gilt für alle $\delta > 0$: bezeichnet man mit $\mathcal{A}_\delta$ den Algorithmus, der $N = \varepsilon^{-1} \ln \delta^{-1}$ unabhängige Aufrufe von $\mathcal{A}$ macht und die *beste* der erhaltenen Antworten ausgibt, so gilt für den Algorithmus $\mathcal{A}_\delta$, dass

$$\Pr[\mathcal{A}_\delta(I) \geq f(I)] \geq 1 - \delta.$$

(Für Minimierungsprobleme gilt eine analoge Aussage wenn wir „$\geq f(I)$“ durch „$\leq f(I)$“ ersetzen.)

Error Probability Reduction

On Cheatsheet! 🎉

Error Reduction

- **Monte Carlo Repetition:** An N-fold repetition of a Monte Carlo algorithm for $N = 4\varepsilon^{-2} \ln \delta^{-1}$ boosts the success probability from $\frac{1}{2} + \varepsilon$ to $\geq 1 - \delta$ (by outputing the majority answer).
- **One-sided MC Repetition:** An N-fold repetition for $N = \varepsilon^{-1} \ln \delta^{-1}$ boosts the success probability of a one-sided error Monte Carlo algorithm from ε to $\geq 1 - \delta$.
- **Target Shooting:** If the target-shooting algorithm identifies a set $S \subseteq U$ with $N \geq 3\frac{|U|}{|S|}\varepsilon^{-2} \ln(2/\delta)$ trials, then with probability $\geq 1 - \delta$ the output lies in the interval

$$\left[(1-\varepsilon)\frac{|S|}{|U|}, (1+\varepsilon)\frac{|S|}{|U|}\right].$$

Exercise S9

Reducing the Error of Monte Carlo Algorithms

Exercise S9.1 – *Reducing the Error of Monte Carlo Algoriths*

Alice, Bob and Claire have learned that Monta Carlo algorithms can often be improved by running them multiple times. They are each given a blackbox [1] Monte Carlo algorithm $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$ with a certain bound on the error. The task is to construct new algorithms $\mathcal{A}$_*improved*, $\mathcal{B}$_*improved* resp. $\mathcal{C}$_*improved* (using $\mathcal{A}$, $\mathcal{B}$ resp. $\mathcal{C}$ as subroutine) with a success probability of at least 99%. Try to keep the number of calls to the subroutines small.

(a) Given a graph (V, E), Alice tries to find a vertex set $\emptyset \neq S \subsetneq V$ that minimizes $|\delta(S)|$. She is given a Monte Carlo Algorithm $\mathcal{A}$ that, given a graph, returns some vertex set $\emptyset \neq S \subsetneq V$. With probability at least $p_{\mathcal{A}} \geq 1/\binom{n}{2}$ this set minimizes $|\delta(S)|$. Use this algorithm $\mathcal{A}$, to construct another algorithm $\mathcal{A}$_*improved* for the same problem, that has a success probability of at least 99%. *Hint:* $1 + x \leq e^x$

(b) Bob wants to check if a number is prime. He already knows a Monte Carlo algorithm $\mathcal{B}$ which takes as input a natural number N. If N is prime, the algorithm always returns 'prime'. If N is not prime, the algorithm returns 'not a prime' with probability $p_B \geq 1/2$. Use Bobs algorithm $\mathcal{B}$, to construct another algorithm $\mathcal{A}$_*improved* for the same problem, that has a success probability of at least 99%.

(c) Claire chose the most difficult problem. She wants to determine if a given graph has ans Hamiltonian cycle. She thought of an algorithm $\mathcal{C}$ that, given a graph, out puts 'YES' or 'NO'. If a graoh G has (resp. does not have) a Hamiltonian cycle, the algorithms $\mathcal{C}(G)$ returns 'YES' (resp. 'No') with probability $p_{\mathcal{C}} \geq 3/4$. Help Claire to find an algorithm C_*improved* thatis correct with a probability of at least 99%.

(d) Unfortunately, you forgot what Clairs initial algorithm $\mathcal{C}$ was. Thus, you have to find an initial algorithm yourself. Describe a fast Monte Carlo algorithm that, given a graph, outputs the correct answer 'YES' or 'NO' with probability 1/2. Can you boost this algorithm to a success probability of 99%?

(e) **Difficult**: Assume you are given the Monte Carlo algorithm $\mathcal{C}$ from part (c). Can you construct an algorithm that not only determines whether there is a Hamiltonian cycle, but even computes one if it exists. Of course, again only with a success probability of 99%.

Primality Test

Preview

Primality Test

Satz 2.77 (Kleiner fermatscher Satz). Ist $n \in \mathbb{N}$ prim, so gilt für alle Zahlen $0 < a < n$

$$a^{n-1} \equiv 1 \mod n.$$

Primality Test

If n is a **prime number**, then the set of integers $\{0, 1, \ldots, n-1\}$ forms a **Field** under addition and multiplication modulo n. (remember Dsicrete Mathematics lecture)

In any field, a polynomial of degree d can have at most d roots. Specifically, for the quadratic congruence:

$$x^2 \equiv 1 \pmod{n}$$

for $0 < x < n$, there are exactly **two solutions** if n is prime:

- $x = 1$
- $x = n - 1$ (which is equivalent to $-1 \pmod{n}$)

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 $a \in \{2, 3, \ldots, n-1\}$ 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'

Primality Test

The **Miller-Rabin** algorithm is a **Monte Carlo algorithm** used to determine whether a given integer n is prime.

- **Runtime:** $O(\ln n)$.
- **Correctness for Primes:** If n is a prime number, the algorithm always returns the answer **'Prime'**.
- **Accuracy for Composite Numbers:** If n is a composite number, the algorithm identifies it correctly by returning **'not a prime'** with a probability of at least $3/4$.
- **Error Reduction:** We can make the error probability (the chance of incorrectly labeling a composite number as prime) **arbitrarily small** by repeating the test multiple times, according to Satz 2.74.