

Miller-Rabin

Primzahltest

Steffen Greber und Raphael Stein
22.Oktober 2008

1. Einleitung

Diese Arbeit befasst sich mit dem Miller-Rabin-Test. Er gilt als einer der besten Wahrscheinlichkeitstests für Primzahlen. Benannt ist der Test nach Gary Miller und Michael O. Rabin und wurde 1975 veröffentlicht.

2. Argumentation

Zur Veranschaulichung folgt eine Matrix, die in der y-Achse die Zahlen von 1 bis zur getesteten Primzahl p minus 1 hat. Die x-Achse entspricht den Exponenten von 1 bis zur getesteten Primzahl p minus 1. Berechnet wird nun die Zahl der y-Achse hoch den Exponenten der x-Achse modulus der zu testenden Primzahl p .

$$M[i][j] = i^j \bmod p$$

Als Beispiel nehmen wir die zur Primzahl 13 zugehörige Matrix.

```
a[i, j] := power_mod(i, j, 13);  
m: genmatrix(a, 12, 12);
```

[1	1	1	1	1	1	1	1	1	1	1	1]
[2	4	8	3	6	12	11	9	5	10	7	1]
[3	9	1	3	9	1	3	9	1	3	9	1]
[4	3	12	9	10	1	4	3	12	9	10	1]
[5	12	8	1	5	12	8	1	5	12	8	1]
[6	10	8	9	2	12	7	3	5	4	11	1]
[7	10	5	9	11	12	6	3	8	4	2	1]
[8	12	5	1	8	12	5	1	8	12	5	1]
[9	3	1	9	3	1	9	3	1	9	3	1]
[10	9	12	3	4	1	10	9	12	3	4	1]
[11	4	5	3	7	12	2	9	8	10	6	1]
[12	1	12	1	12	1	12	1	12	1	12	1]

Auffällig an diesen Matrizen ist, dass die letzte Spalte komplett aus 1 besteht. Diese Form haben nur Primzahlen oder Carmichaelzahlen (Bsp: 561, 1105, 1729). Primzahlen haben zudem noch in der mittleren Spalte eine 1 oder „-1“.

Die 12 entspricht hier einer -1 , denn:

$$\begin{aligned} \text{mod}(-1, 13) &; & 12 \\ \text{mod}(12, 13) &; & 12 \end{aligned}$$

Dieses Muster macht sich der Miller-Rabin-Test zu nutzen.

Wenn man die mittlere Spalte betrachtet ($j=6$), kann man sehen, dass wenn man diese Zahlen quadriert und wieder Moduls rechnet die Zahlen in der letzten Spalte berechnet werden können. Da für die mittlere Spalte gilt i^6 und diese zum Quadrat wäre $(i^6)^2$ und das entspricht der Zahl in der letzten Spalte.

Miller-Rabin macht sich das Gegenteil zum Nutzen.

Man zieht so lange die Wurzel aus i bis der Exponent ungrade wird. Wie oft man diese Wurzel ziehen kann, wird wie folgt berechnet:

$$p-1 = q \cdot 2^k$$

Man versucht die Primzahl p minus 1 durch eine Zweierpotenz und einen Primfaktor zu zerlegen. Am Zahlenbeispiel 13 bedeutet das:

$$13-1 = 3 \cdot 2^2$$

Um die Variablen q und k zu berechnen wird erst eine Variable q definiert., die den Wert $p-1$ besitzt

$$q := p-1$$

und die Variable k , die am Anfang den Wert 0 hat.

$$k := 0$$

Solange q eine grade Zahl ist, kann man eine 2 herausfaktorisieren die dann beim Exponenten k angerechnet wird.

$$\text{while evenp}(q) \text{ do } (k:k+1, q:q/2)$$

Diese Schleife teilt die Zahl q so lange durch 2 bis sie ungrade ist. $\text{evenp}(x)$ testet ob die Zahl grade ist.

Bei unsern Beispiel 13 wird die Zahl $q=3$ und $k=2$ errechnet.

Nun wird eine Zufällige Zahl ermittelt die größer als 1 und kleiner als $p-1$ ist.

$$x := \text{random}(p-2) + 2$$

Man betrachtet nun die Spalte q . Falls dort eine 1 oder -1 (Hier in dem Beispiel 12) steht, muss zwangsläufig in der Spalte $q \cdot 2^k$ auch eine 1 stehen.

if (power_mod(x,q,p)=1 or power_mod(x,q,p)=-1)

Falls das zutrifft, ist die gesuchte Zahl eine Primzahl, denn wenn man 1 oder -1 quadriert bekommt man in der letzten Spalte ebenfalls eine 1.

[1	1	1	1	1	1	1	1	1	1	1]
[2	4	8	3	6	12	11	9	5	10	7	1
[3	9	1	3	9	1	3	9	1	3	9	1
[4	3	12	9	10	1	4	3	12	9	10	1
[5	12	8	1	5	12	8	1	5	12	8	1
[6	10	8	9	2	12	7	3	5	4	11	1
[7	10	5	9	11	12	6	3	8	4	2	1
[8	12	5	1	8	12	5	1	8	12	5	1
[9	3	1	9	3	1	9	3	1	9	3	1
[10	9	12	3	4	1	10	9	12	3	4	1
[11	4	5	3	7	12	2	9	8	10	6	1
[12	1	12	1	12	1	12	1	12	1	12	1

[1	1	1	1	1	1	1	1	1	1	1	1	1	1]
[2	4	8	3	6	12	11	9	5	10	7	1	15	13	9
[3	9	1	3	9	1	3	9	1	3	9	1	12	2	6
[4	3	12	9	10	1	4	3	12	9	10	1	4	16	13
[5	12	8	1	5	12	8	1	5	12	8	1	3	15	7
[6	10	8	9	2	12	7	3	5	4	11	1	10	9	3
[7	10	5	9	11	12	6	3	8	4	2	1	6	8	5
[8	12	5	1	8	12	5	1	8	12	5	1	9	4	15
[9	3	1	9	3	1	9	3	1	9	3	1	8	4	2
[10	9	12	3	4	1	10	9	12	3	4	1	11	8	12
[11	4	5	3	7	12	2	9	8	10	6	1	7	9	14
[12	1	12	1	12	1	12	1	12	1	12	1	14	15	10
[13	16	4	1	13	16	4	1	13	16	4	1	13	16	4
[14	9	7	13	12	15	6	16	3	8	10	4	5	2	11
[15	4	9	16	2	13	8	1	15	4	9	16	2	13	8
[16	1	16	1	16	1	16	1	16	1	16	1	16	1	16

Grau: Hier findet man in der 2. Spalte ($2 \cdot 2^0$) keine 1 bzw. -1 . Daraufhin betrachtet man die 4. Spalte ($2 \cdot 2^1$), wo sich eine 1 befindet. In der letzten Spalte ist ebenfalls eine 1 zu finden. Deshalb ist die Zahl eine Primzahl.

Nun schauen wir uns im Gegensatz dazu eine Matrix von keiner Primzahl, 21, an.

[1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1]	
[2	4	8	3	6	12	11	9	5	10	7	1	15	13	9	1	11	1	2	4]
[3	9	1	3	9	1	3	9	1	3	9	1	12	2	6	1	12	15	3	9]
[4	3	12	9	10	1	4	3	12	9	10	1	4	16	13	1	16	1	4	16]
[5	12	8	1	5	12	8	1	5	12	8	1	3	15	7	1	17	1	5	4]
[6	10	8	9	2	12	7	3	5	4	11	1	10	9	3	1	6	15	6	15]
[7	10	5	9	11	12	6	3	8	4	2	1	6	8	5	1	7	7	7	7]
[8	12	5	1	8	12	5	1	8	12	5	1	9	4	15	1	8	1	8	1]
[9	3	1	9	3	1	9	3	1	9	3	1	8	4	2	1	18	15	9	18]
[10	9	12	3	4	1	10	9	12	3	4	1	11	8	12	1	19	1	10	16]
[11	4	5	3	7	12	2	9	8	10	6	1	7	9	14	1	2	1	11	16]
[12	1	12	1	12	1	12	1	12	1	12	1	14	15	10	1	3	15	12	18]
[13	16	4	1	13	16	4	1	13	16	4	1	13	16	4	1	13	1	13	1]
[14	9	7	13	12	15	6	16	3	8	10	4	5	2	11	1	14	7	14	7]
[15	4	9	16	2	13	8	1	15	4	9	16	2	13	8	1	15	15	15	15]
[16	1	16	1	16	1	16	1	16	1	16	1	16	1	16	1	4	1	16	4]
[17	16	20	4	5	1	17	16	20	4	5	1	17	16	20	4	5	1	17	16]
[18	9	15	18	9	15	18	9	15	18	9	15	18	9	15	18	9	15	18	9]
[19	4	13	16	10	1	19	4	13	16	10	1	19	4	13	16	10	1	19	4]
[20	1	20	1	20	1	20	1	20	1	20	1	20	1	20	1	20	1	20	1]

Lila: Hier findet man in der 10. Spalte ($5 \cdot 2^1$) eine 1, allerdings befindet sich in der 5. Spalte ($5 \cdot 2^0$) weder eine 1 noch eine -1 . Somit kann diese Rechnung nicht stimmen.

Blau: Hier findet man in der 5. Spalte weder eine 1 noch eine -1 . Deshalb betrachtet man die 10. Spalte, die Mittlere: Da hier auch keine 1 ist, kann in der letzten Spalte auch unmöglich eine 1 sein. Somit kann diese Rechnung auch nicht stimmen.

Diese Schleife sucht nun die Spalten von k bis 0 nach -1 ab. Falls dies zutrifft, wäre das zum Quadrat wieder 1.

```
while k>0 do (if power_mod(x,q*2^k,p)=p-1 then isp:1, k:k-1)
```


Zusammengefasst als Funktion sieht das dann so aus:

3. Quellcode

```
milller_rabin(p,r):=block ( [q,k,x,isp],
for i: 1 thru r do
q:p-1,
k:0,
isp:-1,
while evenp(q) do (k:k+1, q:q/2),
x: random(p-2)+2,
if (power_mod(x,q,p)=1 or power_mod(x,q,p)=-1) then isp:1,
if isp=-1 then while k>0 do (if power_mod(x,q*2^k,p)=p-1 then
isp:1, k:k-1),
isp
)$
```

Zur Erklärung: mit dem Parameter r kann man angeben, wie oft man den Test ausführen möchte. Mit jedem weiteren Test erhöht sich die Wahrscheinlichkeit um 75%. Wenn man den Test 25 mal ausführt, ist die Wahrscheinlichkeit, dass ein Fehler auftritt kleiner als ein Hardwarefehler.

isp ist die Variable, die voreingestellt auf -1 ist (= keine Primzahl). Falls während des Tests festgestellt wird, dass diese Zahl eine Primzahl ist, wird sie auf 1 gestellt.

4. Quellenangaben

http://en.wikipedia.org/wiki/Miller-Rabin_primality_test

<http://dkicomp.dki.tu-darmstadt.de/~robert/maxima-einfuehrung.html>

Informatik-Unterricht RSA