Theoretische Informatik

Probabilistische Algorithmen

Technische Hochschule Rosenheim

SS 2019

Prof. Dr. J. Schmidt

Inhalt

- Pseudo-Zufallszahlen
- Monte-Carlo Methoden
  - Beispiel: Probabilistische Primzahltests

Probabilistische Algorithmen liefern

Technische Hochschule **Rosenheim**

- entweder eine Approximation des tatsächlichen Ergebnisses
  - zufälliges Abtasten des Wertebereichs
  - sehr viele Abtastpunkte
  - z.B.: Berechnung von Integralen, Computergrafik (Photon Mapping)
- oder eine Aussage, die nur mit einer gewissen Wahrscheinlichkeit korrekt ist
  - z.B. Primzahltest
    - Ergebnis: Zahl ist nicht prim → immer richtig
    - Ergebnis: Zahl ist prim → nur mit sehr hoher Wahrscheinlichkeit richtig
- in vielen Fällen wird die Berechnung erst dadurch praktikabel
- es werden Zufallszahlen benötigt

Zufallszahlen

Technische Hochschule **Rosenheim**

- **echte** Zufallszahlen
  - verwenden natürliche zufällige Prozesse
    - radioaktiver Zerfall, Rauschen von elektronischen Bauelementen, quantenphysikalische Prozesse
  - können von normalen Rechnern nicht generiert werden (von Quantencomputern schon)
- **Pseudo**zufallszahlen
  - algorithmische Berechnung von „Zufallszahlen“
  - typischerweise
    - iterative Berechnung
    - wiederholen sich nach bestimmter Anzahl Zahlen
  - deterministisch: bei gleichem Startwert ist Folge exakt reproduzierbar
  - Initialisierung z.B. durch
    - Systemzeit
    - aktueller Zustand des Speichers, Register, Festplattenposition, …

Verteilungen

Technische Hochschule Rosenheim

- praktisch wichtig
  - Gleichverteilung
  - Normalverteilung (Gaußverteilung)

- Zufallszahlengeneratoren erzeugen praktisch immer **gleichverteilte** Zahlen
  - daraus lassen sich normalverteilte Zahlen berechnen

Test auf Zufälligkeit

Technische Hochschule **Rosenheim**

- sind die erzeugten Zahlen gut?
  - entsprechen sie der gewünschten Verteilung?

- dazu: statistische Tests, z.B.
  - χ^2-Test (Pearson 1900)
  - Kolmogorov-Smirnov-Test (1933/39)

Erzeugung gleichverteilter Zufallszahlen

- weit verbreitet: **lineares Modulo-Kongruenzverfahren**
  - Lehmer 1949
- berechne (ganzzahlige) Zufallszahlen rekursiv aus
  $x_{n+1} = (a\, x_n + c) \bmod m$
- es ist
  - m: Modulus $0 < m$
  - a: Multiplikator $0 \leq a < m$
  - c: Inkrement $0 \leq c < m$
  - x_0 Startwert $0 \leq x_0 < m$
- erzeugt Zahlen im Intervall [0; m – 1]

Parameterwahl

- wie sind m, a, c für gute Zufallszahlen zu wählen?
- $m = 2, c = 0, a = 2, x_0 = 0$: $x_{n+1} = 2x_n \bmod 2$
  - 0, 0, 0, 0, 0, … → nicht sehr zufällig
- $m = 2, c = 0, a = 2, x_0 = 1$: $x_{n+1} = 2x_n \bmod 2$
  - 1, 0, 0, 0, 0, … → nicht sehr zufällig
- $m = 2, c = 1, a = 2, x_0 = 0$: $x_{n+1} = (2x_n + 1) \bmod 2$
  - 0, 1, 1, 1, 1, … → nicht sehr zufällig
- $m = 2, c = 1, a = 1, x_0 = 0$: $x_{n+1} = (x_n + 1) \bmod 2$
  - 0, 1, 0, 1, 0, 1, 0, 1, 0, … → nicht sehr zufällig

Parameterwahl

- m = beliebig, $c = 1$, $a = 1$, $x_0 = 0$: $x_{n+1} = (x_n + 1) \bmod m$
  - 0, 1, 2, 3, …, m – 1, 0, 1, 2, 3, … → nicht sehr zufällig
- $m = 10$, $c = 7$, $a = 7$, $x_0 = 7$: $x_{n+1} = (7x_n + 7) \bmod 10$
  - 7, 6, 9, 0, 7, 6, 9, 0, … → nicht sehr zufällig
- Fazit: Wahl der Parameter ist extrem wichtig

Parameterwahl

- $x_{n+1} = (a\, x_n + c) \bmod m$
- $a \geq 2$ $a = 0$ und $a = 1 \rightarrow$ erzeugt keine Zufallsfolge
- $c = 0$
  - schnellere Berechnung
  - kürzere Periodenlänge
- Man erhält **maximale Periodenlänge** m genau dann wenn
  - c und m keine gemeinsamen Primfaktoren haben
  - a – 1 ein Vielfaches von p ist, für jeden Primfaktor p von m
  - a – 1 ein Vielfaches von 4 ist, wenn m ein Vielfaches von 4 ist

Parameterwahl

- m: Verwendung der Wortlänge des Rechners, z.B.: 2^{32}
- Beispiele
  - stdlib im gcc
    - $m = 2^{32}$
    - $a = 1103515245$
    - $c = 12345$
  - Numerical Recipes
    - $m = 2^{32}$
    - $a = 1664525$
    - $c = 1013904223$
  - Java Random Klasse
    - $m = 2^{48}$
    - $a = 25214903917$
    - $c = 11$
- wobei nicht immer alle Bit des Ergebnisses verwendet werden
  - höherwertige Bit produzieren längere Perioden

Umrechnungen

Technische Hochschule **Rosenheim**

- gegeben: erzeugte Zufallszahl r im Intervall [0; m – 1]
- Umrechnung auf ein anderes ganzzahliges Intervall [A, B]:
  $x = A + (r \bmod (B - A + 1))$
- Umrechnung auf ein anderes reellwertiges Intervall [A, B]:
  $x = A + r\,(B - A) / (m - 1)$
- Umrechnung auf **Normalverteilung**
  - Standardnormalverteilung $\mu = 0$, $\sigma = 1$
    - Polarmethode nach Box, Muller, Marsaglia, 1958/1962
  - beliebige Normalverteilung
    - sei x normalverteilt mit $\mu = 0$, $\sigma = 1$
    - dann ist $ax + b$ normalverteilt mit $\mu = b$, $\sigma = a$

Polarmethode

- generiere zwei im Intervall [-1; +1] gleichverteilte Zufallszahlen v_1 und v_2
- Berechne $S = v_1^2 + v_2^2$
  - wiederhole diese Schritte so lange, bis S < 1
  - dies ist im Mittel 1,27 mal nötig, mit Standardabweichung 0,587
- es ergeben sich zwei standardnormalverteilte Zufallszahlen x_1 und x_2 aus

$$x_1 = v_1\sqrt{\frac{-2\ln S}{S}} \qquad x_2 = v_2\sqrt{\frac{-2\ln S}{S}}$$

Numerische Integration

- Beispiel für Monte-Carlo Algorithmus
- Berechne Approximation des Werts von $F = \int_a^b f(x)dx$
- Idee:
  - berechne umschließendes Rechteck (Fläche R) von $f(x)$ – gegeben durch Maxima/Minima im Intervall [a; b]
  - generiere N Paare von Zufallszahlen, die Koordinaten innerhalb des Rechtecks definieren
  - Zähle, wie viele Punkte N_f unterhalb der Funktion $f(x)$ liegen
  - Ein Näherungswert von F ergibt sich aus

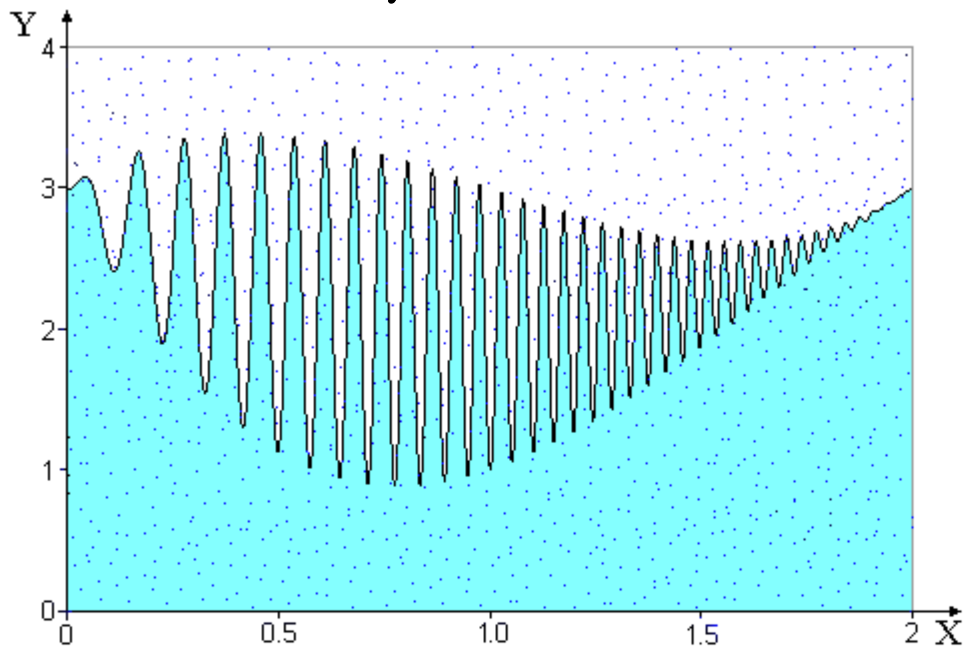
$$F = R\frac{N_\mathrm{f}}{N}$$

- das funktioniert genauso für mehrdimensionale Funktionen

Numerische Integration – Beispiel

Technische Hochschule Rosenheim

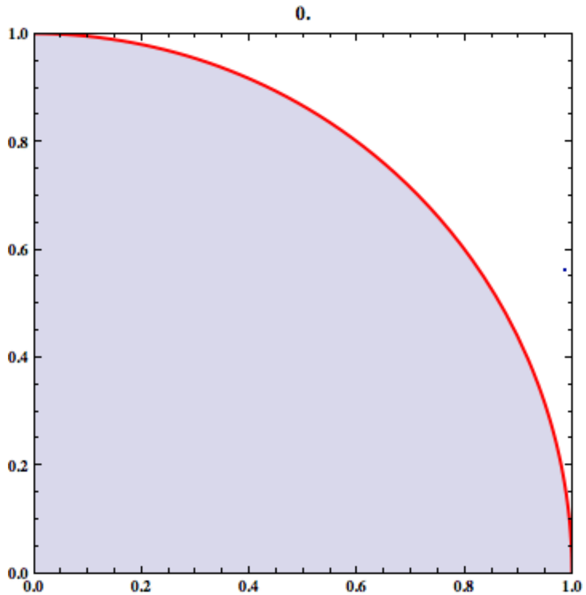
- Berechne das Integral $F = \int_0^2 \left(2 + (x-1)^2 + \sin\left[40 \cdot (x + x^2)\right] \cdot x \cdot (x-2)^2\right) dx$



- Monte-Carlo mit 10000 zufälligen Punkte liefert F = 4,671
- exakter Wert (auf drei Nachkommastellen): F = 4,667

Approximation von π

- Kreisfläche: $A = r^2\pi \quad \rightarrow \quad \pi = A / r^2$
- Verwende Viertelkreis als Funktion
- Teste, ob für zufällige Punkte (x, y) gilt $\sqrt{x^2 + y^2} \leq r$
  - dann liegt der Punkt im Kreis
- Berechne damit Fläche wie vorher
- Berechne $\pi/4$ aus Gleichung für Kreisfläche



[PI]

Primzahltests

Technische Hochschule Rosenheim

- gegeben: natürliche Zahl n
- Frage: ist n eine Primzahl?

- praktische Anwendung: Public Key Kryptographie
  - z.B. RSA (1978)
  - Schlüssel: Produkt aus zwei sehr großen Primzahlen
  - 1024 – 2048 binäre Ziffern für Schlüssel (entspricht ca. 308 bzw. 616 Dezimalziffern)

- Verfahren
  - Sieb des Eratosthenes – exponentielle Laufzeit
  - AKS-Test (2002) – polynomielle Laufzeit $\rightarrow$ PRIMES $\in$ P
    - für praktische Zwecke zu langsam
  - stattdessen: probabilistische Primzahltests – polynomielle Laufzeit

Fermats kleiner Satz

- Pierre de Fermat (ca. 1607 – 1665)
- Ist p eine Primzahl p, dann gilt für jede natürliche Zahl a, die kein Vielfaches von p ist:
$$a^{p-1} \bmod p = 1$$
- Umkehrung gilt nicht
  - es gibt auch Zahlen, die die Gleichung erfüllen, obwohl sie nicht prim sind
  - z.B.: $p = 11 * 31 = 341 \rightarrow 2^{340} \bmod 341 = 1$

- Fermatscher Primzahltest
  - prüfe für viele a, ob Fermats kleiner Satz erfüllt ist
    - wenn es ein a gibt, für das er fehlschlägt: p ist nicht prim
    - sonst: keine Aussage (interpretiert als: wahrscheinlich prim)
  - Problem: es gibt Zahlen p,
    - die nicht prim sind,
    - für die aber für **alle** a Fermats kleiner Satz erfüllt ist → Carmichael-Zahlen

Miller-Rabin Test

- veröffentlicht 1976
- jede ungerade Zahl ist darstellbar als $n = 1 + q2^k$
- wenn n prim, gilt nach Fermat:
  $a^{n-1} \bmod n = 1 \quad \rightarrow \quad a^{q2^k} \bmod n = 1$
- es gilt auch:
  $a^q \bmod n = 1$ oder $a^{q2^r} \bmod n = n - 1 = -1$
  für ein r mit $0 \leq r \leq k - 1$

- **Idee**: Berechne die Folge $(a^q, a^{2q}, a^{4q}, \ldots, a^{q2^{k-1}}, a^{q2^k})$
- Für eine Primzahl muss die Folge einer der folgenden Formen haben:
  - $(1, 1, 1, \ldots, 1)$
  - $(x_1, x_2, x_3, \ldots, x_m, -1, 1, 1, \ldots, 1)$ x_i beliebige Zahlen

Miller-Rabin Test – Beispiel

- Teste n = 11 mit a = 2
  - $11 = 1 + 5 * 2 \rightarrow q = 5, k = 1$
  - $2^5 \bmod 11 = -1$
  - $2^{10} \bmod 11 = 1$
  - → wahrscheinlich prim

- Teste n = 65 mit a = 2
  - $65 = 1 + 1 * 2^6 \rightarrow q = 1, k = 6$
  - $2^1 \bmod 65 = 2$
  - $2^2 \bmod 65 = 4$
  - $2^4 \bmod 65 = 16$
  - $2^8 \bmod 65 = 61$
  - $2^{16} \bmod 65 = 16$
  - $2^{32} \bmod 65 = 61$
  - $2^{64} \bmod 65 = 16$
  - → sicher nicht prim

Miller-Rabin Test – Beispiel

- Teste n = 561 mit a = 2
  - $561 = 1 + 35 * 2^4 \rightarrow q = 35, k = 4$
  - $2^{35} \bmod 561 = 263$
  - $2^{70} \bmod 561 = 166$
  - $2^{140} \bmod 561 = 67$
  - $2^{280} \bmod 561 = 1$
  - $2^{560} \bmod 561 = 1$
  - $\rightarrow$ sicher nicht prim
  - dies ist die kleinste Carmichael-Zahl

- Anmerkung: Die Berechnung vereinfacht sich sehr, wenn man folgende Beziehung nutzt:

$$(x \cdot y) \bmod n = \big((x \bmod n) \cdot (y \bmod n)\big) \bmod n$$

Miller-Rabin Test – Anmerkungen

Technische Hochschule **Rosenheim**

- der am weitesten verbreitete Primzahltest
- Fehlerwahrscheinlichkeit
  - das Ergebnis „nicht prim“ ist immer zu 100% richtig
  - das Ergebnis „prim“ ist für ein zufälliges a aus [2; n – 1] mit Wahrscheinlichkeit ¼ falsch
  - durch wiederholtes Testen mit verschiedenen a kann man diese beliebig klein machen
    - z.B.: 12x testen → Fehlerwahrscheinlichkeit $(\frac{1}{4})^{12}$ = 0,00000596%

- bei zusammengesetzten Zahlen liefert der Test keine Aussage über die Primfaktoren!
  - Primfaktorzerlegung ist wesentlich schwieriger
  - wie schwierig, ist derzeit nicht bekannt
    - wahrscheinlich nicht in P
    - aber mit sehr hoher Wahrscheinlichkeit auch nicht NP-vollständig (sonst wäre P = NP, da Faktorisierung nachweislich in NP **und** co-NP liegt)

Wie viele Primzahlen < n gibt es?

- sei $\pi(n)$ die Anzahl der Primzahlen kleiner gleich n
- Primzahlsatz: $\pi(n) \sim \frac{n}{\ln n}$
  - die tatsächliche Anzahl ist sogar etwas größer
  - Vermutung von Gauß (1792/93), Legendre (1797/98)

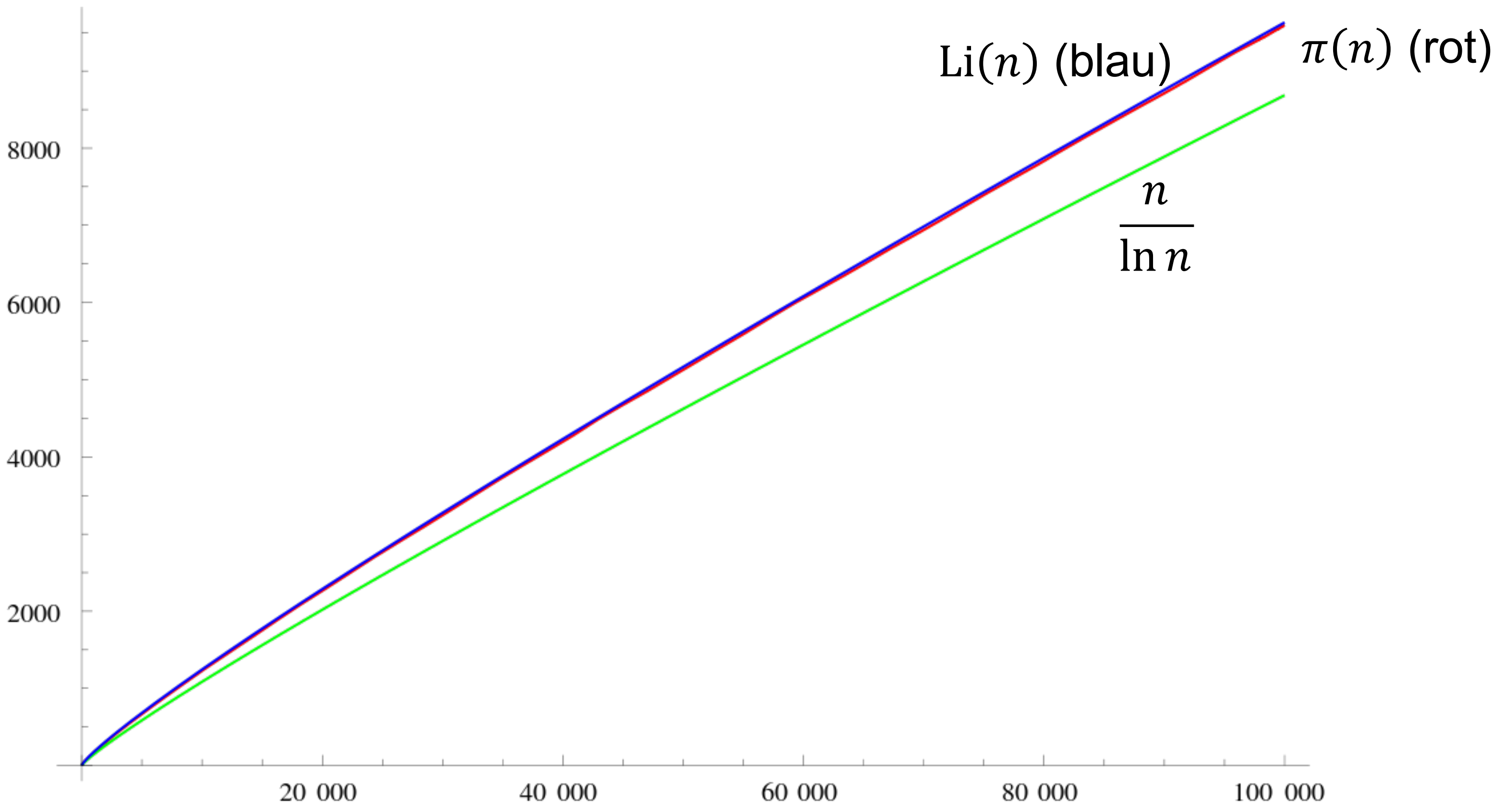
- bessere Approximation: $\pi(n) \sim \mathrm{Li}(n)$

  mit $\mathrm{Li}(n) = \int_2^n \frac{1}{\ln x} dx$
  - Vermutung von Dirichlet (1838)

- Beweise: Hadamard und Vallée-Poussin (1896)

Wie viele Primzahlen < n gibt es?

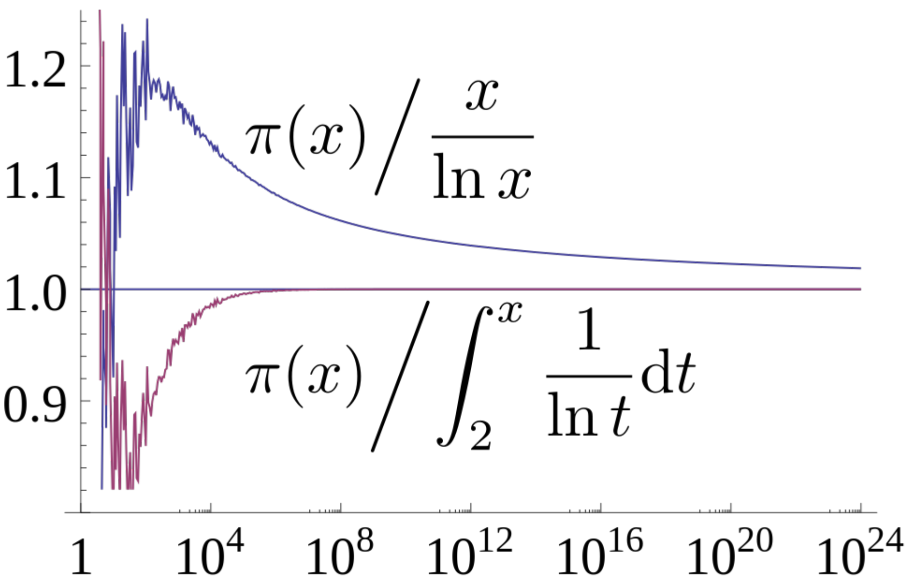
Technische Hochschule Rosenheim



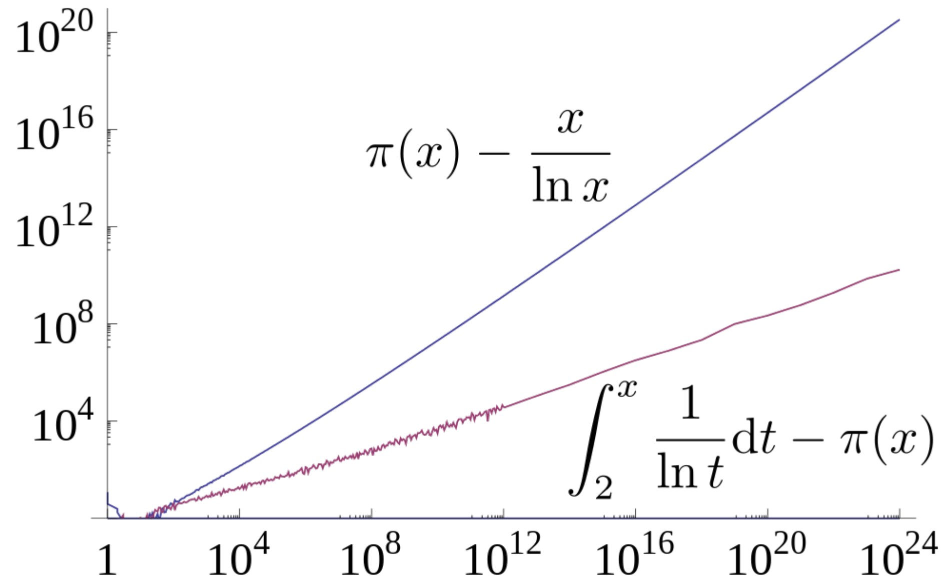
[PNT]

Wie viele Primzahlen < n gibt es?

Konvergenz



absoluter Fehler



(Public Domain)

Wahrscheinlichkeit für Primzahl

- Wahrscheinlichkeit p, dass eine zufällig gezogene Zahl x kleiner n prim ist (approximiert durch relative Häufigkeit):

$$p(x\ \mathrm{prim}) = \frac{\frac{n}{\ln n}}{n} = \frac{1}{\ln n}$$

- Wahrscheinlichkeit p, dass eine zufällig gezogene **ungerade** Zahl x kleiner n prim ist:

$$p(x\ \mathrm{prim}) = \frac{\frac{n}{\ln n}}{n/2} = \frac{2}{\ln n}$$

Wahrscheinlichkeit für Primzahl

- Wahrscheinlichkeit p, dass eine zufällig aus dem Intervall $[10^a;\ 10^{a+1}]$ gezogene **ungerade** Zahl x kleiner n prim ist:

$$p(x\ \mathrm{prim}) = \frac{\dfrac{10^{a+1}}{\ln 10^{a+1}} - \dfrac{10^a}{\ln 10^a}}{{}^1/_2\,(10^{a+1} - 10^a)}$$

- ergibt in erster Näherung:

$$p(x\ \mathrm{prim}) = \frac{\dfrac{10^{a+1}}{\ln 10^{a+1}}}{{}^1/_2\,10^{a+1}} = \frac{2}{\ln 10^{a+1}} = \frac{2}{(\mathrm{a}+1)\ln 10}$$

…also das gleiche wie auf der Folie vorher

Beispiel

Wahrscheinlichkeit p, dass eine zufällig aus dem Intervall $[10^{300};\ 10^{301}]$ gezogene **ungerade** Zahl x kleiner n prim ist:

$p(x \text{ prim}) = \frac{2}{301 \ln 10} \approx 0{,}00288567$ ca. 0,29%

exakter Wert ohne Näherung: $0{,}00288461$ ca. 0,29%

bessere Approximation durch Li-Integral:

$$p(x \text{ prim}) = \frac{\mathrm{Li}(10^{a+1}) - \mathrm{Li}(10^{a})}{{}^{1}\!/_{2}\,(10^{a+1} - 10^{a})}$$

ergibt für das Beispiel: $0{,}00321095$ ca. 0,32%

Wahrscheinlichkeit, dass von 100 gezogenen Zahlen mindestens eine prim ist:

$$1 - (1 - 0{,}0032)^{100} \approx 27{,}4\%$$

Zusammenfassung

Technische Hochschule **Rosenheim**

- probabilistische Algorithmen
  - liefern Lösungen, die mit bestimmter Wahrscheinlichkeit richtig sind (z.B. Primzahltest)
  - oder Approximationen (z.B. numerische Integration)
- es gibt viele weitere Anwendungen der Monte-Carlo Methode
  - physikalische Simulationen
  - Mikroelektronik
  - Finanzwesen
  - …
- in allen Fällen werden gute Pseudo-Zufallszahlen benötigt

Quellen

Die Folien entstanden auf Basis folgender Literatur

- H. Ernst, J. Schmidt und G. Beneken: Grundkurs Informatik. Springer Vieweg, 6. Aufl., 2016.
- D.E. Knuth: *The Art of Computer Programming*. Vol. 2, *Seminumerical Algorithms*. 3. Auflage, Addison-Wesley, 1998.
- H. Scheid: Zahlentheorie. 2. Auflage, BI Wissenschaftsverlag, 1994.

Bilder

[PI] Wikimedia.org, Autor: Caitlin Jo
http://commons.wikimedia.org/wiki/File:Pi_30K.gif
Lizenz: [1]

[PNT] Wikimedia.org, Autor: Noel Bush
https://commons.wikimedia.org/wiki/File:PrimeNumberTheorem.svg
Lizenz: [1]

[1] Attribution-ShareAlike 3.0 Unported
http://creativecommons.org/licenses/by-sa/3.0/deed.en