Proseminar SS08 - Miller-Rabin-Primzahltest

von André Dau
Technische Unviersität München
Vorlesung Perlen der Informatik 2, 2008
Professor Tobias Nipkow

17. April 2008

Inhaltsverzeichnis

1 Einleitung

Schon seit langem ist das Problem der Primfaktorzerlegung und der Bestimmung von Primzahlen ein wichtiges Thema der Zahlentheorie. Nicht nur als Forschungsgebiet der reinen Mathematik sind Primzahlen interessant, auch in der Praxis spielen sie eine wichtige Rolle. Beispielsweise beruhen einige kryptologische Verfahren darauf, dass man relativ schnell große Primzahlen finden, aber nur schwer Primfaktoren bestimmen kann. Ein prominentes Beispiel hierfür ist das RSA-Kryptographie-Verfahren.
Aus diesem Grund versucht man Algorithmen zu finden, die effizient Primaltität bestimmen können. Als effiziente Tests bezeichnet man im Allgemeinen Algorithmen, die Primaltiät in polynomieller Zeit zur Bitlänge der zu testenden Zahl entscheiden.
Bis zur Veröffentlichung des AKS-Testes im Jahre 2002 war kein effizienter vollständig verifzierter deterministischer Algorithmus bekannt.
1976 veröffentlichte Gary L. Miller einen deterministischen Algorithmus, der unter Annahme der erweiterten Riemannsche Hypothese(ERH) in Zeit $\mathbf{O}(log^4 n)$ entscheiden kann, ob eine Zahl n prim ist und damit in polynomieller Zeit durchläuft. Jedoch beruhte er auf der unbewiesenen ERH.
Rabin veröffentlichte 1977 eine abewandelte Form von Millers Algorithmus, der nicht mehr die ERH vorraussetzt und eine Laufzeit in $\mathbf{O}(log^3 n)$ hat. Jedoch ist der Algorithmus probablisitisch. Es besteht also eine kleine Wahrscheinlichkeit, dass eine als prim erkannte Zahl in Wirklichkeit nicht prim ist. Mit Rabins Algorithmus kann man relativ sicher Primalität effizient bestimmen.
Im Folgenden wird dieser als Miller-Rabin Primzahltest bekannte Algorithmus vorgestellt, untersucht und seine Korrektheit bewiesen.
Der Miller-Rabin-Test ist heute weit verbreitet und wird in zahlreichen kryptographischen Programmen verwendet.

2 Primzahltests

2.1 Zusammengesetztheit und Primalität

Eine Primzahl p ist eine positive ganze Zahl mit $p \geq 2$, die genau zwei verschiedene Teiler besitzt, nämlich die 1 und p.
Eine positive ganze Zahl $n \geq 2$ ist genau dann prim, wenn n eine Primzahl ist. Andernfalls nennt man sie zusammengesetzt, da sie sich als Produkt von mindestens zwei Primzahlen schreiben lässt.
Die 1, die 0 und alle negativen ganzen Zahlen sind weder prim noch zusammengesetzt.

2.2 Naives Dividieren

Die naheliegendste und einfachste Methode eine positive ganze Zahl $n > 2$ auf Zusammengesetztheit zu überprüfen besteht darin, sie durch alle positiven ganzen Zahlen $k \leq \sqrt{n}$ zu teilen. Indem man nur durch 2 und ungerade Zahlen teilt, reduziert man zusätzlich den Aufwand.
Offensichtlich ist diese Methode sehr ineffizient, liefert jedoch immer korrekte Ergebnisse und zusätzlich alle Teiler von n.

2.3 Determistische Primzahltests

Deterministische Primzahltests liefern immer ein korrektes Ergebnis. Sie liefern damit einen Beweis für die Zusammengesetztheit oder Primalität der zu testenden Zahl. Das naive Dividieren stellt einen deterministischen Primzahltest dar.
Mittlerweile gibt es Primzahltests, die eine polynomielle Laufzeit bezüglich der Bitlänge der zu testenden Zahl besitzen. Im Jahre 2002 wurde mit dem AKS-Primzahltest erstmals ein solcher Test vorgestellt.

2.4 Probabilistische Primzahltests

Ein probabilistischer Primzahltest enthält, wie der Name schon sagt, eine Zufallskomponente. Bei probabilistischen Tests besteht eine gewisse Wahrscheinlichkeit, dass der Algorithmus ein falsches Ergebnis liefert.
Im Allgemeinen sind probabilIstische Tests effizienter als deterministische Tests. Sie werden deshalb bei der Primzahlsuche verwendet, um Primzahlkandidaten zu sondieren, bevor ein deterministischer Test das Ergebnis überprüft.
Beim Miller-Rabin-Verfahren handelt es sich um einen probabilistischen Primzahltest.

2.4.1 Zeugen, Lügner und Pseudoprimzahlen

Viele probabilistische Primzahltestalgorithmen benötigen als Eingabe neben der zu testenden Zahl n eine Hilfsbasis a. Man sagt a ist ein Zeuge für die Zusammengesetztheit von n, falls der Primzahltest unter Verwendung der Basis a die Zahl n als zusammengesetzt erkennt.
Umgekehrt bezeichnet man a als Lügner, falls n zusammengesetzt ist, der Primzahltest jedoch unter Verwendung von a die Zahl n als "prim" bezeichnet.
Eine Zahl n ist eine Pseudoprimzahl zur Basis a, falls a ein Lügner bezüglich n ist.

3 Definitionen

Definition 1 ***Kongruenz***
Zwei ganze Zahlen $a, b \in \mathbb{Z}$ heißen kongruent modulo n mit $n \in \mathbb{Z}^+$ genau dann, wenn sie den selben Rest modulo n lassen. Formal bedeutet dies:

$$a \equiv b \ (mod\ n) \ :\Leftrightarrow n|(a-b) \tag{1}$$

Definition 2 ***Äquivalenzklassen***
Eine Äquivalenzklasse $[a]_n$ enthält anschaulich alle ganzen Zahlen, die modulo einer positiven ganzen Zahl n den selben Rest wie a lassen:

$$[a]_n := \{z|z \in \mathbb{Z} \wedge z \equiv a \ (mod\ n)\ \} \tag{2}$$

mit $a \in \mathbb{Z}$, $n \in \mathbb{Z}^+$
Man nennt a einen Repräsentanten der Äquivalenzklasse $[a]_n$.
Im Folgenden werden die eckigen Klammern weggelassen, wenn aus dem Kontext eindeutig hervorgeht, dass es sich um eine Äquivalenzklasse und nicht um eine ganze Zahl handelt.

Definition 3 ***Addition und Multiplikation von Äquivalenzklassen***

$$[a]_n + [b]_n := [a+b]_n \tag{3}$$
$$[a]_n \cdot [b]_n := [a \cdot b]_n \tag{4}$$

mit $a, b \in \mathbb{Z}$, $n \in \mathbb{Z}^+$
Die Verknüpfungen sind wohldefiniert, da für alle $k_a, k_b \in \mathbb{Z}$ gilt:

$$(k_a n + a) + (k_b n + b) \equiv (k_a + k_b)n + (a+b) \qquad (mod\ n) \tag{5}$$
$$\equiv a + b \qquad (mod\ n) \tag{6}$$

Entsprechendes gilt für die Multiplikation.

Definition 4 ***Vergleichsoperatoren bei Äquivalenzklassen***
Im Rahmen dieser Arbeit wird der Vergleichsoperator über den kleinsten nichtnegativen Repräsentanten einer Äquivalenzklasse definiert. Dieser kleinste Repräsentant entspricht anschaulich dem Rest, den ein Element der Äquivalenzklasse bei der Division durch n lässt.

$$[a]_n < [b]_n :\Leftrightarrow min(\{r|r \in [a]_n \wedge r \geq 0\}) < min(\{s|s \in [b]_n \wedge s \geq 0\}) \tag{7}$$
$$[a]_n > [b]_n :\Leftrightarrow min(\{r|r \in [a]_n \wedge r \geq 0\}) > min(\{s|s \in [b]_n \wedge s \geq 0\}) \tag{8}$$

mit $a, b \in \mathbb{Z}$, $n \in \mathbb{Z}^+$

Definition 5 mod ***-Operator***
Innerhalb von Algorithmen wird mod *auch als Operator benutzt. a* mod *b liefert den Rest bei der Division von a durch b.*
Das Ergebnis erg der Operation a mod *b wird hier formal wie folgt definiert:*

$$erg \equiv a \ (mod\ b) \tag{9}$$
$$mit\ 0 \leq erg \leq b-1 \tag{10}$$

Definition 6 ***Restklassenring*** $(\mathbb{Z}_n, +, \cdot)$
Der Restklassenring $(\mathbb{Z}_n, +, \cdot)$ *enthält anschaulich die Menge aller möglichen Reste modulo* n *mit den oben definierten Verknüpfungen der Addition und Multiplikation.*

$$\mathbb{Z}_n := \{[a]_n | a \in \mathbb{Z} \wedge 0 \leq a \leq n-1\} \tag{11}$$

Vereinbarung 1 ***Größter gemeinsamer Teiler***
Im Folgenden wird mit $ggT(a, b)$ *der größte gemeinsame Teiler von* a *und* b *bezeichnet.* a *und* b *sind genau dann teilerfremd, wenn* $ggT(a, b) = 1$.

Definition 7 ***Multiplikative Gruppe modulo*** n $(\mathbb{Z}_n^*, \cdot)$
$\mathbb{Z}_n^*$ *enthält alle Äquivalenzklassen* $[a]_n$ *mit* $1 \leq a \leq n-1$*, die zu* n *teilerfremd sind. Unter der Multiplikation bildet* $\mathbb{Z}_n^*$ *eine abelsche Gruppe, was hier jedoch nicht bewiesen wird und im weitern Verlauf auch nicht wichtig ist.*

$$\mathbb{Z}_n^* := \{[a]_n | a \in \mathbb{Z} \wedge 1 \leq a \leq n-1 \wedge ggT(a, n) = 1\} \tag{12}$$

4 Fermat-Test

Der Fermat-Test beruht auf dem kleinen Satz von Fermat. Es gibt verschiedene Varianten dieses Tests.
Gegenüber dem Miller-Rabin-Test besitzt er einige gravierende Schwächen, die später erläutert werden.

4.1 Kleiner Satz von Fermat

Theorem 1 ***(Kleiner Satz von Fermat)***
Für alle Primzahlen $p \in \mathbb{Z}^+$ und alle $a \in \mathbb{Z}$ gilt:

$$a^p \equiv a \ (mod\ p) \tag{13}$$

Beweis per vollständiger Induktion über a
Sei $p \in \mathbb{Z}^+$ eine Primzahl.

Induktionsanfang für $a = 1$
Es gilt offensichtlich $1^p \equiv 1$ (mod p)

Induktionsannahme: Es gelte $a^p \equiv a$ (mod p)

Induktionsschluss: Dann gilt auch $(a+1)^p \equiv a+1$ (mod p)
Denn es gilt nach dem binomischen Lehrsatz:

$$(a+1)^p = a^p + 1^p + \sum_{k=1}^{p-1} \binom{k}{p} a^k \cdot 1^{p-k} \tag{14}$$

Die Binomialkoeffizienten in der Summe haben für $k \in \mathbb{Z}^+, 1 \leq k \leq p-1$ die Form

$$\binom{k}{p} = \frac{p \cdot (p-1) \cdot \ldots \cdot (p-k+1)}{k \cdot (k-1) \cdot \ldots \cdot 1} \tag{15}$$

$$\text{mit } \binom{k}{p} \in \mathbb{Z}^+ \tag{16}$$

Für $k = 1$ ist $\binom{k}{p} = p$ und damit ist $\binom{k}{p}$ in diesem Fall durch p teilbar.
Für $k > 1$ gilt: Da p prim ist, hat p nur genau zwei Teiler, also gilt:

$$\frac{(p-1) \cdot (p-2) \cdot \ldots \cdot (p-k+1)}{k \cdot (k-1) \cdot \ldots \cdot 1} \in \mathbb{Z}^+ \tag{17}$$

$$\text{also } p \mid \left(p \cdot \frac{(p-1) \cdot (p-2) \cdot \ldots \cdot (p-k+1)}{k \cdot (k-1) \cdot \ldots \cdot 1} \right) = \binom{k}{p} \tag{18}$$

$$\Rightarrow \binom{k}{p} \equiv 0 \text{ (mod p)} \tag{19}$$

Damit ergeben sich folgende Äquivalenzen

$$(a+1)^p \equiv a^p + 1^p + \sum_{k=1}^{p-1} \binom{k}{p} a^k \cdot 1^{p-k} \text{ (mod p)} \tag{20}$$

Induktions-Annahme und Gl.(19) (21)

$$\equiv a + 1 + \sum_{k=1}^{p-1} 0 \cdot a^k \cdot 1^{p-k} \text{ (mod p)} \tag{22}$$

$$\equiv a + 1 \text{ (mod p)} \tag{23}$$

Gemäß dem Prinzip der vollständigen Induktion ist der kleine Satz von Fermat damit bewiesen.

Theorem 2 ***(Korollar des kleinen Fermatschen Satzes)***
Für alle Primzahlen $p \in \mathbb{Z}^+$ und alle ganzen Zahlen $a \in [1, p-1]$ gilt:

$$a^{p-1} \equiv 1 \text{ (mod } p) \tag{24}$$

Beweis
Der Beweis ergibt sich durch Umformung der Fermatschen Gleichung. Da $a \in [1, p-1]$, ist a kein Vielfaches von p, also $a \not\equiv 0$ (mod p) . Also darf man beide Seiten von Gl.(13) durch a dividieren und erhält Th.(2).

4.2 Algorithmus des Fermat-Testes

Bei dem Fermatschen Primzahltest wird für eine zu prüfende positive ganze Zahl $n > 2$ die Einhaltung des kleinen Satzes von Fermat, bzw. dessen Korollar zu einer Testbasis a geprüft. In folgendem Algorithmus ist $a = 2$. Da für eine Eingabe $n$zusätzlich $n > 2$ gefordert wird, gilt somit in jedem Fall $a \in [1, n-1]$, wodurch Th.(2) verwendet werden kann.
Wird der Satz nicht eingehalten, so ist die 2 ein Zeuge für die Zusammengesetztheit von n. n ist dann definitiv nicht prim. Ansonsten ist die Zahl möglicherweise eine Primzahl. Die Unsicherheit des Testes entsteht durch die Existenz von zusammengesetzten Zahlen, die für einige Werte von a die Gleichung von Fermat erfüllen. Diese Zahlen nennt man Fermatsche Pseudoprimzahlen, da sie hinsichtlich des Fermat-Testes zum Teil ähnliche Eigenschaften wie Primzahlen besitzen.

Funktion *isComposite*
Eingabe: positive ganze Zahl $n > 2$
Ausgabe: *true* , falls 2 ein Zeuge für die Zusammengesetztheit von n ist
false , falls n eine Primzahl oder eine Pseudoprimzahl zur Basis 2 ist
Algorithmus: **if** $2^{n-1} \not\equiv 1$ (mod n)
then return *true*
else return *false*

4.3 Fehlerrate des Fermat-Testes

Die Fehlerrate des Fermat-Testes ist sogar recht gering. Es gibt beispielsweise bis zu einer Grenze von 10.000 nur 22 Werte bei denen der Fermat-Test mit Testbasis $a = 2$ irrt. Die Irrtumswahrscheinlichkeit bei einer zufällig ausgewählten Zahl n mit Länge β geht für $\beta \to \infty$ gegen null.
Damit eignet sich der Test bedingt, um im Umfeld einer zufällig gewählten Zahl Primzahlen zu finden.
Indem man zusätzlich andere Basen als die 2 testet und die zu testende Basis zufällig wählt, kann man den Test zwar verbessern, jedoch nicht alle Fehler beseitigen. Es existieren nämlich so genannte Carmichaelzahlen, die die Gleichung von Fermat für alle $a \in \mathbb{Z}_n^*$ erfüllen:

$$a^{n-1} = [1]_n \text{ , für alle } a \in \mathbb{Z}_n^* \tag{25}$$

Die Wahrscheinlichkeit bei großen Carmichaelzahlen zufällig eine Basis $b \in [1, n-1]$ mit $[b]_p \in \mathbb{Z}_n - \mathbb{Z}_n^*$ zu wählen ist sehr gering. Carmichaelzahlen sind also besonders schwierige Pseudoprimzahlen für den Fermat-Test. Die geringe Fehlerrate des Fermat-Test für zufällig gewählte Eingabezahlen liegt an der Seltenheit der Carmichaelzahlen. Die ersten Carmichaelzahlen sind 561, 1105 und 1729. Carmichaelzahlen sind aus mindestens drei teilerfremden ungeraden Primfaktoren zusammengesetzt und erfüllen noch weitere Eigenschaften. Deshalb ist ihre Dichte relativ gering. Man kann jedoch zeigen, dass es unendlich viele Carmichaelzahlen gibt.
Der Fermat-Test eignet sich also nicht als zuverlässiger Primzahltest.

5 Miller-Rabin-Primzahltest

5.1 Entstehung

Der Miller-Rabin-Test wird auch Miller-Selfridge-Rabin-Test genannt, weil Selfridge einen ähnlichen Algorithmus wie den Millers bereits zwei Jahre vor Millers Veröffentlichung verwendete. 1976 schlug Miller in einer Arbeit einen Algorithmus vor, der zum Teil auf dem kleinen Satz von Fermat fußte, jedoch einen zusätzlichen Test einführte, der das Problem der Carmichaelzahlen beheben sollte.
Ähnlich wie beim Fermat-Test prüft Millers Algorithmus verschiedene Basen für die zu testende Zahl. In seiner Originalversion war der Test deterministisch. Es wurden systematisch Basen bis zu einer bestimmten Grenze geprüft. Die Grenze ergab sich aus der erweiterten Riemannschen Hypothese.
Rabin änderte den Algorithmus zum heute bekannten Miller-Rabin-Test, indem er nur für einzelne zufällig gewählte Basen prüfte, ähnlich wie beim randomisierten Fermat-Test. Dadurch war der Test nicht mehr von der unbewiesenen ERH abhängig, aber auch nicht mehr deterministisch. Es bestand nun die Wahrscheinlichkeit einer falschen Ausgabe.
Im Gegensatz zum Fermat-Test gibt es für den Miller-Rabin-Test keine problematischen Eingaben, für die der Test immer falsche Ergebnisse liefert. Im Gegenteil lässt sich die obere Schranke für die Irrtumswahrscheinlichkeit unabhängig von der Eingabezahl auf Kosten einer etwas höheren Laufzeit beliebig senken. In seiner Arbeit machte Rabin genaue Aussagen über die obere Schranke für die Fehlerrate.
Ein entscheidender Vorteil von Rabins Änderung gegenüber Millers Version war die schnellere Laufzeit bei großen Eingabezahlen und der Verzicht auf die unbewiesene erweiterte Riemannsche Hypothese.
Der Miller-Rabin-Test ist heute weit verbreitet.

5.2 Grundlegende mathematische Sätze

Der Miller-Rabin-Test beruht im Wesentlichen auf zwei Theoremen über Primzahlen.
Der erste Satz ist der kleine Satz von Fermat, der bereits bewiesen wurde.
Nachfolgend der zweite Satz:

Theorem 3 *Sei p eine Primzahl. Innerhalb von $\mathbb{Z}_p$ besitzt $[1]_p$ keine nicht-triviale Wurzel. Eine nicht-triviale Wurzel ist ein Element $m \in \mathbb{Z}_p$ mit $m \neq [1]_p$ und $m \neq [-1]_p$, sodass $m^2 = [1]_p$. Anders ausgedrückt bedeutet dies:*

$$(z^2 = 1) \Rightarrow (z = [\pm 1]_p) \text{ ,mit } z \in \mathbb{Z}_p, \ p \text{ Primzahl} \tag{26}$$

Beweis
Sei p eine Primzahl. Sei x eine Quadratwurzel von $[1]_p$:

$$x^2 = 1 \tag{27}$$
$$\Leftrightarrow \quad x^2 - 1 = 0 \tag{28}$$
$$\Leftrightarrow \quad (x+1)(x-1) = 0 \tag{29}$$

mit $x \in \mathbb{Z}_p$

Fall 1: $1 < x < p - 1$
Es gilt dann $1 \leq x - 1 < x + 1 \leq p - 1$. $(x - 1)$ und $(x + 1)$ sind wegen der Primalität von p damit in jedem Fall teilerfremd zu p. Also ist auch $(x + 1) \cdot (x - 1)$ teilerfremd zu p.

$$(x + 1)(x - 1) \neq [p]_p = [0]_p \tag{30}$$

Für Fall 1 ist x keine Lösung von Gl.(29).

Fall 2: $x = [1]_p$ oder $x = [-1]_p$
In diesem Fall ist x eine Lösung für Gl.(29), wie man durch Einsetzen sieht.

Damit sind alle Fälle für x abgedeckt. Es gibt also genau zwei Lösungen $x = [1]_p$ und $x = [-1]_p$ für Gl.(27), falls p prim ist.

5.3 witness-Funktion

Die *witness*-Funkion ist der Kern des Algorithmus. Falls sie *true* zurückliefert, so ist die zu testende Zahl auf jeden Fall zusammengesetzt. Ansonsten ist sie prim oder pseudoprim.
Es gibt verschiedene Varianten der *witness*-Funktion, die aber alle auf Th.(1)-(3) beruhen.

5.3.1 Algorithmus der witness-Funktion

Funktion *witness(a,n)*
Eingabe: positive ungerade ganze Zahl $n > 2$, die getestet werden soll
positive ganze Zahl a mit $1 \leq a \leq n - 1$ als Testbasis
Ausgabe: *true* , falls n als zusammengesetzt erkannt wird
false , falls n prim oder pseudoprim zur Basis a ist
Algorithmus:
1– Finde $s, d \in \mathbb{Z}^+$, sodass d ungerade ist und $n - 1 = 2^s \cdot d$
2– **if** $a^d \not\equiv 1$ (mod n) und für alle ganzen Zahlen $r \in [0, s - 1]$ gilt: $a^{2^r \cdot d} \not\equiv -1$ (mod n)
3– **then return** *true*
4– **else return** *false*

5.3.2 Korrektheit der witness-Funktion

Im Folgenden wird gezeigt, dass n auf jeden Fall zusammengesetzt ist, falls die *witness*-Funktion *true* zurückliefert. Liefert sie *false* zurück, so kann n prim oder zusammengesetzt sein.

Beweis
Die *witness*-Funktion liefert genau dann *true* zurück, falls für n bei einer zufällig gewählten ganzen Zahl $a \in [1, n - 1]$ gilt:

$$a^d \not\equiv 1 \text{ (mod n)} \tag{31}$$

$$a^{2^r \cdot d} \not\equiv -1 \text{ (mod n)} \quad \text{, für alle ganzen Zahlen } r \in [0, s - 1] \tag{32}$$

Angenommen die *witness*-Funktion gibt *true* zurück.
Da gefordert wird, dass n ungerade ist, ist $n - 1$ gerade und durch 2 teilbar. Also ist $s \geq 1$. Die Intervalle $[0, s - 1]$ und $[1, s]$ sind damit nicht leer.

Mann kann dann zwischen verschiedenen Fällen unterscheiden:
Fall 1: Es existiert eine ganze Zahl $t \in [1, s]$, sodass $a^{2^t \cdot d} \equiv 1 \text{ (mod n)}$
Sei

$$t_m := min(\{t | (t \in \mathbb{Z}) \wedge (1 \leq t \leq s) \wedge (a^{2^t \cdot d} \equiv 1 \text{ (mod n) })\}) \tag{33}$$

Wegen $0 \leq t_m - 1 \leq s - 1$ und Gl.(32) gilt:

$$a^{2^{t_m - 1} \cdot d} \not\equiv -1 \text{ (mod n)} \tag{34}$$

Es ist $0 \leq t_m - 1 \leq s - 1$.
Für $t_m - 1 = 0$ gilt untenstehende Gleichung wegen $2^0 \cdot d = d$ und Gl.(31).
Für $1 \leq t_m - 1 \leq s - 1$ gilt sie wegen der Minimalität von t_m:

$$a^{2^{t_m - 1} \cdot d} \not\equiv 1 \text{ (mod n)} \tag{35}$$

Wegen $a^{2^{t_m} \cdot d} \equiv 1 \text{ (mod n)}$ und

$$\begin{aligned} a^{2^{t_m - 1} \cdot d} \cdot a^{2^{t_m - 1} \cdot d} &= a^{2^{t_m - 1} \cdot d + 2^{t_m - 1} \cdot d} && (36) \\ &= a^{2 \cdot (2^{t_m - 1} \cdot d)} && (37) \\ &= a^{2^{t_m} \cdot d} && (38) \end{aligned}$$

wäre aber $a^{2^{t_m - 1} \cdot d}$ eine nicht-triviale Wurzel von 1 modulo n.
Damit kann n in diesem Fall wegen Th.(3) keine Primzahl sein.

Fall 2: Es existiert keine ganze Zahl $t \in [1, s]$, sodass $a^{2^t \cdot d} \equiv 1 \text{ (mod n)}$
Damit wäre insbesondere $a^{2^s \cdot d} \not\equiv 1 \text{ (mod n)}$.
Wegen $2^s \cdot d = n - 1$ gilt damit $a^{n-1} \not\equiv 1 \text{ (mod n)}$. Da $a \in [1, n-1]$ kann n wegen dem kleinen Satz von Fermat in diesem Fall keine Primzahl sein.

Damit wurde für alle Fälle die Zusammengesetztheit von n gezeigt, falls *witness true* zurückliefert.

5.3.3 Starke Pseudoprimzahl

Eine zusammengesetzte Zahl n, die von der *witness*-Funktion für eine Testbasis a nicht als zusammengesetzt erkannt wird, nennt man eine starke Pseudoprimzahl zur Basis a.

5.4 Algorithmus des Miller-Rabin-Tests

Den Miller-Rabin-Algorithmus gibt es in verschiedenen Varianten, die jedoch im Kern alle gleich sind. Der zentrale Bestandteil des Algorithmus ist die *witness*-Funktion.
In dieser Variante werden alle geraden Zahlen direkt behandelt. Die *witness*-Funktion wird nur bei ungeraden Zahlen verwendet. Dies erleichtert später den Beweis zur Irrtumswahrscheinlichkeit und ist auch in der Praxis eine sinnvolle und leicht zu realisierende Abfrage.
Während des Algorithmus wird die *witness*-Funktion bis zu k-mal mit verschiedenen Testbasen aufgerufen, wobei k frei wählbar ist und die Fehlerrate mit steigendem k sinkt. Wird die zu testende Zahl in keinem der k-Durchgänge als zusammengesetzt erkannt, ist die zu testende Zahl mit hoher Wahrscheinlichkeit prim.

Funktion *Miller-Rabin-Test(k,n)*
Eingabe: positive ganze Zahl k; gibt die Anzahl der Testbasen für die *witness*-Funktion an
positive ganze Zahl $n > 1$, die getestet werden soll
Ausgabe: *composite* , falls n als zusammengesetzt erkannt wird
prime , falls n prim oder eine ungerade starke Pseudoprimzahl zu allen getesteten Basen ist
Algorithmus:
1– **if** n gerade
2– **then if** $n = 2$
3– **then return** *prime*
4– **else return** *composite*
5– **for** $j \longleftarrow 1$ **to** k
6– **do** Wähle zufällig eine ganze Zahl $a \in [1, n-1]$
7– **if** *witness(a,n)=true*
8– **then return** *composite*
9– **return** *prime*

5.5 Fehlerrate des Miller-Rabin-Tests

Man kann zeigen, dass die obere Grenze für die Irrtumswahrscheinlichkeit des Miller-Rabin-Testes exponentiell zu der Anzahl der verwendeten Testbasen sinkt:

$$Irrtumswahrscheinlichkeit < \left(\frac{1}{4}\right)^k \tag{39}$$

,wobei k die Anzahl der Iterationen des Miller-Rabin-Algorithmus ist.
Die obige Gleichung über die Irrtumswahrscheinlichkeit besagt, dass die Wahrscheinlichkeit, dass der Miller-Rabin-Test mit k-Iterationen bei einer zusammengesetzten Zahl *prime* ausgibt kleiner als $\left(\frac{1}{4}\right)^k$ ist.
Im Gegensatz zum Fermat-Test ist die obere Schranke für die Irrtumswahrscheinlichkeit nicht mehr von der Eingabe abhängig, sondern von der Anzahl der Iterationen.
Der genaue Beweis ist recht aufwendig und erfordert viel Mathematik aus der Zahlentheorie. Nachfolgend soll der Beweis skizziert werden.

Bei geraden ganzen Zahlen liefert der Algorithmus wegen den Zeilen $1-4$ immer ein korrektes Ergebnis.
Da die *witness*-Funktion wie oben gezeigt korrekt ist, liefert der Algorithmus bei allen Primzahlen immer ein korrektes Ergebnis.
Im Intervall $[1, 8]$ sind alle ganzen Zahlen entweder prim oder gerade, wie sich leicht nachprüfen lässt. In diesem Intervall liefert der Miller-Rabin-Test somit immer sicher korrekte Ergebnisse.
Es genügt also, die Irrtumswahrscheinlichkeit bei ungeraden zusammengesetzten Zahlen ≥ 9 zu untersuchen.
Zur Untersuchung der Irrtumswahrscheinlichkeit braucht man einen Hilfssatz, der hier aber nicht bewiesen wird.

Theorem 4 *Sei n eine ungerade zusammengesetzte positive ganze Zahl mit $n \geq 9$. Dann gibt es im Intervall $[1, n-1]$ weniger als $\frac{1}{4}(n-1)$ ganze Zahlen, die keine Zeugen für die Zusammengesetztheit von n sind.*

Sei m eine zusammengesetzte ungerade ganze Zahl mit $m \geq 9$. Sei $L_m \subseteq [1, m-1]$ die Menge der ganzen Zahlen, die keine Zeugen für die Zusammengesetztheit von m sind. m wird nur dann nicht als zusammengesetzt erkannt, falls in jedem Iterationsdurchgang des Algorithmus ein $l \in L_m$ als Testbasis gewählt wird. Da in jeder Iteration die Testbasis zufällig aus dem Intervall $[1, m-1]$ gewählt wird, gilt für die Wahrscheinlichkeit p, dass der Miller-Rabin-Test mit k Iterationen für m *prime* ausgibt:

$$p = \left(\frac{|L_m|}{m-1}\right)^k \tag{40}$$

Laut Th.(4) gilt für $|L_m|$

$$|L_m| < \frac{1}{4}(m-1) \tag{41}$$

Und damit gilt:

$$p = \left(\frac{|L_m|}{m-1}\right)^k \tag{42}$$

$$< \left(\frac{1}{4}\right)^k \tag{43}$$

Damit wäre die eingangs gestellte Aussage über die Irrtumswahrscheinlichkeit bewiesen.

5.6 Laufzeit

Unter Verwendung von modularer Exponentiation durch wiederholtes Quadrieren liegt die Laufzeit in $O(log^3 n)$, wobei n die zu testende Zahl ist. Bezogen auf die Bitlänge der zu testenden Zahl hat der Algorithmus damit kubische Laufzeit, und kann damit als relativ effizienter polynomieller Algorithmus bezeichnet werden.
Nachfolgend ein einfacher Algorithmus für modulare Exponentiation durch wiederholtes Quadrieren:

Funktion *modExp(b,x,n)*
Eingabe: nichtnegative ganze Zahl b; Basis
nichtnegative ganze Zahl x; Exponent
positive ganze Zahl n; modulo n
Ausgabe: Ergebnis d mit $d \equiv b^x \pmod{n}$ und $0 \leq d \leq n-1$
Algorithmus:
1– $d \longleftarrow 1$
2– Sei $< x_k, x_{k-1}, \ldots, x_0 >$ die Binärdarstellung von x
3– **for** $i \longleftarrow k$ **downto** 0
4– **do** $d \longleftarrow (d \cdot d) \mod n$
5– **if** $b_i = 1$
6– **then** $d \longleftarrow (b \cdot d) \mod n$
7– **return** d

Durch Optimierung der Multiplikation liegt die Laufzeit des Miller-Rabin-Testes sogar in $O(\log^2 n \cdot \log\log n \cdot \log\log\log n)$.

6 Schlussbemerkung

Ein anderer Standard-Test neben dem Miller-Rabin-Test ist der deterministische Lucas-Lehmer-Test. Allerdings eignet er sich in der Praxis nur zum Finden von Mersenne-Primzahlen der Form $2^n - 1$.
Seit 2002 gibt es mit dem AKS-Test einen deterministischen Primzahltest, der in polynomieller Zeit zur Bitlänge läuft. Damit wurde gezeigt, dass $PRIMES \in P$ liegt, also Primalität effizient und sicher bestimmt werden kann. Seitdem wurde dieser Test stark verbessert.
Dennoch hat der Miller-Rabin-Algorithmus immer noch eine sehr hohe Relevanz und wird in Kryptographiesystemen zusammen mit anderen Verfahren zum Finden von Primzahlen in der Größenordnung von über 100 Bit verwendet.
Die relative Simplizität des Miller-Rabin-Testes ermöglicht eine einfache und dennoch effiziente Implementierung, was ihn auch für Hobbyentwickler interessant macht.
Sogar der Fermat-Test wird heute noch verwendet, weil er einfach und schnell ist. Ein bekanntes Beispiel ist das verbreitete Programm PGP (Pretty Good Provacy) von Phil Zimmermann. Es testet einen Primzahlkandidaten zu den Basen $2, 3, 5, 7$. Da der Test nur zum Finden von großen Primzahlen genutzt wird und Carmichaelzalen extrem selten sind, ist die Wahrscheinlichkeit, dass der Test einen Fehler macht, sehr gering.

7 Literatur

[1] Brockmann, Tobias: Primzahlerzeugung / Westfälische Wilhelms-Universität Münster, Institut für Wirtschaftsinformatik, Praktische Informatik in der Wirtschaft. URL http://www-wi.uni-muenster.de/pi/lehre/ws0708/seminar/Abgaben/Primzahlerzeugung.pdf, Wintersemester 2007/08. – Seminararbeit. Im Rahmen des Seminars Multimedia

[2] Cormen, Thomas H. ; Leiserson, Charles E. ; Rivest, Ronald L. ; Stein, Clifford: *Introduction to Algorithms*. 2. MIT Pr., 1990 (The MIT electrical engineering and computer science series)

[3] Hurd, Joe: Verification of the Miller-Rabin Probabilistic Primality Test. In: *Journal of Logic and Algebraic Programming* 50 (2003), May–August, Nr. 1–2, S. 3–21. – URL http://www.cl.cam.ac.uk/~jeh1004/research/papers/miller.html. – Special issue on Probabilistic Techniques for the Design and Analysis of Systems

[4] Miller, Gary L.: Riemann's Hypothesis and Tests for Primality. In: *Journal of Computer and System Sciences* 13 (1976), Nr. 3, S. 300–317

[5] Rabin, Michael O.: Probabilistic algorithm for testing primality. In: *Journal of Number Theory* 12 (1980), Nr. 1, S. 128–138

[6] Wikipedia: *Carmichael number — Wikipedia, The Free Encyclopedia*. 2008. – URL http://en.wikipedia.org/w/index.php?title=Carmichael_number&oldid=205140568. – [Online; accessed 17-April-2008]

[7] Wikipedia: *Fermat primality test — Wikipedia, The Free Encyclopedia*. 2008. – URL http://en.wikipedia.org/w/index.php?title=Fermat_primality_test&oldid=200281272. – [Online; accessed 17-April-2008]

[8] Wikipedia: *Miller-Rabin primality test — Wikipedia, The Free Encyclopedia*. 2008. – URL http://en.wikipedia.org/w/index.php?title=Miller-Rabin_primality_test&oldid=203980152. – [Online; accessed 17-April-2008]