Abzählen von Primzahlen mit DERIVE

Johann Wiesenbauer, Wien

Kurzfassung: Ziel der Arbeit ist die Herleitung von Formeln für die Funktion $\pi(x)$, welche für eine positive reelle Zahl x die Anzahl der Primzahlen $\leq x$ angibt, und ihre Implementierung in dem Computeralgebrasystem DERIVE. Damit zusammenhängend wird auch auch auf probabilistische Primzahltests eingegangen.

1. Einleitung

Primzahlen und ihre faszinierenden Eigenschaften haben zu allen Zeiten die Mathematiker in ihren Bann gezogen und viele berühmte Namen der Mathematikgeschichte wie Fermat, Euler, Lagrange, Legendre, Gauß, Dirichlet usw. ließen sich hier als Beispiel anführen. So wird etwa von Gauß berichtet, daß er schon als Fünfzehnjähriger beim Studium einer Primzahltabelle auf empirische Weise auf den sog. Primzahlsatz gestoßen war, von dem noch die Rede sein wird. Auch später noch hat er, wie er in einem Brief an Enke schreibt (s.[3]), "sehr oft einzelne unbeschäftigte Viertelstunden darauf verwandt, um bald hie bald dort eine Chiliade [d.i. ein Intervall von 1000 Zahlen] abzuzählen".

Was ist es nun, was den besonderen Reiz der Beschäftigung mit Primzahlen ausmacht? Eine mögliche Antwort darauf hat vielleicht Don Zagier in seiner Antrittsvorlesung (s. [3]) gegeben, indem er sagte: "Es gibt zwei Tatsachen über die Verteilung der Primzahlen, von denen ich hoffe, Sie dermaßen zu überzeugen, daß sie immer in Ihrem Herzen eingraviert sind. Die eine ist, daß Primzahlen, trotz ihrer einfachen Definition und Rolle als Bausteine der natürlichen Zahlen, zu den willkürlichsten, widerspenstigsten Objekten gehören, die der Mathematiker überhaupt studiert. Sie wachsen wie Unkraut unter den natürlichen Zahlen, scheinbar keinem anderen Gesetz als dem Zufall unterworfen, und kein Mensch kann voraussagen, wo wieder eine sprießen wird, noch einer Zahl ansehen, ob sie prim ist oder nicht. Die andere Tatsache ist viel verblüffender, denn sie besagt just das Gegenteil - daß die Primzahlen die ungeheuerlichste Regelmäßigkeit aufzeigen, daß sie durchaus Gesetzen unterworfen sind und diesen mit fast peinlicher Genauigkeit gehorchen."

Beispiele für die Gültigkeit obiger Aussage werden wir im letzten Abschnitt kennenlernen, wenn wir gewissermaßen auf den Spuren von Gauß Primzahlen abzählen und Aussagen über deren Verteilung aufstellen. Anders als Gauß können wir dabei aber auf Hilfsmittel zurückgreifen - von denen dieser wahrscheinlich nicht einmal zu träumen gewagt hätte -, wie etwa auf das an Österreichs Mittelschulen weitverbreitetete Computeralgebrasystem DERIVE. (Für die in dieser Arbeit angestellten Berechnungen wurde dabei speziell die Version 4.06 von DERIVE for Windows verwendet und die angeführten Rechenzeiten beziehen sich auf einen Pentium 166 PC mit 48 MB.)

Um trockene komplexitätstheoretische Betrachtungen zu vermeiden, habe ich zu dem meisten angeführten Algorithmen DERIVE-Routinen geschrieben, welche zum einfachen Ausprobieren verleiten sollen. "Learning by doing" ist sozusagen der leitende Grundgedanke, der an unseren Schulen in der Regel viel zu kurz kommt. Daneben war es mir auch ein Anliegen, dem alten Vorurteil, DERIVE eigne sich sich nicht für das Programmieren komplizierter Aufgabenstellungen, entgegenzuwirken. Tatsächlich wird der vielleicht etwas höhere Aufwand beim Programmieren durch die im Vergleich zu anderen Computeralgebrasystem kürzeren Rechenzeiten in der Regel mehr als wettgemacht.

2. Was ist eine Primzahl?

Man könnte diese Frage auch so stellen: Welche charakterisierende Eigenschaft von Primzahlen erweist sich in Hinblick auf eine möglichst schnelle Überprüfung am geeignetesten? Vielöleicht wird man dabei zuerst an die Standarddefinition einer Primzahl denken, wonach eine zusammengesetzte Zahl $p>1$ genau dann Primzahl ist, wenn ihre einzigen positiven Teiler 1 und p selbst sind. Tatsächlich wäre aber eine direkte Verwendung dieser Definition, d.h. ein einfaches Durchprobieren, ob eine gegebene natürliche Zahl n außer 1 und n noch weitere positive Teiler hat, i.allg. viel zu aufwendig.

Es ist daher ganz lehrreich und für das folgende von großer Bedeutung wie Computeralgebrasysteme dieses Problem lösen. Im Fall von DERIVE wird die gegebene Zahl n zunächst daraufhin überprüft, ob sie einen Primfaktor $\leq \min(2^{10}, \sqrt{n})$ besitzt. Trifft dies zu oder ist n=1, so ist n sicher keine Primzahl und die Überprüfung daher abgeschlossen. Im gegenteiligen Falle allerdings ist n nur dann sicher Primzahl, wenn es die Bedingung $n \leq 2^{20}$ (=1048576) erfüllt, ansonsten muß die Überprüfung fortgesetzt werden. Im Falle von DERIVE werden dazu standardmäßig 6 sog. Rabin-Miller-Tests durchgeführt, auf welche nun etwas näher eingegangen werden soll.

Die Grundlage für den Rabin-Miller-Test bildet der sog. "Kleine Fermatsche Satz", welcher aussagt, daß für eine beliebige Primzahl p und für eine beliebige ganze Zahl a gilt $a^p \equiv a \bmod p$. (Für einen "schulgerechten" Beweis dieser Aussage s. etwa [2].) Im Fall, daß gilt (a,p)=1, insbesondere also für $0 < a < p$, ist dies offenbar äquivalent zu $a^{p-1} \equiv 1 \bmod p$.

Ist daher n die zu testende Zahl und können wir eine ganze Zahl a mit $0 < a < n$ finden , sodaß $a^{n-1} \equiv 1 \bmod n$ nicht gilt, so ist n sicher zusammengesetzt. Gilt auch die Umkehrung? Daß dies leider nicht der Fall ist, läßt sich leicht mit Hilfe von DERIVE zeigen. Wir geben dazu die nachfolgende Routine PSP(s,a) an, welche alle Elemente n einer Menge s daraufhin überprüft, ob sie für a die Bedingung $a^{n-1} \equiv 1 \bmod n$ erfüllen, ohne prim zu sein.

```
PSP(s,a):=SELECT(NOT(PRIME(n_)) AND MOD(a^(n_-1),n_)=1,n_,s)
```

Ihre Bezeichnung kommt daher, daß man solche Elemente auch Pseudoprimzahlen zur Basis a nennt, i.Z. PSP(a). Insbesondere ergibt sich nun aus

PSP([1,3,...,10000],2)= [341,561,645,1105,1387,1729,1905,2047,2465,
2701,2821,3277,4033,4369,4371,4681,5461,6601,7957,8321,8481,8911]

nach nur 5.1 s Rechenzeit, daß es immerhin 22 Pseudoprimzahlen zur Basis 2 unter 10000 gibt. (Leibniz, der an ihre Nichtexistenz glaubte, wäre darüber sicher sehr erstaunt gewesen!)

Tatsächlich kann man aber nun einen großen Teil der obigen PSP(2) noch durch die einfache Beobachtung ausschließen, daß die Kongruenz $x^2 \equiv 1 \bmod p$ für eine Primzahl p nur die Lösungen $\pm 1 \bmod p$ besitzt. (Ist u nämlich eine Lösung der Kongruenz, so folgt aus $p \mid (u+1)(u-1)$ wegen der Primzahleigenschaft von p sofort, daß $p \mid (u+1)$ oder $p \mid (u-1)$.)

Ausgehend von der Gleichung $a^m \equiv 1 \bmod n$, wo m gerade ist (zu Beginn setzt man $m := n - 1$, wobei n als ungerade vorausgesetzt wird), ist nun sicher $a^{m/2} \bmod n$ eine "Wurzel" aus 1 mod n. Um n auf Primalität zu testen, hat man dann folgende Fälle zu unterscheiden:

1. $a^{m/2} \bmod n$ ergibt einen Wert, der verschieden von ± 1 ist, in welchen Falle n den Test nicht bestanden hat.
2. $a^{m/2} \bmod n$ ergibt -1, oder, falls $m/2$ ungerade ist, +1. In diesem Fall wollen wir den Test als bestanden ansehen.
3. $a^{m/2} \bmod n$ ergibt den Wert +1, jedoch $m/2$ ist gerade. In diesem Fall wird der Test mit $m/2$ statt m wiederholt.

Betrachten wir etwa die kleinste Pseudoprimzahl 341 zur Basis 2. Wie sich aus

MODS(2^170,341)=1,

MODS(2^85,341)]=32

unmittelbar ergibt, werden wir zunächst auf den Fall 3 und dann weiter auf den Fall 1 geführt, was beweist, daß 341 zusammengesetzt sein muß. Tatsächlich gilt

FACTOR(341) = 11·31

In der Praxis erweist sich als günstig die einzelnen Rechenschritte in der umgekehrten Reihenfolge vorzunehmen. Man beginnt dazu mit der Berechnung von $a^s \bmod n$, wobei die ungerade Zahl s der eindeutigen Darstellung $n = s2^t + 1$ (t>0) entnommen ist. Der Test gilt dann als bestanden, wenn entweder $a^s \bmod n = \pm 1$ ist oder man durch höchstens (t-1)-maliges Quadrieren dieses Wertes einmal auf -1 mod n stößt. Ein DERIVE-Programm könnte dann etwa so aussehen (man beachte, daß n dazu unbedingt ungerade sein muß!):

```
RABIN_MILLER(n,a):=IF((ITERATE(IF(k_=2 OR a_=-1,[a_,k_],
    [MODS(a_^2,n),k_/2]),[a_,k_],ITERATE([-ABS(MODS(a^o_,n)),
    (n-1)/o_],o_,ITERATE(IF(MOD(n_,2)=1,n_,n_/2),n_,n-1),1))) SUB 1=-1,
    true,false)

SPSP(s,a):=SELECT(NOT(PRIME(n_)) ∧ RABIN_MILLER(n_,a),n_,s)
```

Indem man nun für alle ungeraden zusammengesetzten Zahlen bis 10000 diejenigen berechnet, welchen den Rabin-Miller-Test für die Basis zu 2 bestehen

- diese werden auch starke Pseudoprimzahlen für die Basis 2 genannt, i.Z. SPSP(2) - erhält man nach nur 22.2 s folgendes Ergebnis:

SPSP([1,3,...,10000],2)=[2047,3277,4033,4681,8321]

Von den 22 Pseudoprimzahlen zur Basis 2 haben sich also immerhin noch 5 auch gegenüber dem Rabin-Miller-Test zur Basis 2 als "resistent" erwiesen! Es hindert uns aber niemand daran, den Rabin-Miller-Test für eine weitere Basis, wie z.B. 3 nocheinmal durchzuführen. Als Ergebnis erhalten wir, wie zu erwarten

SPSP([2047, 3277, 4033, 4681, 8321], 3) = [ ]

d.h. es gibt keine zusammengesetzten ungeraden Zahlen unter 10000, welche die Rabin-Miller-Tests für die Basen 2 und 3 bestehen. Tatsächlich sollte, wie Rabin gezeigt hat, bei k Rabin-Miller-Tests für zufällig ausgewählten Basen a mit $1<a<n$ die Irrtumswahrscheinlichkeit für die Aussage "n ist prim" auf weniger als $1/4^k$ sinken. Die nachstehende Tabelle zeigt jedoch, daß die tatsächliche Irrtumswahrscheinlichkeit i.allg. viel kleiner sein dürfte.

Basen a	Kleinste SPSP(a) für alle diese a
2	2047=23·89
2,3	1373653=829·1657
2,3,5	25326001=2251·11251
2,3,5,7	3215031751=151·751·28351
2,3,5,7,11	2152302898747=6763·10627·29947
2,3,5,7,11,13	3474749660383=1303·16927·157543
2,3,5,7,11,13,17	341550071728321=10670053·32010157
2,3,5,7,11,13,17,19	341550071728321=10670053·32010157

Wie ferner Miller gezeigt hat, folgt aus der Annahme der Gültigkeit der sog. "Verallgemeinerten Riemannnschen Vermutung", daß der Rabin-Miller-Test sogar zu einem "sicheren" Test wird, falls man ihn für alle Basen a mit $1< a< < 2(\ln n)^2$ durchführt. Dem steht allerdings die Aussage gegenüber, daß es zu einer beliebigen Auswahl von Basen stets eine Zahl n gibt, welche SPSP(a) für alle Basen der betrachteten Menge ist.

Wie aus den bisherigen Betrachtungen folgt, ist der in DERIVE eingebaute Primalitätstest nur bis 2^{20} absolut sicher, denn für diesen Zahlenbereich kommt man noch ohne Rabin-Miller-Tests aus. Leider ist mir nicht bekannt, in welcher Weise die Auswahl der bis zu 6 Basen tatsächlich erfolgt. Sicher ist nur, daß sie deterministisch ist, da die Ergebnisse reproduzierbar sind, und daß auch nicht einfach die Primzahlen in ihrer natürlichen Reihenfolge genommen werden, so wie in obiger Tabelle. Trotzdem ist es mir nach langen Versuchen gelungen, eine Zahl zu konstruieren, welche alle eingebauten Primalitätstests besteht, von der ich allerdings nicht beweisen konnte, daß sie die kleinste derartige Zahl ist. Man sehe sich dazu die folgenden Zeilen an:

n := 22564845703

MOD(n, 106219) = 0

PRIME(n) = true

NEXT_PRIME(n - 1) = 22564845703

FACTOR(n) = 22564845703

PRIME(n, 7) = false

NEXT_PRIME(n - 1, 7) = 22564845803

Wie man sehen kann, funktioniert auch FACTOR für dieses n nicht, da ja vor dem Faktorisieren vernünftigerweise immer ein Primalitätstest durchgeführt wird, der eben hier nur leider schief geht. Dankenswerterweise haben die DERIVE-Programmierer für die Funktionen PRIME(n) und NEXT_PRIME(n) optional als zweiten Parameter die Anzahl der Rabin-Miller-Tests vorgesehen, womit also der Benutzer selbst über das Ausmaß der Sicherheit bei Primalitätstests entscheiden kann. Leider gilt dies nicht auch für FACTOR, sodaß diese Funktion wirklich ihren Dienst versagt.

Auch wenn die Programmierer von DERIVE derzeit an einer Revision des eingebauten Primalitätstests arbeiten - was insbesondere zur Folge haben könnte, daß dann 22564845803 als zusammengesetzt erkannt wird - , so ändert dies nichts daran, daß er für große Zahlen grundsätzlich von probabilistischer Natur ist und daher u.U. auch falsche Ergebnisse liefern kann. Wir werden daher im folgenden die darauf aufbauenden Funktionen NEXT_PRIME(n) und PRIME(n) nur dann verwenden, wenn n im "sicheren" Bereich $n<2^{20}$ ist.

Was ist nun wirklich die für das Abzählen von Primzahlen geeignetste Charakterisierung? Die vielleicht etwas überraschende Antwort: Primzahlen können für unsere Zwecke am besten dadurch beschrieben werden, daß sie nach Anwendung gewisser Siebmethoden "übrigbleiben". Wie man sich dies im einzelnen vorzustellen hat, wird sich aus den folgenden Ausführungen ergeben, welche die Implementierung der Funktion $\pi(x)$, welche für eine positive reelle Zahl x die Anzahl der Primzahlen $\leq x$ angibt, zum Ziel haben.

3. Implementierung von $\pi(x)$

Eine der einfachsten Siebmethoden kennen wir schon von der Schule her unter dem Namen "Sieb des Eratosthenes". Wir wollen nun als einführendes Beispiel eine leicht modifizierte Form davon benützen, um alle Primzahlen bis 100 abzuzählen und verwenden dazu nachfolgende DERIVE-Routinen

```
STREICHE(s,a):=(ITERATE([VECTOR(IF(MOD(k_,p_)=0,0,s_ SUB k_),
  k_,1,DIMENSION(s)),NEXT_PRIME(p_)],[s_,p_],[s,2],a)) SUB 1

MATRIX(s,n):=VECTOR(VECTOR(s SUB j_,j_,i_*n+1,i_*n+n),
  i_,0,DIMENSION(s)/n-1)

SIEB(s,a,n):=MATRIX(STREICHE(s,a),n)
```

wobei die erste aus dem Vektor s (in der Regel wird dieser von der Form [1,2,..,m] sein, doch wollen wir uns hier nicht festlegen) alle Zahlen 0 setzt - also bildlich gesprochen "streicht"-, welche durch eine der ersten a Primzahlen $p_1,p_2,...,p_a$ teilbar sind, und die zweite dazu dient, den resultierenden Vektor in übersichtlicher Form als Tabelle mit n Spalten darzustellen. Durch Hintereinanderausführung dieser beiden Routinen erhalten wir dann SIEB(s,a,n). Beispielsweise erhält man so, indem man aus den Zahlen ≤ 100 alle diejenigen "aussiebt", welche durch eine der ersten 4 Primzahlen 2,3,5 oder 7 teilbar sind, das folgende Bild:

```
SIEB([1, ..., 100], 4, 20)
```

1	0	0	0	0	0	0	0	0	0	11	0	13	0	0	0	17	0	19	0
0	0	23	0	0	0	0	0	29	0	31	0	0	0	0	0	37	0	0	0
41	0	43	0	0	0	47	0	0	0	0	0	53	0	0	0	0	0	59	0
61	0	0	0	0	0	67	0	0	0	71	0	73	0	0	0	0	0	79	0
0	0	83	0	0	0	0	0	89	0	0	0	0	0	0	0	97	0	0	0

Bezeichnet nun allgemein $\Phi(x,a)$ die Anzahl der ganzen Zahlen ≤ x, welche durch keine der ersten a Primzahlen teilbar ist, so erhält man speziell in unserem Beispiel nach dem Inklusion-Exklusionsprinzip dafür den Wert

$$\begin{aligned}\Phi(100,4)= 100 &- [100/2] - [100/3] - [100/5] - [100/7] +[100/(2\cdot 3)] + \\ &+ [100/(2\cdot 5)] + [100/(2\cdot 7)] + [100/(3\cdot 5)] +[100/(3\cdot 7)] + \\ &+ [100/(5\cdot 7)] - [100/(2\cdot 3\cdot 5)] - [100/(2\cdot 3\cdot 7)] -[100/(2\cdot 5\cdot 7)] - \\ &- [100/(3\cdot 5\cdot 7)] + [100/(2\cdot 3\cdot 5\cdot 7)] = \\ =100 &- 50 - 33 - 20 - 14 +16 +10 + 7 + 6 + 4 + 2 - 3 - 2 - 1 - 0 + 0=22\end{aligned}$$

in Übereinstimmung mit dem Ergebnis, welches man aus obiger Tabelle durch direktes Abzählen gewinnt. Da aber jede zusammengesetzte Zahl ≤ 100 durch eine der Primzahlen 2,3,5,7 teilbar sein muß, erhält man, indem man die 4 "zuviel ausgesiebten" Primzahlen 2,3,5,7 wieder dazugibt und andererseits berücksicht, daß 1 per definitionem keine Primzahl ist, das korrekte Ergebnis

$$\pi(100) = \Phi(100,4) +4 - 1 = 25.$$

Dies läßt sich für eine beliebige positive reelle Zahl x offenbar sofort verallgemeinern zu

$$\pi(x) = \Phi(x,a) +a - 1 \text{ mit } a= \pi(\sqrt{x}),$$

da jede zusammengesetzte Zahl ≤ x einen Primfaktor ≤$\sqrt{x}$ haben muß. Dies ist eine bekannte Formel von Legendre, mit deren Hilfe sich $\pi(x)$ rekursiv auf $\pi(\sqrt{x})$ zurückführen läßt, vorausgesetzt man kann den Wert von $\Phi(x,\pi(\sqrt{x}))$ berechnen. Leider ist gerade die Berechnung von $\Phi(x,a)$ für großes a die Hauptschwierigkeit. Man behilft sich daher in der Praxis oft so, daß man für a kleinere Werte als $\pi(\sqrt{x})$ nimmt, und anschließend versucht, die verbleibenden zusammengesetzten Zahlen noch gesondert auszusieben.

Bezeichnet dazu $P_k(x,a)$ die Anzahl der Zahlen $\le x$, welche das Produkt von k (nicht notwendig verschiedenen) Primfaktoren $> p_a$ sind, so gilt offenbar

$P_1(x,a) = \max(\pi(x) - a, 0)$,

$P_k(x,a) = P_{k-1}(x/p_{a+1},a)+...+P_{k-1}(x/p_{a+m},a+m-1)$

wobei m die größte natürliche Zahl mit $p_{a+m}^k \le x$ ist. (Man beachte dazu, daß man $P_{k-1}(x/p_{a+j}, a+j-1)$, $1 \le j \le m$, interpretieren kann als die Anzahl aller Zahlen $\le x$, welche aus genau k Primfaktoren $\ge p_{a+j}$ bestehen.)

Hätte man z.B. in unserem zuvor betrachten Beispiel nur $a=2$ genommen, so erhält man durch Anwendung von FACTOR folgende Matrix:

```
FACTOR(SIEB([1, ..., 100], 2, 10))
```

1	0	0	0	5	0	7	0	0	0
11	0	13	0	0	0	17	0	19	0
0	0	23	0	5^2	0	0	0	29	0
31	0	0	0	5·7	0	37	0	0	0
41	0	43	0	0	0	47	0	7^2	0
0	0	53	0	5·11	0	0	0	59	0
61	0	0	0	5·13	0	67	0	0	0
71	0	73	0	0	0	7·11	0	79	0
0	0	83	0	5·17	0	0	0	89	0
7·13	0	0	0	5·19	0	97	0	0	0

Wie man sieht, ist also $P_2(100,2) \neq 0$, und man kann erhält dessen Wert nach obigem durch

$$\begin{aligned} P_2(100,2) &= P_1(100/5,2) + P_1(100/7,3) = \\ &= (\pi(20) - 2) + (\pi(100/7) - 3) = \\ &= (8 - 2) + (6 - 3) = 9. \end{aligned}$$

Tatsächlich gibt es, wie man sofort nachprüft, 9 zusammengesetzte Zahlen in obiger Tabelle, und zwar 6 mit 5 als kleinstem Primfaktor und 3 mit 7 als kleinstem Primfaktor. Insbesondere erhält man wieder

$$\begin{aligned} \pi(100) &= \Phi(100,2) + 2 - 1 - 9 = 100 - 100/2 - [100/3] + [100/6] + 2 - 1 - 9 = \\ &= 100 - 50 - 33 + 16 + 2 - 1 + 9 = 25. \end{aligned}$$

Allgemeiner ergibt sich so für $\pi(x)$ die Formel

$$\pi(x) = \Phi(x,a) + a - 1 - P_2(x,a) - ... - P_{m-1}(x,a) \text{ mit } p_{a+1}^{m-1} \le x < p_{a+1}^m .$$

Obige Bedingung für m ist dabei insbesondere dann stets erfüllt, wenn man $a = \pi(\sqrt[m]{x})$ wählt. (Für $m<3$ ist natürlich obige Summe der Terme $P_j(x,a)$ mit $1<j<m$ leer, also 0, womit man als Spezialfall die Formel von Legendre erhält.)

Wie wollen nun obige Überlegungen dazu verwenden um $\pi(x)$ in DERIVE zu implementieren. Für Hilfszwecke konstruieren wir zunächst eine allererste noch ganz primitive Version

```
PPI(x, s) := (ITERATE(IF(p_ > x, [p_, k_], [NEXT_PRIME(p_), k_ + 1]),
  [p_, k_], [s, -1])) SUB 2
```

PPI(x,s) berechnet die Anzahl der Primzahlen p mit $s<p\leq x$ und benutzt dazu die eingebaute Funktion NEXT_PRIME(n), welche in dem von uns nachfolgend betrachteten Bereich $n < 10^6$ noch absolut verläßlich ist. Wir verwenden nun PPI(x,s) dazu, um die nachfolgenden Listen zu berechnen, welche später als "Stützstellen" für eine Primzahltabelle bis 10^6 dienen sollen:

```
INSERT_ELEMENT(0, VECTOR(PPI(x_, 0), x_, SELECT(GCD(k_, 30) = 1,
  k_, 7, 1001))) = [0, 4, 5, 6, 7, 8, 9, 10, ...,  166, 166, 167, 168, 168]
```

```
ITERATES([s_ + PPI(x_ + 100, x_), x_ + 100], [s_, x_], [PPI(1000, 0), 1000],
  990)` SUB 1 = [168, 184, 196, 211, 222,..., 9556, 9564, 9574, 9584, 9592]
```

```
ITERATES([s_ + PPI(x_ + 500, x_), x_ + 500], [s_, x_], [PPI(100000, 0), 100000],
  1800)` SUB 1 = [9592, 9632, 9673, ..., 78394, 78433, 78466, 78498]
```

Damit konstruieren wir nun die nachfolgenden Funktionen, wobei die darin vorkommenden Listen bei der Eingabe mit Hilfe der Funktionstaste F3 von oben übernommen wurden:

```
PIT(x) := IF(x < 7, [0, 0, 1, 2, 2, 3, 3] SUB FLOOR(x + 1), [0, 4, 5, 6, 7, 8, 9, 10,
  ...,166, 166, 167, 168, 168] SUB (- FLOOR(x/30) + FLOOR(x/15) + FLOOR(x/10)
  + FLOOR(x/6) - FLOOR(x/5) - FLOOR(x/3) - FLOOR(x/2) + FLOOR(x)))
```

```
IP(x, p, s) := IF(p > x, s, IP(x, NEXT_PRIME(p), s + 1))
```

```
PIHT(x) := IP(x, NEXT_PRIME(FLOOR(x, 100)·100), [168, 184, 196, 211, 222,...,
  9556, 9564, 9574, 9584, 9592] SUB (FLOOR(x, 100) - 9))
```

```
PIM(x) := IP(x, NEXT_PRIME(FLOOR(x, 500)·500), [9592, 9632, 9673, 9719,...,
  78394, 78433, 78466, 78498] SUB (FLOOR(x, 500) - 199))
```

Hier werden verschiedene, auch für sich genommen recht lehrreiche Techniken verwendet um eine Primzahltabelle bis 10^6 zu konstruieren: Bis 1002 speichern wir nur für von Werte x, welche weder durch 2,3 oder 5 teilbar sind, die zugehörigen Werte für $\pi(x)$, in dem Bereich $1002 < x < 100100$ nur für Werte die durch 100 teilbar sind und schließlich für $100099<x<1000500$ nur für Werte von x, die durch 500 teilbar sind. (Für dazwischenliegende Werte von x wird in PIHT(x) bzw. PIM(x) mit Hilfe der NEXT_PRIME-Funktion von DERIVE "interpoliert".) Damit ergibt sich $\pi(x)$ für alle $x < 1000500$ zu

```
SMALLPI(x) := IF(x < 1003, PIT(x), IF(x < 100100, PIHT(x), PIM(x)))
```

Die restliche Implementierung von $\pi(x)$ sieht dann so aus:

```
PRIMEPI(x) := SMALLPI(x)
```

```
SEL(x, k, a) := DELETE_ELEMENT(ITERATES(NEXT_PRIME(p_), p_,
  ITERATE(NEXT_PRIME(q_), q_, 1, a), MAX(PRIMEPI(x^(1/k)) - a, 0)))
```

P(x, k, a) := IF(k = 1, MAX(PRIMEPI(x) - a, 0), SUM(P(x/u_, k - 1, PRIMEPI(u_) - 1), u_, SEL(x, k, a)))

NEWLIST(p, l) := APPEND(l, VECTOR(SIGN(l_)·FLOOR(ABS(l_)), l_, SELECT(ABS(l_) ≥ 1, l_, - l/p)))

Φ(x, a) := SUM((ITERATE([NEWLIST(p_, f_), NEXT_PRIME(p_)], [f_, p_], [[FLOOR(x)], 2], a)) SUB 1)

PRIMEPI(x) := IF(x > 1000499, Φ(x, 15) + 14 - SUM(P(x, i_, 15), i_, 2, 10), SMALLPI(x))

4. Approximationen für $\pi(x)$

Während zu Gauß' Zeiten die Werte für $\pi(x)$ nur etwa bis 3000000 bekannt waren - zum Vergleich: Mit Hilfe obiger Routinen braucht DERIVE für die Berechnung von PRIMEPI(3000000) = 216816 gerade mal 10 Sekunden -, kann man mit obiger Implementierung von $\pi(x)$ ohne allzu großen Aufwand in den Milliardenbereich vorstoßen. Als Beispiel wollen wir die folgende Tabelle für die Funktion $x/\pi(x)$ anlegen, welche als Maß für die durchschnittliche "Lücke" zwischen zwei Primzahlen ≤ x dienen kann. (Die Gesamtrechenzeit dafür betrug weniger als eine Stunde.)

x	$\pi(x)$	$x/\pi(x)$
10^1	4	2.5
10^2	25	4.0
10^3	168	6.0
10^4	1229	8.1
10^5	9592	10.4
10^6	78498	12.7
10^7	664579	15.0
10^8	5761455	17.4
10^9	50847534	19.7

Wie man sofort sieht, ist die Funktion $x/\pi(x)$ als Funktion des Exponenten von x annähernd linear mit der Steigung ≈ 2.3. Wegen ln 10 ≈ 2.3 führt dies auf die Abschätzung $x/\pi(x) \approx \ln x$ bzw. $\pi(x) \approx x/\ln x$. Obwohl nun die Absolutdifferenz mit wachsendem x beliebig groß werden kann, so vermutete schon der damals 15-jährige Gauß - wie bereits in der Einleitung erwähnt - aufgrund von weit weniger Daten, daß der relative Fehler bei der Ersetzung von $\pi(x)$ durch $x/\ln x$ mit wachsendem x gegen 0 geht, d.h. daß $\pi(x)$ und $x/\ln x$ "asymptotisch gleich" sind, i.Z. $\pi(x) \sim x/\ln x$. Ebendies ist der Inhalt des berühmten Primzahlsatzes, der allerdings erst 1896 von Hadamard und de la Vallée-Poussin bewiesen werden konnte.

Obige Ergebnisse kann man auch so deuten, daß die "Primzahldichte" in der Umgebung von x etwa $1/\ln x$ beträgt, wonach dann der sog. Integrallogarithmus

$$\mathrm{li}(x)=\int_0^{\infty}\frac{dt}{\ln t}$$

eigentlich eine noch bessere Näherung für $\pi(x)$ darstellen sollte, worauf ebenfalls Gauß schon hingewiesen hat. (Um die Singularität bei t=1 zu vermeiden, wird als untere Grenze des Integrals auch oft 2 genommen, doch ist die Differenz, welche $\approx$ 1.04 beträgt, für unsere Zwecke ohne Belang.) Natürlich gilt auch für den Integrallogarithmus $\pi(x) \sim \mathrm{li}(x)$, es ist dies eigentlich nur eine andere Form des Primzahlsatzes.

Unter der Voraussetzung der Richtigkeit der sog. Riemannschen Vermutung, in welcher behauptet wird daß die nichtreellen Nullstellen der Riemannschen ζ-Funktion - sie ist in DERIVE unter der Bezeichnung ZETA(z) fix implementiert - alle auf der Geraden Re(z)=1/2 liegen, könnte man sogar beweisen, daß gilt

$$|\pi(x) - \mathrm{li}(x)| \le C\sqrt{x}\ln x$$

für eine positive reelle Konstante C, wofür man auch $\pi(x) = \mathrm{li}(x) + \mathrm{O}(\sqrt{x}\ln x)$ schreibt. Dies würde bedeuten, das beim Vergleich von li(x) mit $\pi(x)$ etwa die Hälfte der führenden Stellen richtig ist, was in der nachstehenden Tabelle, welche ebenfalls wieder mit DERIVE erstellt wurde, recht gut zutrifft:

x	li(x) - $\pi(x)$	R(x) - $\pi(x)$
10^2	5	1
10^3	10	0
10^4	17	-2
10^5	38	-5
10^6	130	29
10^7	339	88
10^8	754	97
10^9	1701	-79

Die hierbei gleich mitangeführte Funktion R(x) wird nach ihrem Entdecker Riemannsche Funktion genannt und man kann sie vielleicht am einfachsten durch die nachstehende schnell konvergente Reihe definieren (s. [1])

$$\mathrm{R}(x) = 1 + \sum_{k=1}^{\infty}\frac{1}{k\zeta(k+1)}\frac{(\ln x)^k}{k!}$$

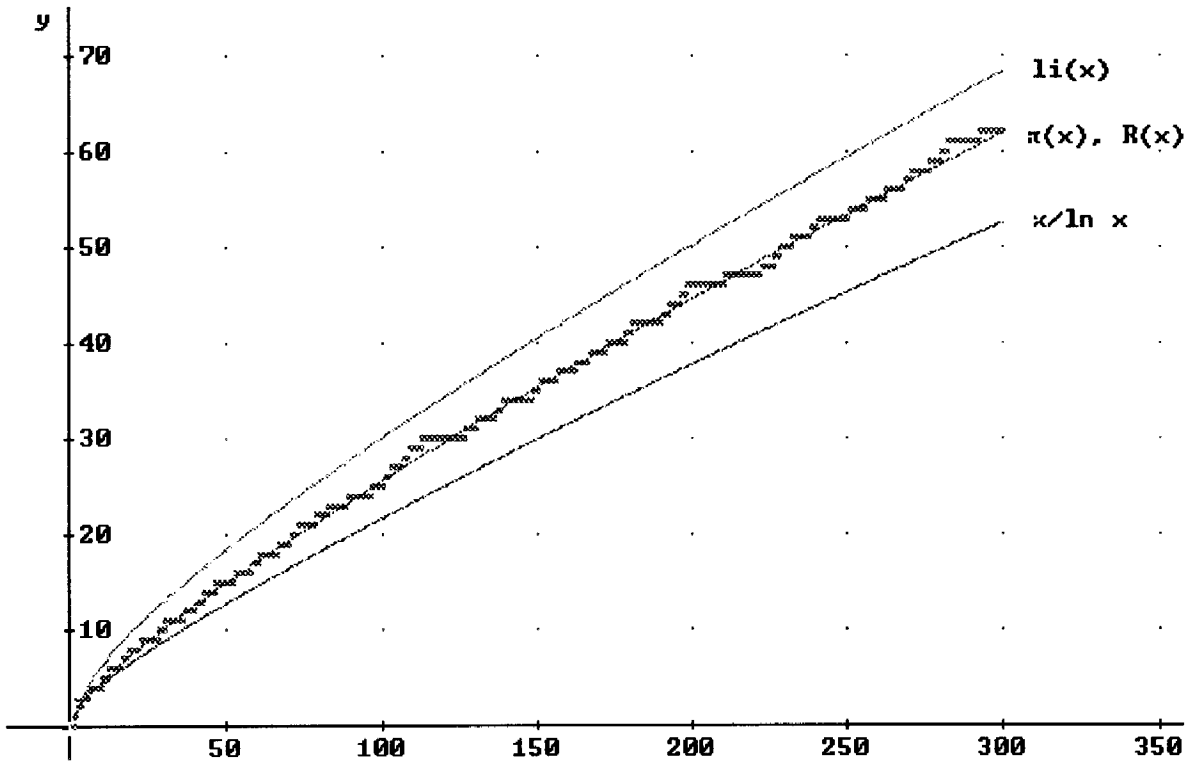
Für eine Implementierung von R(x) genügt es für den oben betrachteten Bereich die ersten 100 Glieder dieser Reihe zu nehmen. Es empfielt sich dazu, den Vektor

VECTOR(1/(k·ZETA(k + 1)·k!), k, 100)

im voraus approximativ zu berechen. Weist man danach(!) das Ergebnis v zu, so kann man nun R(x) einfach definieren durch

R(x) := APPROX(1 + VECTOR(LN(x)^k_, k_, 1, 100)·v)

Wie man aus obiger Tabelle entnehmen kann, ist R(x) eine unglaublich gute Approximation von $\pi(x)$ und unterstreicht damit sehr eindrucksvoll das in der Einleitung Gesagte. Tatsächlich ließe sich selbst R(x) noch unter Einbeziehung der Nullstellen der Riemannschen ζ-Funktion beliebig weiter verbessern, doch wollen wir es mit dem nachfolgenden Schaubild, in dem alle drei Approximationen für $\pi(x)$ noch einmal zusammen zu sehen sind und man den Treppencharakter von $\pi(x)$ noch gut erkennt, bewenden lassen.



Literatur

[1] Riesel H., "Prime numbers and computer methods for factorization", Birkhäuser Verlag, Boston-Basel-Stuutart, 1985

[2] Wiesenbauer J., "Number Theory with DERIVE - Some Suggestions for Classroom Teaching", in: DERIVE in Education (ed. Heugl-Kutzler), Chartwell-Bratt, 1994

[3] Zagier D., "Die ersten 50 Millionen Primzahlen", Mathem. Miniaturen 1, Birkhäuser Verlag, 39-73 (1981)

Anschrift des Verfassers:

Johann Wiesenbauer,
Inst. f. Algebra und Diskrete Mathematik
der Techn. Univ.Wien
Wiedner Hauptstr. 8-10,
A 1040 Wien