5 Randomisierte Algorithmen für Probleme aus der Zahlentheorie

In diesem Kapitel betrachten wir randomisierte Algorithmen zu Problemen, die mit dem Rechnen mit ganzen Zahlen zu tun haben. Dies sind ein Primzahltest, ein daraus abgeleitetes Verfahren zur Erzeugung von zufälligen Primzahlen großer Bitlänge und ein Verfahren zum Ziehen von Quadratwurzeln modulo p, wo p eine Primzahl ist. Insbesondere wegen ihrer Anwendung in der Kryptographie haben diese Verfahren sehr große praktische Bedeutung.

5.1 Fakten aus der Zahlentheorie und grundlegende Algorithmen

5.1.1 Teiler, Reste, Euklidischer Algorithmus

Unsere Zahlenbereiche:

- $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$,
- $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, 3, \ldots\}$

Wir stellen uns die Zahlen immer als zu einer passenden Basis b dargestellt vor: Binärdarstellung, Dezimaldarstellung, Hexadezimaldarstellung, Darstellung zur Basis 256 (eine Ziffer ist ein Byte) oder 2^{32} (eine Ziffer ist ein 32-Bit-Wort, passend für die Darstellung in einem Rechner).

Die Anzahl der Ziffern in der Darstellung von $a \in \mathbb{N}$ zur Basis b ist $\lceil \log_b(a+1) \rceil$.

Verwendete Operationen: Addition, Subtraktion, Multiplikation, Division (mit Rest).

Wir nehmen an, dass zwei einziffrige Zahlen in Zeit $O(1)$ addiert und subtrahiert werden können („von der Hardware“). Addition und Subtraktion zweier n-ziffriger Zahlen kostet dann Zeit $O(n)$, Multiplikation einer n-ziffrigen und einer ℓ-ziffrigen Zahl mit der Schulmethode kostet Zeit $O(n\ell)$, ebenso Division einer n-ziffrigen durch eine ℓ-ziffrige Zahl.

Fakt 5.1 ***Division mit Rest***
Zu $x \in \mathbb{Z}$ und $m \geq 2$ gibt es ein r mit $0 \leq r < m$ und ein q mit $x = qm + r$. Die Zahlen q und r sind eindeutig bestimmt.

Die Zahl r („**Rest**“) bezeichnen wir mit $\boldsymbol{x \bmod m}$. Sie hat die Darstellung $x - qm$, unterscheidet sich also von x um ein Vielfaches von m. Der „**Quotient**“ $q = \lfloor x/m \rfloor$ wird auch mit $\boldsymbol{x \operatorname{div} m}$ bezeichnet. Wir betrachten die Operationen „Addition modulo m“, definiert durch $\mathbb{Z}^2 \ni (x, y) \mapsto (x+y) \bmod m$, und „Multiplikation modulo m“, definiert durch $\mathbb{Z}^2 \ni (x, y) \mapsto xy \bmod m$.

Fakt 5.2
Für jedes $m \geq 2$ bildet die Menge $\mathbb{Z}_m = \{0, 1, \ldots, m-1\}$ mit den Operationen *Addition modulo* m und *Multiplikation modulo* m einen Ring mit 1.

Das heißt im Detail: Die Operationen $+ \pmod m$ und $\cdot \pmod m$ führen nicht aus dem Bereich $\mathbb{Z}_m$ heraus. Die Addition erfüllt alle Rechenregeln für Gruppen. Insbesondere hat jedes Element a ein additives Inverses $-a$ (beachte $-a = m - a$ für $0 < a < m$ und $-0 = 0$). Die Multiplikation hat die 1 als neutrales Element und ist assoziativ; für Addition und Multiplikation gelten die Distributivgesetze.

Lemma 5.3
Für jedes $m \geq 2$ ist die Abbildung $\mathbb{Z} \to \mathbb{Z}_m$, $x \mapsto x \bmod m$, ein Homomorphismus; d. h. für $x, y \in \mathbb{Z}$ gilt:

(i) $(x + y) \bmod m = ((x \bmod m) + (y \bmod m)) \bmod m$,

(ii) $xy \bmod m = ((x \bmod m)(y \bmod m)) \bmod m$.

Dieses Lemma wird hauptsächlich benutzt, um Rechnungen zu vereinfachen. Um einen arithmetischen Ausdruck auszuwerten (inklusive Potenzen), kann man auf beliebige Zwischenergebnisse die mod m-Operation anwenden.

Beispiel: Um $13^7 \bmod 11$ zu berechnen, rechnet man

$$(13 \bmod 11)^7 \bmod 11 = (2^5 \bmod 11) \cdot 4 \bmod 11 = (-1) \cdot 4 \bmod 11 = 7.$$

Fakt 5.4
Für jedes $m \geq 2$ gilt:
$x \bmod m = y \bmod m$ genau dann wenn $x - y$ durch m teilbar ist.

Beispiel: $29 \bmod 12 = 53 \bmod 12 = 5$ und $53 - 29 = 24$ ist durch 12 teilbar.

Schreibweise: $x \equiv y \pmod{m}$ bedeutet, dass $x - y$ durch m teilbar ist. Man liest: x ist kongruent y modulo m.

Fakt 5.5
Für jedes $m \geq 2$ gilt:

$$x_1 \equiv y_1 \pmod{m} \text{ und } x_2 \equiv y_2 \pmod{m} \;\Rightarrow\; x_1 + x_2 \equiv y_1 + y_2 \pmod{m},$$

und die analoge Feststellung für die Multiplikation.

Dies hat den Effekt, dass man bei mit „$\equiv$“ geschriebenen Transformationen von $\{+.-, \cdot\}$-Ausdrücken stets Zahlen oder Unterausdrücke durch kongruente Zahlen bzw. Unterausdrücke ersetzen kann.

Beispiel: $13^7 \equiv 2^7 \equiv 2^5 \cdot 2^2 = 32 \cdot 4 \equiv (-1) \cdot 4 \equiv -4 \equiv 7 \pmod{11}$.

Eine Zahl $y \in \mathbb{Z}$ heißt **Teiler** von x, wenn es ein z mit $x = y \cdot z$ gibt. Notation: $y \mid x$. (Wenn y nicht Teiler von x ist, schreibt man $y \nmid x$.) Bezüglich der Teilbarkeitsrelation verhalten sich x und $-x$ gleich. Man beachte, dass jede Zahl y Teiler von 0 ist, aber nur 1 und -1 Teiler von 1 sind.

Definition 5.6 ***Größter gemeinsamer Teiler***
Für $x, y \in \mathbb{Z}$ sei $\mathrm{ggT}(x, y)$ die (eindeutig bestimmte) nichtnegative Zahl d mit:

$$\text{(i) } d \mid x \text{ und } d \mid y; \quad \text{(ii) wenn } c \mid x \text{ und } c \mid y \text{ gilt, dann folgt } c \mid d.$$

Wenn x und y nicht beide 0 sind, liefert dies den größten gemeinsamen Teiler im Standardsinn (größte Zahl in $\mathbb{Z}$, die sowohl x als auch y teilt). Weiter legen wir fest[1]: $\mathrm{ggT}(0, 0) = 0$. Zwei Zahlen x und y mit $\mathrm{ggT}(x, y) = 1$ heißen ***teilerfremd***. Der größte gemeinsame Teiler zweier Zahlen ist eindeutig bestimmt. Es gibt einen wohlbekannten effizienten Algorithmus zur Ermittlung des größten gemeinsamen Teilers. (Dieser beweist auch die Existenz.) Die Feststellung $\mathrm{ggT}(x, y) = \mathrm{ggT}(|x|, |y|)$ besagt,

[1] Die Teilbarkeitsrelation | ist transitiv, also eine partielle Quasiordnung. Bezüglich dieser partiellen Quasiordnung sind 1 und -1 kleinste Elemente (sie teilen jede ganze Zahl) und 0 ist größtes Element (jede ganze Zahl teilt die 0). Im Sinn dieser partiellen Ordnung ist 0 also das größte Element, das 0 teilt. Aus dieser Sicht ergibt die Festlegung $\mathrm{ggT}(0, 0) = 0$ Sinn.

dass man sich auf den Fall $x, y \geq 0$ beschränken kann. Weiter gelten die Gleichungen

$$\begin{aligned}
&\mathrm{ggT}(a,0) = a \text{ , für beliebige } a \geq 0,\\
&\mathrm{ggT}(a,b) = \mathrm{ggT}(b,a) \text{ und}\\
&\mathrm{ggT}(a,b) = \mathrm{ggT}(b, a \bmod b) \text{ , für } b > 0. \qquad (\text{Und: } a \bmod b < b.)
\end{aligned}$$

Diese liefern ein iteratives Verfahren.

Algorithmus 5.1 ***Euklidischer Algorithmus***
Input: Zwei natürliche Zahlen x und y.
Methode:

```
1    a, b: integer; a ← x; b ← y;
2    while b > 0 repeat
3        (a, b) ← (b, a mod b);      // simultane Zuweisung
4    return a.
```

Fakt 5.7
Algorithmus 5.1 gibt $\mathrm{ggT}(x,y)$ aus. Die gesamte Anzahl von ausgeführten Bitoperationen ist $O(\log(x)\log(y))$.

Man beachte, dass $\lceil \log(x+1) \rceil \approx \log x$ die Anzahl der Bits ist, die man braucht, um x aufzuschreiben. Damit hat der Euklidische Algorithmus bis auf einen konstanten Faktor denselben Aufwand wie die Multiplikation von x und y, wenn man die Schulmethode benutzt.

Beispiel: Auf Eingabe $x = 10534$, $y = 12742$ ergibt sich der in Tab. 5.1.1 angegebene Ablauf. Die Zahlen a_i und b_i bezeichnen den Inhalt der Variablen `a` und `b`, nachdem die Schleife in Zeilen 2–3 i-mal ausgeführt worden ist. Die Ausgabe ist $46 = \mathrm{ggT}(10534, 12742)$.

Beispiel: (a) 21 und 25 sind teilerfremd. Es gilt $31 \cdot 21 + (-26) \cdot 25 = 651 - 650 = 1$.
(b) Es gilt $\mathrm{ggT}(21, 35) = 7$, und $2 \cdot 35 - 3 \cdot 21 = 7$.

Die folgende sehr nützliche Aussage verallgemeinert diese Beobachtung:

Lemma 5.8 ***... von Bezout***
(a) Wenn x, y teilerfremd sind, gibt es s, t mit $sx + ty = 1$.
(b) Für $x, y \in \mathbb{Z}$, nicht beide gleich 0, gibt es s, t mit $sx + ty = \mathrm{ggT}(x,y)$.

Die Koeffizienten s, t aus dem Lemma von Bezout kann man mit einer Erweiterung des Euklidischen Algorithmus sehr effizient berechnen. Die Rechenzeiten des folgenden

i	a_i	b_i
0	10534	12742
1	12742	10534
2	10534	2208
3	2208	1702
4	1702	506
5	506	184
6	184	138
7	138	46
8	46	0

Tabelle 1: Ablauf des Euklidischen Algorithmus auf Beispieleingabe $x = 10534$, $y = 12742$

Algorithmus sind (in O-Notation) ebenso groß wie die des gewöhnlichen Euklidischen Algorithmus.

Algorithmus 5.2 ***Erweiterter Euklidischer Algorithmus***
EINGABE: Zahlen $x, y \geq 0$.
METHODE:

```
0    a, b, sa, ta, sb, tb, q: integer;
1    (a, b) ← (x, y);
2    (sa, ta, sb, tb) ← (1, 0, 0, 1);
3    while b > 0 repeat
4        q ← a div b;
5        (a, b) ← (b, a − q · b);
6        (sa, ta, sb, tb) ← (sb, tb, sa − q · sb, ta − q · tb);
7    return (a, sa, ta);
```

Genau wie im ursprünglichen Euklidischen Algorithmus findet die eigentliche Arbeit in der **while**-Schleife statt (4 Zeilen). Die Idee hinter dem Algorithmus ist folgende. Wie im (einfachen) Euklidischen Algorithmus werden in den Variablen `a` und `b` Zahlen a, $b \geq 0$ mitgeführt, die stets $\mathrm{ggT}(a, b) = d = \mathrm{ggT}(x, y)$ erfüllen. Die Variablen $\mathtt{s_a}$, $\mathtt{t_a}$, $\mathtt{s_b}$, $\mathtt{t_b}$ enthalten immer Zahlen s_a, t_a, s_b, t_b, die folgende Gleichungen erfüllen:

$$a = s_a \cdot x + t_a \cdot y \quad \text{und} \quad b = s_b \cdot x + t_b \cdot y.$$

Diese Gleichung wird durch die Initialisierung hergestellt. In einem Schleifendurchlauf wird a durch b ersetzt und (s_a, t_a) durch (s_b, t_b), und es wird b durch $a - q \cdot b$ ersetzt sowie (s_b, t_b) durch $(s_a - q \cdot s_b, t_a - q \cdot t_b)$. Dadurch bleiben die Gleichungen gültig. Wenn schließlich $b = 0$ geworden ist, gilt $d = \mathrm{ggT}(x, y) = a = s_a \cdot x + t_a \cdot y$. Das bedeutet, dass die Ausgabe das gewünschte Ergebnis darstellt. Als Beispiel betrachten wir den Ablauf des Algorithmus auf der Eingabe $(x, y) = (10534, 12742)$. Die Zahlen a_i ,b_i, $s_{a,i}$, $t_{a,i}$, $s_{b,i}$, $t_{b,i}$ bezeichnen den Inhalt von `a`, `b`, $\mathtt{s_a}$, $\mathtt{t_a}$, $\mathtt{s_b}$, $\mathtt{t_b}$ nach dem i-ten Schleifendurchlauf. Die Zahl q_i ist der Quotient im i-ten Durchlauf.

i	a_i	b_i	$s_{a,i}$	$t_{a,i}$	$s_{b,i}$	$t_{b,i}$	q_i
0	10534	12742	1	0	0	1	–
1	12742	10534	0	1	1	0	0
2	10534	2208	1	0	−1	1	1
3	2208	1702	−1	1	5	−4	4
4	1702	506	5	−4	−6	5	1
5	506	184	−6	5	23	−19	3
6	184	138	23	−19	−52	43	2
7	138	46	−52	43	75	−62	1
8	46	0	75	−62	−277	229	3

Die Ausgabe ist $(46, 75, -62)$. Man überprüft leicht, dass $46 = \mathrm{ggT}(10534, 12742) = 75 \cdot 10534 - 62 \cdot 12742$ gilt.

Fakt 5.9

(a) Wenn Algorithmus 5.2 auf Eingabe (x, y) die Ausgabe (d, s, t) liefert, dann gilt $d = \mathrm{ggT}(x, y) = sx + ty$.

(b) Die Anzahl der Schleifendurchläufe ist dieselbe wie beim gewöhnlichen Euklidischen Algorithmus.

(c) Die Anzahl von Ziffernoperationen für Algorithmus 5.2 ist $O(\log(x)\log(y))$.

Wir notieren noch eine wichtige Folgerung aus dem Lemma von Bezout.[2]

[2]In der Schule begründet man diese Tatsache mit Hilfe der Primzahlzerlegung. Diese ist aber gar nicht nötig.

Fakt 5.10
Wenn x und y teilerfremd sind und a sowohl durch x als auch durch y teilbar ist, dann ist a auch durch xy teilbar.

Beweis: Weil x und y Teiler von a sind, kann man $a = ux$ und $a = vy$ schreiben, für ganze Zahlen u, v. Weil x und y teilerfremd sind, folgt aus Lemma 5.8, dass man $1 = \text{ggT}(x, y) = sx+ty$ schreiben kann, für ganze Zahlen s, t. Dann ist $a = asx+aty = vysx + uxty = (vs + ut)xy$, also ist xy Teiler von a. □

5.1.2 Eigenschaften der Multiplikation in $\mathbb{Z}_m$

In jeder Struktur $\mathbb{Z}_m$ spielen die Elemente, die ein *multiplikatives Inverses* haben, eine spezielle Rolle; sie erhalten eine eigene Bezeichnung.

Lemma 5.11
Für jedes $m \geq 2$ und jedes $a \in \mathbb{Z}$ gilt:
Es gibt ein b mit $ab \bmod m = 1$ genau dann wenn $\text{ggT}(a, m) = 1$.

Wegen der Homomorphieeigenschaft (Lemma 5.3) kommt es für die Frage, ob $ab \bmod m = 1$ gilt, nur auf $a \bmod m$ und $b \bmod m$ an. Wir können unsere Überlegungen also auf Elemente von $\mathbb{Z}_m$ beschränken. Wenn $a, b \in \mathbb{Z}_m$ und $ab \bmod m = 1$, nennen wir b ein *multiplikatives Inverses* zu a modulo m.

Beispiel: Aus $6 \cdot 21 + (-5) \cdot 25 = 1$ folgt, dass 6 ein multiplikatives Inverses zu 21 modulo 25 ist.

Definition 5.12
Für $m \geq 2$ sei $\mathbb{Z}_m^* := \{a \in \mathbb{Z}_m \mid \text{ggT}(a, m) = 1\}$.

Fakt 5.13
Für jedes $m \geq 2$ gilt:
$\mathbb{Z}_m^*$ mit der Multiplikation modulo m als Operation ist eine Gruppe.

(Das folgt direkt aus Lemma 5.11.)

Beispiel: $\mathbb{Z}_{21}^* = \{1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20\}$ und $\mathbb{Z}_7^* = \{1, 2, 3, 4, 5, 6\}$.

Am schönsten ist die Situation, wenn alle Zahlen aus Z_m außer der 0 in $\mathbb{Z}_m^*$ liegen. Dies ist genau dann der Fall, wenn m Primzahl ist.

Definition 5.14 (a) Eine Zahl $p \geq 1$ heißt **Primzahl**, wenn p genau zwei positive Teiler hat.[a] (Diese Teiler sind dann 1 und p.)

(b) Eine Zahl $x \geq 1$ heißt **zusammengesetzt**, wenn sie einen Teiler y mit $1 < y < x$ besitzt.[b]

[a] Die Zahl 1 hat nur einen positiven Teiler, nämlich 1. Also ist 1 keine Primzahl.
[b] Die Zahl 1 besitzt keinen Teiler y mit $1 < y < 1$. Also ist 1 auch nicht zusammengesetzt.

Proposition 5.15
Für jedes $m \geq 2$ gilt:
m ist Primzahl $\Leftrightarrow$ $\mathbb{Z}_m^* = \{1, \ldots, m-1\}$ $\Leftrightarrow$ $\mathbb{Z}_m$ ist ein Körper.

Die zweite Äquivalenz ist dabei klar: Der Ring $\mathbb{Z}_m$ ist nach Definition ein Körper genau dann wenn jedes Element von $\mathbb{Z}_m - \{0\}$ ein multiplikatives Inverses besitzt.

Beispiel: Wir rechnen modulo 13 und geben für jedes $a \in \mathbb{Z}_{13}^*$ das Inverse b sowie das Produkt $a \cdot b$ an (das natürlich bei der Division durch 13 Rest 1 lassen muss).

a	1	2	3	4	5	6	7	8	9	10	11	12
b	1	7	9	10	8	11	2	5	3	4	6	12
$a \cdot b$	1	14	27	40	40	66	14	40	27	40	66	144

14 ist keine Primzahl, und es gibt keine Zahl b mit $2 \cdot b \bmod 14 = 1$; das heißt, dass $2 \notin \mathbb{Z}_{14}^*$ und daher, dass $\mathbb{Z}_{14}$ kein Körper ist.

Basis für randomisierte Primzahltests ist das folgende grundlegende Ergebnis.

Fakt 5.16 ***Kleiner Satz von Fermat***
Wenn p eine Primzahl ist, dann gilt:

$$a^{p-1} \bmod p = 1, \text{ für jedes } a \in \mathbb{Z}_p^*.$$

Beweis: Betrachte die Abbildung

$$g_a \colon \mathbb{Z}_p^* \ni x \mapsto ax \bmod p \in \mathbb{Z}_p^*.$$

Diese Abbildung ist injektiv, da für das zu a inverse Element b gilt: $b \cdot g_a(x) \bmod p = bax \bmod p = x$. Also:

$$\{1, \ldots, p-1\} = \{g_a(1), \ldots, g_a(p-1)\}.$$

Wir multiplizieren:

$$1 \cdot \ldots \cdot (p-1) \bmod p = g_a(1) \cdot \ldots \cdot g_a(p-1) \bmod p = a^{p-1} \cdot (1 \cdot \ldots \cdot (p-1)) \bmod p.$$

Wenn wir beide Seiten mit dem multiplikativen Inversen von $1 \cdot \ldots \cdot (p-1) \bmod p$ multiplizieren, erhalten wir $1 = a^{p-1} \bmod p$. □

Wir bemerken, dass auch die Umkehrung des kleinen Satzes von Fermat gilt. Wenn $a \in (\mathbb{Z}_m - \{0\}) - \mathbb{Z}_m^*$, also $\mathrm{ggT}(a,m) \neq 1$ gilt, dann folgt $a^{m-1} \bmod m \neq 1$. (Aus $a^{m-1} \bmod m = 1$ folgt $a^{m-1} - qm = 1$ für ein q, also $\mathrm{ggT}(a,m) = 1$.)

Ab hier soll p immer eine Primzahl bezeichnen. Normalerweise ist dabei $p > 2$, also p ungerade.[3]

Oft kommt es darauf an, Potenzen $x^n \bmod m$ zu berechnen. Dabei kann n eine sehr große Zahl sein. *Beispiel* für eine solche Aufgabe: Berechne

$$3^{1384788374932954500363985493554603584759389} \bmod 28374618732464817362847326847331872341234.$$

Es gibt eine unmittelbar einleuchtende rekursive Formel, die den Weg zu einer effizienten Berechnung weist:

$$x^n \bmod m = \begin{cases} 1 & \text{, wenn } n = 0, \\ x \bmod m & \text{, wenn } n = 1, \\ ((x^2 \bmod m)^{n/2}) \bmod m & \text{, wenn } n \geq 2 \text{ gerade ist,} \\ ((x^2 \bmod m)^{(n-1)/2} \bmod m) \cdot x \bmod m & \text{, wenn } n \geq 2 \text{ ungerade ist.} \end{cases}$$

Diese Formeln führen unmittelbar zu folgender rekursiver Prozedur.

Algorithmus 5.3 ***Schnelle modulare Exponentiation, rekursiv***
function modexp(x, n, m) // berechnet $y = x^n \bmod m$
EINGABE: Ganze Zahlen x, $n \geq 0$, and $m \geq 2$, mit $0 \leq x < m$.
METHODE:

```
0   if n = 0 then return 1;
1   if n = 1 then return x;
2   y ← modexp(x * x, ⌊n/2⌋, m);  // rekursiver Aufruf
3   if n ist ungerade then y ← y * x mod m
4   return y.
```

[3] „All primes are odd except 2, which is the oddest of all." (D. E. Knuth) — Ein Wortspiel damit, dass „odd" sowohl „ungerade" als auch „merkwürdig" bedeuten kann.

Um $x^n \bmod m$ für beliebige x zu berechnen, ruft man modexp$(x \bmod m, n, m)$ auf. Man erkennt sofort, dass in jeder Rekursionsebene die Bitanzahl des Exponenten um 1 sinkt, dass also die Anzahl der Rekursionsebenen etwa $\log n$ beträgt. In jeder Rekursionsstufe ist eine oder sind zwei Multiplikationen modulo m auszuführen, was $O((\log m)^2)$ Ziffernoperationen erfordert (Schulmethode). Damit kommt man bei der schnellen Exponentiation selbst bei der einfachsten Implementierung der Multiplikation mit $O((\log n)(\log m)^2)$ Ziffernoperationen zum Ergebnis.

Lemma 5.17
Die Berechnung von $a^n \bmod m$ benötigt $O(\log n)$ Multiplikationen und Divisionen modulo m von Zahlen aus $\{0, \ldots, m^2 - 1\}$ sowie $O((\log n)(\log m)^2)$ Bitoperationen.

Bemerkung: Man kann denselben Algorithmus in einem beliebigen *Monoid* (Bereiche mit einer assoziativen Operation und einem neutralen Element) benutzen. Einen Monoid bilden zum Beispiel die natürlichen Zahlen mit der Multiplikation oder die Menge Σ^* über einem Alphabet Σ mit der Konkatenation. In Abschnitt 5.4 benutzen wir die schnelle Exponentiation für einen etwas ungewöhnlichen Monoid.

5.1.3 Der Chinesische Restsatz und die Eulersche φ-Funktion

Der „Chinesische Restsatz“ besagt im Wesentlichen, dass für teilerfremde Zahlen m und n die Strukturen $\mathbb{Z}_m \times \mathbb{Z}_n$ (mit komponentenweisen Operationen) und $\mathbb{Z}_{mn}$ isomorph sind.

Wir beginnen mit einem Beispiel, nämlich $m = 3$, $n = 8$, also $mn = 24$. Die folgende Tabelle gibt die Reste der Zahlen $x \in \{0, 1, \ldots, 23\}$ modulo 3 und modulo 8 an. Die Restepaare wiederholen sich zyklisch für andere $x \in \mathbb{Z}$.

x	0	1	2	3	4	5	6	7	8	9	10	11
$x \bmod 3$	0	1	2	0	1	2	0	1	2	0	1	2
$x \bmod 8$	0	1	2	3	4	5	6	7	0	1	2	3

x	12	13	14	15	16	17	18	19	20	21	22	23
$x \bmod 3$	0	1	2	0	1	2	0	1	2	0	1	2
$x \bmod 8$	4	5	6	7	0	1	2	3	4	5	6	7

Wenn wir die Einträge in Zeilen 2 und 3 als 24 Paare in $\mathbb{Z}_3 \times \mathbb{Z}_8$ ansehen, erkennen wir, dass sie alle verschieden sind, also auch alle Möglichkeiten in $\{0,1,2\} \times \{0,1,\ldots,7\}$ abdecken. D. h.: Die Abbildung $x \mapsto (x \bmod 3, x \bmod 8)$ ist eine Bijektion zwischen $\mathbb{Z}_{24}$ und $\mathbb{Z}_3 \times \mathbb{Z}_8$. Zudem spiegeln sich arithmetische Operationen auf den Elementen von $\mathbb{Z}_{24}$ in den Resten modulo 3 und 8 wider. Beispielsweise liefert die Addition von $(2,7)$ und $(2,1)$ das Resultat $(1,0)$, das der Addition von 23 und 17 mit dem Resultat $40 = 16 \bmod 24$ entspricht. Genauso ist

$$(2^5 \bmod 3, 3^5 \bmod 8) = (2,3),$$

was der Beobachtung $11^5 \bmod 24 = 11$ entspricht.

Der Chinesische Restsatzsagt im wesentlichen, dass eine solche strukturelle Entsprechung zwischen den Resten modulo mn und Paaren von Resten modulo m bzw. n immer gilt, wenn m und n teilerfremd sind.

Fakt 5.18 ***Chinesischer Restsatz***
m und n seien teilerfremd. Dann ist die Abbildung[a]

$$\Phi\colon \mathbb{Z}_{mn} \ni x \mapsto (x \bmod m, x \bmod n) \in \mathbb{Z}_m \times \mathbb{Z}_n$$

bijektiv. Weiterhin: Wenn $\Phi(x) = (x_1, x_2)$ und $\Phi(y) = (y_1, y_2)$, dann gilt:

(a) $\Phi(x +_{mn} y) = (x_1 +_m y_1, x_2 +_n y_2)$;

(b) $\Phi(x \cdot_{mn} y) = (x_1 \cdot_m y_1, x_2 \cdot_n y_2)$;

(c) $\Phi(1) = (1,1)$.

(Dabei bezeichnen $+_j$ und $\cdot_j$ die Addition und die Multiplikation modulo j.)

[a] Φ ist der griechische Buchstabe *Phi*.

Für mathematisch-strukturell orientierte Leser/innen: Die Gleichungen (a) und (b) kann man etwas abstrakter auch so fassen, dass die Abbildung Φ ein Ring-mit-1-Isomorphismus zwischen $\mathbb{Z}_{mn}$ und $\mathbb{Z}_m \times \mathbb{Z}_n$ ist.

Wir wollen jetzt noch untersuchen, wie sich Zahlen, die zu m und n teilerfremd sind, in der Sichtweise des Chinesischen Restsatzes verhalten.

Proposition 5.19
Wenn man die Abbildung Φ aus dem Chinesischen Restsatz auf $\mathbb{Z}_{mn}^*$ einschränkt, ergibt sich eine Bijektion zwischen $\mathbb{Z}_{mn}^*$ und $\mathbb{Z}_m^* \times \mathbb{Z}_n^*$.

Der Chinesische Restsatz und die nachfolgenden Bemerkungen und Behauptungen lassen sich leicht auf $r > 2$ paarweise teilerfremde Faktoren $n_1, \ldots, n_r$ verallgemeinern. Die Aussagen lassen sich durch vollständige Induktion über r beweisen. Prop. 5.19 liefert auch eine Formel für die Kardinalitäten der Mengen $\mathbb{Z}_m^*$ für beliebige $m \geq 2$.[4]

Definition 5.20 ***Eulersche φ-Funktion***
Für $m \geq 2$ sei

$$\varphi(m) := |\mathbb{Z}_m^*| = |\{x \mid 0 \leq x < m, \mathrm{ggT}(x, m) = 1\}|.$$

Einige Beispielwerte sind in Tab. 2 angegeben.

m	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\varphi(m)$	1	2	2	4	2	6	4	6	4	10	4	12	6	8	8

Tabelle 2: Eulersche φ-Funktion für kleine m

Folgendes ist eine unmittelbare Konsequenz aus Proposition 5.19:

Lemma 5.21
Für teilerfremde Zahlen n und m gilt $\varphi(mn) = \varphi(m) \cdot \varphi(n)$.

Uns interessiert (später) besonders der Fall $\varphi(p) = p - 1$ für Primzahlen p und $\varphi(pq) = \varphi(p) \cdot \varphi(q) = (p-1)(q-1)$ für verschiedene Primzahlen p und q. Man kann aber auch eine allgemeine Formel für $\varphi(n)$ gewinnen.

Lemma 5.22
Für $m \geq 2$ gilt

$$\varphi(m) = m \cdot \prod_{\substack{p \text{ prim} \\ p \mid m}} \left(1 - \frac{1}{p}\right).$$

Beweis: Wenn m eine Primzahlpotenz p^t ist, dann besteht $\mathbb{Z}_m^*$ aus den Zahlen in $\mathbb{Z}_m = \{0, 1, \ldots, p^t - 1\}$, die nicht durch p teilbar sind. Da es in $\mathbb{Z}_m$ insgesamt p^t Zahlen gibt und p^{t-1} Vielfache von p, gilt $\varphi(m) = p^t - p^{t-1} = m - m/p = m(1 - 1/p)$. Nun nehmen wir an, dass

$$m = p_1^{t_1} \cdots p_s^{t_s}$$

[4] φ ist der griechische Buchstabe *phi*, gesprochen *fi*.

gilt, für verschiedene Primzahlen $p_1, \ldots, p_s$ und $t_1, \ldots, t_s \geq 1$. Die Faktoren $p_1^{t_1}, \ldots, p_s^{t_s}$ sind teilerfremd. Mit Lemma 5.21, $(s-1)$-mal angewendet, erhalten wir

$$\varphi(m) = \prod_{i=1}^{s} \varphi(p_i^{t_i}) = \prod_{i=1}^{s} p_i^{t_i}(1 - 1/p_i) = m \cdot \prod_{i=1}^{s} \left(1 - \frac{1}{p_i}\right). \qquad \square$$

Mit dieser Formel lassen sich die Werte in Tabelle 2 schnell verifizieren. (*Beispiel*: $\varphi(12) = 12(1-1/2)(1-1/3) = 12 \cdot (1/2) \cdot (2/3) = 4$.)

Bemerkung: Die simple Formel in Lemma 5.22 könnte zu dem Schluss verleiten, dass sich $\varphi(m)$ zu gegebenem m immer leicht berechnen lässt. Aber Achtung: Man muss dazu die Menge der Primfaktoren von m kennen. Dies läuft darauf hinaus, das Faktorisierungsproblem für m zu lösen, und hierfür kennt man keine effizienten Algorithmen. Tatsächlich ist auch kein effizienter Algorithmus bekannt, der es erlaubt, $\varphi(m)$ aus m zu berechnen.

5.1.4 Primzahlen

Wir erinnern an Definition 5.14.

Fakt 5.23
Wenn p eine Primzahl ist und $p \mid xy$ gilt, dann gilt $p \mid x$ oder $p \mid y$.

Beweis: Wenn $p \mid x$, sind wir fertig. Also können wir $p \nmid x$ annehmen. Das heißt, dass $\mathrm{ggT}(p, x) = 1$ ist. Nach dem Lemma von Bezout können wir $1 = sp + tx$ schreiben, für ganze Zahlen s, t. Daraus folgt: $y = spy + txy$. Nun ist xy durch p teilbar, also auch $y = spy + txy$. $\square$

Satz 5.24 ***Fundamentalsatz der Zahlentheorie***
Jede Zahl $x \geq 1$ kann als Produkt von Primzahlen geschrieben werden. Die Faktoren sind dabei bis auf die Reihenfolge eindeutig bestimmt.

Fakt 5.25
Jede zusammengesetzte Zahl x besitzt einen Primfaktor p mit $p \leq \sqrt{x}$.

Bemerkung: Wir betrachten das resultierende naive Faktorisierungsverfahren: Teste die Zahlen in $\{2, \ldots, \lfloor\sqrt{x}\rfloor\}$ nacheinander darauf, ob sie x teilen; wenn ein Faktor p gefunden wurde, wende dasselbe Verfahren auf $x' = x/p$ an. Dieses Verfahren hat im

schlechtesten Fall Rechenzeit mindestens $\Theta(\sqrt{x}) = \Theta(2^{(\log x)/2})$, also exponentiell in der Bitlänge von x. Wie wir später genauer diskutieren werden, sind für das Auffinden der Primzahlzerlegung einer gegebenen Zahl x überhaupt keine effizienten Algorithmen bekannt (also Algorithmen mit Laufzeiten $O((\log x)^c)$ für konstantes c). Aber es gibt effiziente Algorithmen, mit denen man feststellen kann, ob eine Zahl x eine Primzahl ist oder nicht. Dieser Unterschied in der Schwierigkeit des Faktorisierungsproblems und des Primzahlproblems liegt einer ganzen Reihe von kryptographischen Verfahren zugrunde.

Satz 5.26 ***Euklid***
Es gibt unendlich viele Primzahlen.

Über die Verteilung der Primzahlen (ihre „Dichte“) in $\mathbb{N}$ gibt der berühmte Primzahlsatz Auskunft.[5] Mit $\pi(x)$ bezeichnen wir die Anzahl der Primzahlen, die nicht größer als x sind.

Satz 5.27 ***Primzahlsatz***

$$\lim_{x\to\infty} \frac{\pi(x)}{x/\ln x} = 1.$$

Das heißt, dass für große x in $(x, 2x]$ etwa $\frac{2x}{\ln(2x)} - \frac{x}{\ln x} \approx \frac{x}{\ln x}$ Primzahlen zu erwarten sind. Die μ-ziffrigen Zahlen bilden das Intervall $[2^{\mu-1}, 2^{\mu})$. Der Anteil der Primzahlen in diesem Intervall ist näherungsweise

$$\frac{2^{\mu-1}/\ln(2^{\mu-1})}{2^{\mu-1}} \approx \frac{1}{(\ln 2)(\mu-1)} \approx 1{,}44/\mu.$$

Für $\mu \approx 2000$ ist der relative Anteil von Primzahlen im interessanten Zahlenbereich also etwa $\approx 1{,}44/2000 \approx 1/1400$. Er sinkt umgekehrt proportional zur Ziffernzahl.

Für unsere Zwecke etwas angenehmer zu benutzen als der Primzahlsatz ist folgende etwas schwächere Aussage, die eine konkrete Konstante angibt:

Satz 5.28 ***Ungleichung von Finsler***
Für jede ganze Zahl $m \geq 2$ liegen im Intervall $(m, 2m]$ mindestens $m/(3\ln(2m))$ Primzahlen:

$$\pi(2m) - \pi(m) \geq \frac{m}{3\ln(2m)}.$$

[5] 1793 von Gauß und unabhängig 1798 von Legendre vermutet; 1896 unabhängig von Hadamard und de La Vallé Poussin bewiesen.

Ein vollständiger, vergleichsweise einfacher *Beweis* für eine Aussage dieser Art findet sich zum Beispiel in dem Lehrbuch „Elemente der Diskreten Mathematik: Zahlen und Zählen, Graphen und Verbände“ von Diekert, Kufleitner, Rosenberger (De Gruyter 2013).

Beispiele:

Ziffernzahl μ	m	Finsler-Schranke für $(\pi(2m) - \pi(m))/m$	numerische untere Schranke
256	2^{255}	$\frac{1}{3 \cdot 256 \cdot \ln 2}$	$\frac{1}{533}$
512	2^{511}	$\frac{1}{3 \cdot 512 \cdot \ln 2}$	$\frac{1}{1065}$
1024	2^{1023}	$\frac{1}{3 \cdot 1024 \cdot \ln 2}$	$\frac{1}{2130}$
2048	2^{2047}	$\frac{1}{3 \cdot 2048 \cdot \ln 2}$	$\frac{1}{4260}$

Eine Verdopplung der Ziffernzahl halbiert die untere Schranke für die Primzahldichte.

5.2 Der Miller-Rabin-Primzahltest

In diesem Abschnitt lernen wir einen randomisierten Algorithmus kennen, der es erlaubt, zu einer gegebenen Zahl n zu entscheiden, ob n eine Primzahl ist oder nicht.

Ein idealer Primzahltest sieht so aus:

Eingabe: Eine natürliche Zahl $n \geq 3$.

Ausgabe: 0, falls n eine Primzahl ist; 1, falls n zusammengesetzt ist.

Wozu braucht man Primzahltests? Zunächst ist die Frage „Ist n eine Primzahl?“ eine grundlegende mathematisch interessante Fragestellung. Spätestens mit dem Siegeszug des RSA-Kryptosystems ab den späten 1970er Jahren hat sich die Situation jedoch dahin entwickelt, dass man Algorithmen benötigt, die immer wieder neue vielziffrige Primzahlen (etwa mit 512 oder 1024 Bits[6] bzw. 155 oder 310 Dezimalziffern) bereitstellen können. Den Kern dieser Primzahlerzeugungs-Verfahren (siehe Abschnitt 5.3) bildet ein Verfahren, das eine gegebene Zahl n darauf testet, ob sie prim ist.

[6] `http://www.rsa.com/rsalabs/node.asp?id=2218` am 29.12.2012: „RSA Laboratories currently recommends key sizes of 1024 bits for corporate use and 2048 bits for extremely valuable keys like the root key pair used by a certifying authority (see Question 4.1.3.12). Several recent standards specify a 1024-bit minimum for corporate use.“ Dabei bedeutet *key size* die Bitanzahl eines Produkts $n = p \cdot q$ für Primzahlen p und q mit jeweils der halben Länge.

Ein naiver Primzahltest („versuchsweise Division“), der dem *brute-force*-Paradigma folgt, findet durch direkte Division der Zahl n durch $2, 3, 4, \ldots, \lfloor\sqrt{n}\rfloor$ heraus, ob n einen nichttrivialen Teiler hat. Man kann dieses Verfahren durch einige Tricks beschleunigen, aber die Rechenzeit wächst dennoch mit $\Theta(\sqrt{n})$. Dies macht es für Zahlen mit mehr als 40 Dezimalstellen praktisch undurchführbar, von Zahlen mit mehr als 100 Dezimalstellen ganz zu schweigen.

In diesem Abschnitt beschreiben wir den randomisierten Primzahltest von Miller und Rabin. Dabei handelt es sich um einen Monte-Carlo-Algorithmus mit einseitigem Fehler. Das heißt: Auf Eingaben n, die Primzahlen sind, wird immer 0 ausgegeben; auf Eingaben n, die zusammengesetzt sind, gibt es eine gewisse (von n abhängige) Wahrscheinlichkeit, dass die Ausgabe 0, also falsch ist. Für kein zusammengesetztes n ist diese Wahrscheinlichkeit größer als die „Fehlerschranke“ $\frac{1}{4}$. Wir beweisen nur die Fehlerschranke $\frac{1}{2}$. Im Beweis benutzen wir einfache zahlentheoretische Überlegungen. Eine herausragende Eigenschaft des Miller-Rabin-Tests ist seine Effizienz. Wir werden sehen, dass selbst bei Verwendung der Schulmethoden für Multiplikation und Division die Bitkomplexität des Primzahltests nur $O((\log n)^3)$ ist.

Bemerkung: Der Miller-Rabin-Algorithmus stammt aus dem Jahr 1977; er folgte einem kurz vorher vorgestellten anderen randomisierten Primzahltest (Solovay-Strassen-Test). Für diesen und andere randomisierte Primzahltests (z. B. der „Strong Lucas Probable Prime Test“ oder der „Quadratic Frobenius Test“ von Grantham) sei auf die Literatur verwiesen. Im Jahr 2002 stellten Agarwal, Kayal und Saxena einen deterministischen Primzahltest mit polynomieller Rechenzeit vor. Die Rechenzeit ist z. B. durch $O((\log n)^{7{,}5})$ beschränkt. Dieser Algorithmus stellte insofern einen gewaltigen Durchbruch dar, als er ein Jahrhunderte altes offenes Problem löste, nämlich die Frage nach einem effizienten *deterministischen* Verfahren für das Entscheidungsproblem „ist n Primzahl oder zusammengesetzt“? Andererseits ist seine Laufzeit im Vergleich etwa zu dem hier diskutierten randomisierten Verfahren so hoch, dass nach wie vor die randomisierten Algorithmen benutzt werden, um für kryptographische Anwendungen Primzahlen zu erzeugen.

Da gerade Zahlen leicht zu erkennen sind, beschränken wir im Folgenden unsere Überlegungen auf ungerade Zahlen $n \geq 3$.

5.2.1 Der Fermat-Test

Wir erinnern uns an den kleinen Satz von Fermat (Fakt 5.16):

$$n \text{ ist Primzahl und } 1 \leq a < n \quad \Rightarrow \quad a^{n-1} \bmod n = 1.$$

Wir können diese Aussage dazu benutzen, um „Belege“ oder „Zertifikate“ oder „Zeugen“ dafür anzugeben, dass eine Zahl n zusammengesetzt ist: Wenn wir eine Zahl a mit $1 \leq a < n$ finden, für die $a^{n-1} \bmod n \neq 1$ gilt, dann ist n definitiv keine Primzahl.

Definition 5.29
Sei n eine ungerade Zahl. Eine Zahl a mit $1 \leq a < n-1$ heißt ein **F-Zeuge** für n, wenn $a^{n-1} \bmod n \neq 1$ gilt. Wenn n zusammengesetzt ist und für a mit $1 \leq a < n-1$ die Gleichheit $a^{n-1} \bmod n = 1$ gilt, heißt a ein **F-Lügner** für n.

Wir bemerken, dass ein F-Zeuge belegt, dass es Faktoren $k, \ell > 1$ mit $n = k \cdot \ell$ *gibt*, dass aber solche Faktoren nicht angegeben oder gefunden werden müssen. Dies wird von Primzahltests auch nicht verlangt und normalerweise auch nicht geleistet.

Man sieht sofort, dass 1 und $n-1$ immer F-Lügner sind: Es gilt $1^{n-1} \mod n = 1$ und $(n-1)^{n-1} \equiv (-1)^{n-1} \equiv 1 \pmod{n}$, weil $n-1$ gerade ist.

Für jede zusammengesetzte Zahl n gibt es mindestens einen F-Zeugen. Wir erinnern uns, dass n genau dann zusammengesetzt ist, wenn $\{1, \ldots, n-1\} - \mathbb{Z}_n^* \neq \emptyset$ gilt.

Lemma 5.30
Wenn n keine Primzahl ist, ist jedes $a \in \{1, \ldots, n-1\} - \mathbb{Z}_n^*$ ein F-Zeuge.

Beweis: Sei $d = \mathrm{ggT}(a, n)$. Dann ist auch a^{n-1} durch d teilbar, also $\mathrm{ggT}(a^{n-1} \bmod n, n) = \mathrm{ggT}(a^{n-1}, n) \geq d > 1$. Daher ist $a^{n-1} \bmod n \neq 1$. □

Leider ist für manche zusammengesetzten Zahlen n die Menge $\{1, \ldots, n-1\} - \mathbb{Z}_n^*$ äußerst dünn. Wenn zum Beispiel $n = pq$ für zwei Primzahlen p und q ist, dann gilt $\mathrm{ggT}(a, n) > 1$ genau dann wenn p oder q ein Teiler von a ist. Es gibt genau $p+q-2$ solche Zahlen a in $\{1, \ldots, n-1\}$, was gegenüber n sehr klein ist, wenn p und q annähernd gleich groß sind. Um eine gute Chance zu haben, F-Zeugen zu finden, sollte es also mehr als nur die in $\{1, \ldots, n-1\} - \mathbb{Z}_n^*$ geben.

Beispiel: $n = 91 = 7 \cdot 13$. Tabelle 3 zeigt, dass es 18 Vielfache von 7 und 13 gibt (für größere p und q wird der Anteil dieser offensichtlichen F-Zeugen noch kleiner sein), und daneben weitere 36 F-Zeugen und 36 F-Lügner in $\{1, 2, \ldots, 90\}$. In diesem Beispiel gibt es um einiges mehr F-Zeugen als F-Lügner. Wenn dies für alle zusammengesetzten Zahlen n der Fall wäre, wäre es eine elegante randomisierte Strategie, einfach zufällig nach F-Zeugen zu suchen.

Dies führt zu unserem ersten Versuch für einen randomisierten Primzahltest.

Algorithmus 5.4 ***Fermat-Test***

Vielfache von 7	7, 14, 21, 28, 35, 42, 49, 56, 63, 70, 77, 84
Vielfache von 13	13, 26, 39, 52, 65, 78
F-Zeugen in $\mathbb{Z}_{91}^*$	2, 5, 6, 8, 11, 15, 18, 19, 20, 24, 31, 32, 33, 34, 37, 41, 44, 45, 46, 47, 50, 54, 57, 58, 59, 60, 67, 71, 72, 73, 76, 80, 83, 85, 86, 89
F-Lügner	1, 3, 4, 9, 10, 12, 16, 17, 22, 23, 25, 27, 29, 30, 36, 38, 40, 43, 48, 51, 53, 55, 61, 62, 64, 66, 68, 69, 74, 75, 79, 81, 82, 87, 88, 90

Tabelle 3: F-Zeugen und F-Lügner für $n = 91 = 7 \cdot 13$

EINGABE: Ungerade Zahl $n \geq 5$.
METHODE:
1 Wähle a zufällig aus $\{1, \ldots, n-1\}$;
2 **if** $a^{n-1} \bmod n \neq 1$ **then return** 1 **else return** 0.

Die Laufzeitanalyse liegt auf der Hand: Der teuerste Teil ist die Berechnung der Potenz $a^{n-1} \bmod n$ durch schnelle Exponentiation, die nach den Ergebnissen von Lemma 5.17 $O(\log n)$ arithmetische Operationen und $O((\log n)^3)$ Bitoperationen benötigt. Weiter ist es klar, dass der Algorithmus einen F-Zeugen gefunden hat, wenn er „1“ ausgibt, dass in diesem Fall also n zusammengesetzt sein muss. Umgekehrt ausgedrückt: Wenn n eine Primzahl ist, gibt der Fermat-Test garantiert „0“ aus.

Für $n = 91$ wird das falsche Ergebnis 0 ausgegeben, wenn als a einer der 34 F-Lügner außer 1 und 90 gewählt wird. Die Wahrscheinlichkeit hierfür ist $\frac{34}{88} = \frac{17}{44} \approx 0{,}3864$.

Für viele zusammengesetzte Zahlen n gibt es reichlich F-Zeugen, so dass der Fermat-Test für diese n mit konstanter Wahrscheinlichkeit das korrekte Ergebnis liefert. Wir analysieren das Verhalten des Fermat-Tests für solche „gutmütigen“ Eingabezahlen n (für die $n = 91$ ein typisches Beispiel ist).

Satz 5.31
Sei $n \geq 9$ eine ungerade zusammengesetzte Zahl. Wenn es mindestens einen F-Zeugen $b \in \mathbb{Z}_n^*$ gibt, dann liefert der Fermat-Test auf Eingabe n mit Wahrscheinlichkeit größer als $\frac{1}{2}$ die korrekte Antwort „1“.

Beweis: Sei $b \in \mathbb{Z}_n^*$ ein F-Zeuge. Betrachte die Funktion $g_b\colon L_n^{\mathrm{F}} \to \mathbb{Z}_n^*$, die den F-Lügner a auf $g_b(a) = ba \bmod n$ abbildet. Wie im Beweis von Fakt 5.16 sieht man, dass g_b injektiv ist. Weiter ist $g_b(a)$ für jedes $a \in L_n^{\mathrm{F}}$ ein F-Zeuge:

$$(ba \bmod n)^{n-1} \bmod n = (b^{n-1} \bmod n)(\underbrace{(a^{n-1}) \bmod n}_{=1}) = b^{n-1} \bmod n \neq 1.$$

Wir können also jedem F-Lügner a einen „Zwilling“ $g_b(a)$ in der Menge der F-Zeugen zuordnen. Daraus folgt, dass es in $\mathbb{Z}_n^*$ mindestens so viele F-Zeugen wie F-Lügner gibt. Mit Lemma 5.30 ergibt sich, dass $\{1, \ldots, n-1\}$ mehr F-Zeugen als F-Lügner enthält. Daher ist die Wahrscheinlichkeit, dass die im Fermat-Test gewählte Zahl a ein F-Lügner ist, kleiner als $\frac{1}{2}$. □

Die Fehlerwahrscheinlichkeit $\frac{1}{2}$ ist natürlich viel zu groß. Wir verringern sie durch wiederholte Ausführung des Fermat-Tests.

Algorithmus 5.5 ***Iterierter Fermat-Test***
EINGABE: Ungerade Zahl $n \geq 5$, eine Zahl $\ell \geq 1$.
METHODE:

```
1    repeat ℓ times
2        a ← ein zufälliges Element von {1, ..., n − 1};
3        if a^{n−1} mod n ≠ 1 then return 1;
4    return 0.
```

Wenn die Ausgabe 1 ist, hat der Algorithmus einen F-Zeugen für n gefunden, also ist n zusammengesetzt. D. h.: Wenn n eine Primzahl ist, ist die Ausgabe 0. Andererseits: Wenn n zusammengesetzt ist, und es mindestens einen F-Zeugen $a \in \mathbb{Z}_n^*$ gibt, dann ist nach Satz 5.31 die Wahrscheinlichkeit für die falsche Ausgabe „0“ höchstens $\left(\frac{1}{2}\right)^{\ell} = 2^{-\ell}$. Indem wir ℓ genügend groß wählen, können wir die Fehlerwahrscheinlichkeit so klein wie gewünscht einstellen.

Wenn es darum geht, aus einem genügend großen Bereich zufällig gewählte Zahlen darauf zu testen, ob es sich um eine Primzahl handelt, dann ist der Fermat-Test eine sehr effiziente und zuverlässige Methode. Wir kommen darauf im folgenden Abschnitt nochmals zurück.

Wenn man allerdings über die Herkunft der zu testenden Zahl n keine Information hat und eventuell damit rechnen muss, dass jemand (ein „Gegenspieler“) absichtlich eine besonders schwierige Eingabe vorlegt, dann stößt der Fermat-Test an eine Grenze. Es gibt nämlich „widerspenstige“ zusammengesetzte Zahlen, denen man mit diesem Test nicht beikommen kann, weil alle Elemente von $\mathbb{Z}_n^*$ F-Lügner sind. Mit diesen befasst sich der folgende Abschnitt.

5.2.2 Carmichael-Zahlen

Definition 5.32
Eine ungerade zusammengesetzte Zahl n heißt eine **Carmichael-Zahl**, wenn für alle $a \in \mathbb{Z}_n^*$ die Gleichung $a^{n-1} \bmod n = 1$ gilt.

Die kleinste Carmichael-Zahl ist $561 = 3 \cdot 11 \cdot 17$. Weitere kleine Carmichael-Zahlen sind $1105 = 5 \cdot 13 \cdot 17$ und $1729 = 7 \cdot 13 \cdot 19$. Erst im Jahr 1994 wurde bewiesen, dass es unendlich viele Carmichael-Zahlen gibt, genauer: wenn x genügend groß ist, dann gibt es in $\{n \mid n \leq x\}$ mehr als $x^{2/7}$ Carmichael-Zahlen. Die aktuell beste bekannte untere Schranke ist $x^{1/3}$. Von Erdős (1956) stammt die obere Schranke $x \cdot \exp(\frac{-c \ln x \ln\ln\ln x}{\ln\ln x})$, für eine Konstante $c > 0$, die zeigt, dass Carmichael-Zahlen *viel* seltener als Primzahlen sind. (Mehr Details: `https://en.wikipedia.org/wiki/Carmichael_number` .)

Wenn wir dem Fermat-Test eine Carmichael-Zahl n als Eingabe geben, ist die Wahrscheinlichkeit für die falsche Antwort 0 nach Lemma 5.22 genau

$$\frac{\varphi(n)}{n-1} > \frac{\varphi(n)}{n} = \prod_{\substack{p \text{ prim} \\ p \text{ teilt } n}} \left(1 - \frac{1}{p}\right) > 1 - \sum_{\substack{p \text{ prim} \\ p \text{ teilt } n}} \frac{1}{p}.$$

Diese Wahrscheinlichkeit liegt nahe an 1, wenn n nur wenige und relativ große Primfaktoren hat. An solchen Carmichael-Zahlen besteht etwa im Bereich der Zahlen im Bereich $[10^{15}, 10^{16}]$ kein Mangel, wie ein Blick in entsprechende Tabellen zeigt. Zum Beispiel ist $n = 651693055693681 = 72931 \cdot 87517 \cdot 102103$ eine Carmichael-Zahl mit $\varphi(n)/n > 0.99996$.

Der Wiederholungstrick zur Wahrscheinlichkeitsverbesserung hilft hier leider auch nicht, denn wenn etwa p_0 der kleinste Primfaktor von n ist, und n nur 3 oder 4 Faktoren hat, dann sind $\Omega(p_0)$ Wiederholungen nötig, um die Fehlerwahrscheinlichkeit auf $\frac{1}{2}$ zu drücken. Sobald p_0 mehr als 30 Dezimalstellen hat, ist dies undurchführbar.

Für einen zuverlässigen, effizienten Primzahltest, der für *alle* Eingabezahlen funktioniert, müssen wir über den Fermat-Test hinausgehen. Interessanterweise ist dies praktisch ohne Effizienzverlust möglich.

Für spätere Benutzung stellen wir noch eine Hilfsaussage über Carmichael-Zahlen bereit.

Lemma 5.33
Wenn n eine Carmichael-Zahl ist, dann ist n keine Primzahlpotenz.

Beweis: Wir beweisen die Kontraposition: Wenn $n = p^\ell$ für eine ungerade Primzahl p und einen Exponenten $\ell \geq 2$ ist, dann ist n keine Carmichael-Zahl.

Dazu genügt es, eine Zahl $a \in \mathbb{Z}_n^*$ anzugeben, so dass $a^{n-1} \bmod n \neq 1$ ist. Wir definieren:

$$a := p^{\ell-1} + 1.$$

(Wenn z. B. $p = 7$ und $\ell = 3$ ist, ist $n = 343$ und $a = 49 + 1 = 50$.) Man sieht sofort, dass $a < p^\ell = n$ ist, und dass a nicht von p geteilt wird, also a und n teilerfremd sind; also ist $a \in \mathbb{Z}_n^*$. Nun rechnen wir modulo n, mit der binomischen Formel:

$$\begin{aligned} a^{n-1} &\equiv (p^{\ell-1} + 1)^{n-1} \\ &\equiv \sum_{0 \leq j \leq n-1} \binom{n-1}{j} (p^{\ell-1})^j \qquad (1) \\ &\equiv 1 + (p^\ell - 1) \cdot p^{\ell-1} \pmod{n}. \end{aligned}$$

(Die letzte Äquivalenz ergibt sich daraus, dass für $j \geq 2$ gilt, dass $(\ell - 1)j \geq \ell$ ist, also der Faktor $(p^{\ell-1})^j = p^{(\ell-1)j}$ durch $n = p^\ell$ teilbar ist, also modulo n wegfällt.) Nun ist $p^\ell - 1$ nicht durch p teilbar, also ist $(p^\ell - 1) \cdot p^{\ell-1}$ nicht durch $n = p^\ell$ teilbar. Damit folgt aus (1), dass $a^{n-1} \not\equiv 1 \pmod{n}$ ist, also $a^{n-1} \bmod n \neq 1$. □

Folgerung: Jede Carmichael-Zahl n lässt sich als $n = n_1 \cdot n_2$ schreiben, wo n_1 und n_2 teilerfremde ungerade Zahlen ≥ 3 sind.

(Ganz ähnlich kann man sogar zeigen, dass die Primfaktoren einer Carmichael-Zahl n alle verschieden sein müssen. Weiter haben Carmichael-Zahlen mindestens drei Primfaktoren. Schon aus dieser Tatsache kann man entnehmen, dass Carmichael-Zahlen eher selten sind.)

5.2.3 Nichttriviale Quadratwurzeln der 1

Lemma 5.34
Wenn p eine ungerade Primzahl ist, dann gilt $b^2 \bmod p = 1$, $b \in \mathbb{Z}_p$ genau für $b \in \{1, p-1\}$.

Beweis: Das Polynom $X^2 - 1$ hat Grad 2 und hat daher über dem Körper $\mathbb{Z}_p$ höchstens zwei Nullstellen. Offenbar sind 1 und $-1 \equiv p - 1$ zwei Nullstellen von $X^2 - 1$. □

Die im Lemma angegebene Eigenschaft lässt sich in ein weiteres Zertifikat für zusammengesetzte Zahlen ummünzen:

Wenn es ein $b \in \{2, \ldots, n-2\}$ gibt, das $b^2 \bmod n = 1$ erfüllt,
dann ist n zusammengesetzt.

Zahlen $b \in \{2, \ldots, n-2\}$ mit $b^2 \bmod n = 1$ heißen ***nichttriviale Quadratwurzeln der* 1 modulo *n***. Solche Zahlen gibt es nur, wenn n zusammengesetzt ist. Beispielsweise sind die Zahlen 1, 27, 64 und 90 genau die Quadratwurzeln der 1 modulo 91; davon sind 27 und $64 = 91 - 27$ nichttrivial. Wir beobachten: $27^2 \bmod 91 = 729 \bmod 91 = 1$. Beachte, dass $27 \equiv -1 \pmod 7$ und $27 \equiv 1 \pmod{13}$. Allgemeiner sieht man mit der Verallgemeinerung von Fakt 5.18 (Chinesischer Restsatz) auf r Faktoren leicht ein, dass es für ein Produkt $n = p_1 \cdots p_r$ aus verschiedenen ungeraden Primzahlen $p_1, \ldots, p_r$ genau 2^r Quadratwurzeln der 1 modulo n gibt, nämlich die Zahlen b, $0 \leq b < n$, die $b \bmod p_j \in \{1, p_j - 1\}$, $1 \leq j \leq r$, erfüllen. Wenn n nicht sehr viele verschiedene Primfaktoren hat, ist es also aussichtslos, einfach zufällig gewählte b's darauf zu testen, ob sie vielleicht nichttriviale Quadratwurzeln der 1 sind. Dennoch wird uns dieser Begriff bei der Formulierung eines effizienten Primzahltests helfen.

5.2.4 Der Miller-Rabin-Test

Wir kehren nochmals zum Fermat-Test zurück und sehen uns die dort durchgeführte Exponentiation $a^{n-1} \bmod n$ etwas genauer an. Die Zahl $n-1$ ist gerade, daher kann man sie als $n-1 = u \cdot 2^k$ schreiben, für eine ungerade Zahl u und ein $k \geq 1$. Dann gilt $a^{n-1} \equiv (a^u \bmod n)^{2^k} \bmod n$, und wir können $a^{n-1} \bmod n$ mit $k+1$ Zwischenschritten berechnen: Mit

$$\begin{aligned}
b_0 &= a^u \bmod n \\
b_1 &= b_0^2 \bmod n = a^{u \cdot 2} \bmod n, \\
b_2 &= b_1^2 \bmod n = a^{u \cdot 2^2} \bmod n, \\
&\vdots \\
b_i &= b_{i-1}^2 \bmod n = a^{u \cdot 2^i} \bmod n, \\
&\vdots \\
b_k &= b_{k-1}^2 \bmod n = a^{u \cdot 2^k} \bmod n
\end{aligned}$$

ist $b_k = a^{n-1} \bmod n$. Beispielsweise erhalten wir für $n = 325 = 5^2 \cdot 13$ den Wert $n - 1 = 324 = 81 \cdot 2^2$. In Tabelle 4 berechnen wir a^{81}, a^{162} und a^{324}, alle modulo 325,

a	$b_0 = a^{81}$	$b_1 = a^{162}$	$b_2 = a^{324}$	F-Z.?	MR-Z.?
2	252	129	66	×	×
7	307	324	1		
32	57	324	1		
49	324	1	1		
65	0	0	0	×	×
126	1	1	1		
201	226	51	1		×
224	274	1	1		×

Tabelle 4: Potenzen a^{n-1} mod n mit Zwischenschritten, $n = 325$

für verschiedene a.

Die Grundidee des Miller-Rabin-Tests ist nun, diese verlangsamte Berechnung der Potenz a^{n-1} mod n auszuführen und dabei nach nichttrivialen Quadratwurzeln der 1 Ausschau zu halten.

Im Beispiel sehen wir, dass 2 ein F-Zeuge für 325 ist, der in $\mathbb{Z}_{325}^*$ liegt, und dass 65 ein F-Zeuge ist, der nicht in $\mathbb{Z}_{325}^*$ liegt. Dagegen sind 7, 32, 49, 126, 201 und 224 alles F-Lügner für 325. Wenn wir aber 201^{324} mod 325 mit zwei Zwischenschritten berechnen, dann entdecken wir, dass 51 eine nichttriviale Quadratwurzel der 1 ist, was beweist, dass 325 keine Primzahl ist. Ähnlich liefert die Berechnung mit Basis 224, dass 274 eine nichttriviale Quadratwurzel der 1 ist. Die entsprechenden Berechnungen mit 7, 32 und 49 dagegen liefern keine weiteren Informationen, weil $7^{162} \equiv 32^{162} \equiv -1 \pmod{325}$ und $49^{81} \equiv -1 \pmod{325}$ gilt. Auch die Berechnung der Potenzen von 126 liefert keine nichttriviale Quadratwurzel der 1, weil 126^{81} mod $325 = 1$ gilt.

Wie kann die Folge $b_0, \ldots, b_k$ überhaupt aussehen? Wenn $b_i = 1$ or $b_i = n-1$ gilt, dann sind $b_{i+1}, \ldots, b_k$ alle gleich 1. Daher beginnt die Folge $b_0, \ldots, b_k$ mit einer (eventuell leeren) Folge von Elementen $\notin \{1, n-1\}$ und endet mit einer (eventuell leeren) Folge von Einsen. Diese beiden Teile können von einem Eintrag $n-1$ getrennt sein; dies muss aber nicht so sein. Die möglichen Muster sind in Tabelle 5 angegeben. Dabei steht "$*$" für ein Element $\notin \{1, n-1\}$. Wir unterscheiden vier Fälle, mit einigen Unterfällen:

b_0	b_1	$\cdots$				$\cdots$	b_{k-1}	b_k	Fall	F-Z.?	MR-Z.?
1	1	$\cdots$	1	1	1	$\cdots$	1	1	1		
$n-1$	1	$\cdots$	1	1	1	$\cdots$	1	1	2a		
$*$	$*$	$\cdots$	$*$	$n-1$	1	$\cdots$	1	1	2b		
$*$	$*$	$\cdots$	$*$	$*$	$*$	$\cdots$	$*$	$\neq 1$	3	$\times$	$\times$
$*$	$*$	$\cdots$	$*$	1	1	$\cdots$	1	1	4a		$\times$
$*$	$*$	$\cdots$	$*$	$*$	$*$	$\cdots$	$*$	1	4b		$\times$

Tabelle 5: Potenzen a^{n-1} mod n berechnet mit Zwischenschritten, mögliche Fälle.

Fall 1: $b_0 = 1$.

Fall 2: In $b_0, \ldots, b_{k-1}$ kommt der Wert $n-1$ vor.
(Fall 2a: $b_0 = n-1$; Fall 2b: $b_i = n-1$ für ein $i \in \{1, \ldots, k-1\}$.)

Fall 3: $b_k \neq 1$. – Dann ist n zusammengesetzt, weil a ein F-Zeuge für n ist.
(Alle Werte $b_0, \ldots, b_{k-1}$ müssen von 1 und $n-1$ verschieden sein.)

Fall 4: $b_0 \notin \{1, n-1\}$ und es gibt ein i mit $0 < i \leq k$ mit $b_{i-1} \notin \{1, n-1\}$ und $b_i = 1$.
(Fall 4a: $i < k$; Fall 4b: $i = k$.)

Wir beobachten: Wenn n eine Primzahl ist, dann gilt nach dem kleinen Satz von Fermat für jedes a die Gleichung $b_k = a^{n-1} \bmod n = 1$. Weiter kann es nicht passieren, dass $b_{i-1} \neq n-1$ und $b_i = 1$ ist. Daraus folgt, dass für eine Primzahl n *nur* die Fälle 1 und 2 vorkommen können. Wenn wir bei Eingaben n, die Primzahlen sind, keinen Fehler machen wollen, müssen wir also in Fällen 1 und 2 die Ausgabe 0 erzeugen. (Bei einer zusammengesetzten Zahl n als Eingabe tritt Fall 1 etwa für $a = 1$ auf und Fall 2a für $a = n-1$, weil u ungerade ist. Aber auch Fall 2b kann vorkommen; in Tab. 4 ist etwa $a = 32$ eine solche Zahl.)

In den Fällen 3 und 4 stellt die Zahl a (mit ihren Potenzen $b_0, \ldots, b_k$) hingegen einen Beleg dafür dar, dass n keine Primzahl ist: Im Fall 3 ist a ein F-Zeuge, im Fall 4 ist b_{i-1} eine nichttriviale Quadratwurzel der 1, und das kann nur passieren, wenn n zusammengesetzt ist. Hier können wir also 1 ausgeben.

Das ist auch schon der ganze Algorithmus von Miller-Rabin, abstrakt formuliert: Wähle a aus $\{1, \ldots, n-1\}$ zufällig. Finde heraus, ob Fall 1 oder 2 eintritt (dann ist die Ausgabe 0) oder Fall 3 oder 4 eintritt (dann ist die Ausgabe 1).

Um einen besonders effizienten Algorithmus zu bekommen, wollen wir die Fallunterscheidung noch etwas stromlinienförmiger gestalten. Betrachten von Tab. 5 zeigt, dass Fall 1 oder 2 genau dann eintritt, wenn Folgendes gilt:

$$b_0 = 1 \qquad \text{oder} \qquad n-1 \text{ erscheint in der Folge } b_0, \ldots, b_{k-1} \text{ zu } a. \tag{2}$$

(Interessanterweise spielt das letzte Folgenglied b_k gar keine Rolle für die Unterscheidung!) Dies führt zu folgender Definition in Analogie zu der der F-Zeugen und F-Lügner.

Definition 5.35
Sei $n \geq 3$ ungerade. Schreibe $n - 1 = u \cdot 2^k$, für u ungerade, $k \geq 1$. Eine Zahl a, $1 \leq a < n$, heißt ein **MR-Zeuge für n**, wenn (2) *nicht gilt*, d. h. wenn $a^u \bmod n \neq 1$ und $a^{u \cdot 2^i} \bmod n \neq n-1$ für alle i mit $0 \leq i < k$ gilt (Fall 3/4).
Wenn n zusammengesetzt ist und a, $1 \leq a < n$, kein MR-Zeuge ist, also (2) gilt, dann heißt a ein **MR-Lügner für n** (Fall 1/2). Die Menge der MR-Lügner nennen wir L_n^{MR}.

Aus der obigen Diskussion der möglichen Fälle folgt sofort:

Lemma 5.36
Sei $n \geq 3$ ungerade.

(a) a ist F-Zeuge für $n \quad \Rightarrow \quad a$ ist MR-Zeuge für n (Fall 3).

(b) Es gibt einen MR-Zeugen a für $n \quad \Rightarrow \quad n$ ist zusammengesetzt (Fall 3/4).

Im Jahr 1976 stellte G. M. Miller einen deterministischen Algorithmus vor, der auf der Idee beruhte, die Folge $b_0, \ldots, b_k$ zu betrachten. Er testete die kleinsten $O((\log n)^2)$ Elemente $a \in \{2, 3, \ldots, n-1\}$ auf die Eigenschaft, MR-Zeugen zu sein. Der Algorithmus hat polynomielle Laufzeit und liefert das korrekte Ergebnis, wenn die „erweiterte Riemannsche Vermutung (ERH)“ stimmt, eine berühmte Vermutung aus der Zahlentheorie, die bis heute unbewiesen ist. Um 1977 schlug M. Rabin vor, Millers Algorithmus so zu modifizieren, dass die zu testende Zahl a zufällig gewählt wird. Er und (unabhängig) Louis Monier bewiesen (erschienen 1980), dass dieser Algorithmus konstante Fehlerwahrscheinlichkeit haben wird. Der Algorithmus heißt heute allgemein der ***Miller-Rabin-Test***.

Der Test selbst ist leicht und sehr effizient durchführbar: Man berechnet zunächst $b_0 = a^u \bmod n$ und testet, ob $b_0 \in \{1, n-1\}$ ist. Falls ja, trifft Fall 1 bzw. Fall 2a zu, die Ausgabe ist 0. Falls nein, berechnet man durch iteriertes Quadrieren nacheinander die Werte $b_1, b_2, \ldots$, bis

- entweder der Wert $n-1$ auftaucht (Fall 2b, Ausgabe 0)
- oder der Wert 1 auftaucht (Fall 4a, a ist MR-Zeuge, Ausgabe 1)
- oder $b_1, \ldots, b_{k-1}$ berechnet worden sind, ohne dass Werte $n-1$ oder 1 vorgekommen sind (Fall 3 oder 4b, a ist MR-Zeuge, Ausgabe 1).

In Pseudocode sieht das Verfahren dann folgendermaßen aus. Im Unterschied zur bisherigen Beschreibung wird nur eine Variable `b` benutzt, die nacheinander die Werte $b_0, b_1, \ldots$ aufnimmt.

Algorithmus 5.6 ***Der Miller-Rabin-Primzahltest***
EINGABE: Eine ungerade Zahl $n \geq 5$.
METHODE:

```
1    Bestimme u ungerade und k ≥ 1 mit n − 1 = u · 2^k;
2    wähle zufällig ein a aus {1, ..., n − 1};
3    b ← a^u mod n;                          // mit schnellem Potenzieren
4    if b ∈ {1, n − 1} then return 0;        // Fall 1 oder 2a
5    for j from 1 to k − 1 do                // „wiederhole (k − 1)-mal“
6        b ← b^2 mod n;
7        if b = n − 1 then return 0;         // Fall 2b
8        if b = 1 then return 1;             // Fall 4a
9    return 1.                               // Fall 3 oder 4b
```

Wie steht es mit der Laufzeit des Algorithmus? Man benötigt höchstens $\log n$ Divisionen durch 2, um u und k zu finden. (Zeile 1). (Wenn man benutzt, dass n normalerweise in Binärdarstellung gegeben sein wird, ist die Sache noch einfacher: k ist die Zahl der Nullen, mit der die Binärdarstellung von $n-1$ aufhört, u ist die Zahl, die durch den Binärstring ohne diese letzten Nullen dargestellt wird.) Die Berechnung von $a^u \bmod n$ mit schneller Exponentiation 3 benötigt $O(\log n)$ arithmetische Operationen und $O((\log n)^3)$ Bitoperationen (mit der Schulmethode für Multiplikation und Division). Die Schleife in Zeilen 5–8 wird höchstens k-mal durchlaufen; offenbar ist $k \leq \log n$. In jedem Durchlauf ist die Multiplikation modulo n die teuerste Operation. Insgesamt benutzt der Algorithmus $O(\log n)$ arithmetische Operationen auf Zahlen,

die kleiner als n^2 sind, und $O((\log n)^3)$ Bitoperationen. Wenn man die Laufzeit mit dem Fermat-Test vergleicht, und dazu die Details der schnellen Exponentiation ansieht, stellt man fest, dass die Multiplikationen modulo n in beiden Fällen exakt gleich aussehen, nur dass beim Miller-Rabin-Test „unterwegs“ noch Zwischenergebnisse darauf getestet werden, ob sie gleich 1 oder gleich $n-1$ sind – dies verursacht kaum Mehraufwand.

Nun wenden wir uns dem Ein-/Ausgabeverhalten des Miller-Rabin-Tests zu.

Lemma 5.37
Wenn die Eingabe n für den Miller-Rabin-Test eine Primzahl ist, dann wird 0 ausgegeben.

Beweis: Wir haben oben schon festgestellt, dass Ausgabe 1 genau dann erfolgt, wenn die zufällig gewählte Zahl a ein MR-Zeuge ist. Nach Lemma 5.36(b) gibt es nur dann MR-Zeugen, wenn n zusammengesetzt ist. □

5.2.5 Fehlerschranke für den Miller-Rabin-Test

Wir müssen nun noch die Fehlerwahrscheinlichkeit des Miller-Rabin-Tests für den Fall analysieren, dass n eine zusammengesetzte (ungerade) Zahl ist. Dies wird also für diesen Abschnitt angenommen.

Wir beweisen, dass die Wahrscheinlichkeit, dass der Miller-Rabin-Test die (unerwünschte) Antwort 0 gibt, kleiner als $\frac{1}{2}$ ist. Dazu wollen wir ähnlich vorgehen wie bei der Analyse des Fermat-Tests für Nicht-Carmichael-Zahlen (Beweis von Satz 5.31). Dort haben wir gezeigt, dass die F-Lügner für solche n höchstens die Hälfte von $\mathbb{Z}_n^*$ ausmachen.

Wenn n keine Carmichael-Zahl ist, ist dies leicht: Nach Lemma 5.36(a) gilt $L_n^{\mathrm{MR}} \subseteq L_n^{\mathrm{F}}$, und wir können den Beweis von Satz 5.31 verwenden.

Es bleibt der Fall, dass n eine Carmichael-Zahl ist. Unser Plan ist, eine Menge B_n mit $L_n^{\mathrm{MR}} \subseteq B_n \subseteq \mathbb{Z}_n^*$ zu definieren, die höchstens die Hälfte der Elemente von $\mathbb{Z}_n^*$ enthält.

Weil u ungerade ist, gilt

$$(n-1)^{u\cdot 2^0} \equiv (n-1)^u \equiv (-1)^u \equiv -1 \equiv n-1 \pmod{n}.$$

Sei i_0 das größte $i \in \{0,1,\ldots,k\}$ mit der Eigenschaft, dass es einen MR-Lügner $a_\#$ mit $a_\#^{u\cdot 2^i} \bmod n = n-1$ gibt. (Wir werden dieses $a_\#$ weiter unten nochmals benutzen.)

a	b_0	b_1	$\cdots$		b_{i_0}		$\cdots$	b_{k-1}	b_k
$a_1 = 1$	1	1	$\cdots$	1	1	1	$\cdots$	1	1
$a_2 = n-1$	$n-1$	1	$\cdots$	1	1	1	$\cdots$	1	1
a_3	$*$	$*$	$\cdots$	$*$	$n-1$	1	$\cdots$	1	1
a_4	$*$	$*$	$\cdots$	$n-1$	1	1	$\cdots$	1	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	1	1
a	a^u	$a^{u\cdot 2^1}$	$\cdots$	$a^{u\cdot 2^{i_0-1}}$	$a^{u\cdot 2^{i_0}}$	$a^{u\cdot 2^{i_0+1}}$	$\cdots$	$a^{u\cdot 2^{k-1}}$	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$a_\#$	$*$	$*$	$\cdots$	$*$	$n-1$	1	$\cdots$	1	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$a_?$	$*?$	$*?$	$\cdots$	$*?$	$*?$	$\cdots$	$\cdots$	$\cdots$	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$a_{\varphi(n)}$	?	?	$\cdots$	?	?	?	$\cdots$	?	1

Tabelle 6: Sicht auf B_n als Teilmenge von $\mathbb{Z}_n^*$

Weil n eine Carmichael-Zahl ist, gilt $a^{u\cdot 2^k} \bmod n = a^{n-1} \bmod n = 1$ für alle $a \in \mathbb{Z}_n^*$, und daher $0 \leq i_0 < k$. Wir definieren:

$$B_n := \{a \in \mathbb{Z}_n^* \mid a^{u\cdot 2^{i_0}} \bmod n \in \{1, n-1\}\}. \tag{3}$$

Zur Veranschaulichung betrachte man Tabelle 6. In der dort angegebenen Matrix entspricht jede Zeile einem der $\varphi(n)$ Elemente von $\mathbb{Z}_n^*$, beginnend mit $a_1 = 1$ und $a_2 = -1 \equiv n-1$. In der Zeile für a ist die von a erzeugte Folge $b_0, \ldots, b_k$ eingetragen. Dass n eine Carmichael-Zahl ist, drückt sich dadurch aus, dass alle Einträge in der b_k-Spalte gleich 1 sind. Der Eintrag für $a_2 = -1$ in der b_0-Spalte ist $n-1$. Spalte i_0 ist die am weitesten rechts stehende Spalte, in der es einen Eintrag $-1 \equiv n-1$ gibt, und $a_\#$ ist ein Element von $\mathbb{Z}_n^*$, das zu diesem Eintrag führt. Die Menge B_n besteht aus den Elementen von $\mathbb{Z}_n^*$, die in Spalte i_0 Eintrag 1 oder $n-1$ haben. – Wir werden zeigen, dass diese Menge B_n die gewünschten Eigenschaften hat.

Lemma 5.38

(a) $L_n^{\mathrm{MR}} \subseteq B_n$.

(b) $B_n \subsetneq \mathbb{Z}_n^*$.

(c) $|B_n| \leq \frac{1}{2}|\mathbb{Z}_n^*|$, also $\Pr_{a\in\{1,\ldots,n-1\}}(a \text{ ist MR-Lügner}) < \frac{1}{2}$.

Beweis: Teil (a) ist nur die Überprüfung, dass die Definition technisch sinnvoll ist. Der eigentlich interessante Teil ist (b). Teil (c) ist dann Routine.

(a) Sei a ein beliebiger MR-Lügner.
Fall 1: $a^u \bmod n = 1$. – Dann ist auch $a^{u\cdot 2^{i_0}} \bmod n = 1$, und daher gilt $a \in B_n$. (Die Zeile zu a in Abb. 6 sieht aus wie die von $a_1 = 1$.)
Fall 2: $a^{u\cdot 2^i} \bmod n = n-1$ für ein $i < k$. – Dann ist $0 \leq i \leq i_0$ nach der Definition von i_0. Wenn $i = i_0$ ist, haben wir direkt $a \in B_n$ (Beispiel in Tab. 6: a_3 oder $a_\#$); wenn $i < i_0$ ist, dann gilt

$$a^{u\cdot 2^{i_0}} \bmod n = (a^{u\cdot 2^i} \bmod n)^{2^{i_0-i}} \bmod n = (-1)^{2^{i_0-i}} \bmod n = 1,$$

also ebenfalls $a \in B_n$ (Beispiel in Tab. 6: a_2 oder a_4).

(b) Wir benutzen die in Lemma 5.33 beobachtete Eigenschaft von Carmichael-Zahlen, keine Primzahlpotenz zu sein. Nach diesem Lemma können wir $n = n_1 \cdot n_2$ schreiben, für teilerfremde ungerade Zahlen n_1, n_2. Im Folgenden werden wir diese Zerlegung und die Eigenschaften aus dem Chinesischen Restsatz (Fakt 5.18) benutzen.

Die grobe Idee der nun folgenden Konstruktion ist folgende: Wir haben das Element 1 mit $1^{u\cdot 2^{i_0}} \bmod n = 1$ und das Element $a_\#$ mit $a_\#^{u\cdot 2^{i_0}} \equiv -1 \pmod n$. Aus diesen beiden Elementen „basteln“ wir mit Hilfe des Chinesischen Restsatzes ein $b \in \mathbb{Z}_n^*$, das sich modulo n_1 wie 1 und modulo n_2 wie $a_\#$ verhält. Es wird sich zeigen, dass $b^{u\cdot 2^{i_0}} \bmod n \notin \{1, n-1\}$ gilt, also $b \notin B_n$ ist.

Wir führen diese Idee jetzt formal durch. Sei $x_2 = a_\# \bmod n_2$. Nach dem Chinesischen Restsatz (Fakt 5.18) gibt es eine eindeutig bestimmte Zahl $b \in \{0, 1, \ldots, n-1\}$, die

$$b \equiv 1 \pmod{n_1} \quad \text{und} \quad b \equiv x_2 \pmod{n_2} \tag{4}$$

erfüllt. (Es folgt $b \equiv a_\# \pmod{n_2}$.) Wir zeigen, dass b in $\mathbb{Z}_n^* - B_n$ liegt.

Wir notieren, dass für beliebige $x, y \in \mathbb{Z}$ gilt:

$$x \equiv x' \pmod n \qquad \Rightarrow \qquad x \equiv x' \pmod{n_i}, \text{ für } i = 1, 2. \tag{5}$$

(Wenn $x - x'$ durch n teilbar ist, dann auch durch n_1 und n_2.)

Beh. 1: $b \in \mathbb{Z}_n^*$.
Bew.: Offensichtlich gilt $b^{n-1} \equiv 1^{n-1} \equiv 1 \pmod{n_1}$. Weiter haben wir, modulo n_2 gerechnet:

$$b^{n-1} \equiv a_\#^{n-1} \overset{(5)}{\equiv} ((a_\#^{n-1}) \bmod n) \equiv 1 \pmod{n_2}.$$

Wegen der Eindeutigkeitsaussage im Chinesischen Restsatz folgt daraus $b^{n-1} \equiv 1 \pmod{n}$. Nach Lemma 5.30 folgt $b \in \mathbb{Z}_n^*$.

Beh. 2: $b \notin B_n$.
Bew.: Indirekt. Annahme: $b \in B_n$, d. h. $b^{u \cdot 2^{i_0}} \equiv 1 \pmod{n}$ oder $b^{u \cdot 2^{i_0}} \equiv -1 \pmod{n}$.
1. *Fall*: $b^{u \cdot 2^{i_0}} \equiv 1 \pmod{n}$. – Mit (5) folgt $b^{u \cdot 2^{i_0}} \equiv 1 \pmod{n_2}$. Andererseits gilt

$$b^{u \cdot 2^{i_0}} \equiv x_2^{u \cdot 2^{i_0}} \equiv a_\#^{u \cdot 2^{i_0}} \overset{(5)}{\equiv} (a_\#^{u \cdot 2^{i_0}} \bmod n) \equiv n - 1 \equiv -1 \pmod{n_2},$$

ein Widerspruch, weil n_2 ungerade ist.
2. *Fall*: $b^{u \cdot 2^{i_0}} \equiv -1 \pmod{n}$. – Mit (5) folgt $b^{u \cdot 2^{i_0}} \equiv -1 \pmod{n_1}$. Andererseits gilt

$$b^{u \cdot 2^{i_0}} \equiv 1^{u \cdot 2^{i_0}} \equiv 1 \pmod{n_1},$$

ebenfalls ein Widerspruch, weil n_1 ungerade ist.

(c) Wir verwenden die in (b) konstruierte Zahl $b \in \mathbb{Z}_n^* - B_n$. Wie im Beweis von Satz 5.31 betrachtet man die injektive Funktion $g_b \colon B_n \ni a \mapsto ba \bmod n \in \mathbb{Z}_n^*$. Es gilt $g_b(a) \notin B_n$ für jedes $a \in B_n$, denn

$$g_b(a)^{u \cdot 2^{i_0}} \bmod n = (b^{u \cdot 2^{i_0}} \bmod n)(a^{u \cdot 2^{i_0}}) \bmod n \in \{b^{u \cdot 2^{i_0}} \bmod n, (n - b^{u \cdot 2^{i_0}}) \bmod n\}.$$

Daraus ergibt sich sofort $|B_n| \leq \frac{1}{2}|\mathbb{Z}_n^*|$. □

Wir haben also eine Schranke von $\frac{1}{2}$ für die Irrtumswahrscheinlichkeit im Miller-Rabin-Algorithmus bewiesen. Eine etwas kompliziertere Analyse zeigt, dass sogar die Fehlerschranke $\frac{1}{4}$ gilt; man kann auch zeigen, dass es zusammengesetzte Zahlen n gibt (zum Beispiel $703 = 19 \cdot 37$), bei denen eine Fehlerwahrscheinlichkeit von fast $\frac{1}{4}$ tatsächlich auftritt. Durch ℓ-fache Wiederholung, ebenso wie in Algorithmus 5.5, kann man die Fehlerschranke auf $4^{-\ell}$ reduzieren. Wir kürzen den Miller-Rabin-Test mit „MiRa-Test“ ab. Man beachte, dass bei jedem neuen Aufruf eine neue Zufallszahl a gewählt wird.

Algorithmus 5.7 ***Iterierter MR-Test***
EINGABE: Ungerade Zahl $n \geq 5$, eine Zahl $\ell \geq 1$.

METHODE:
1 **repeat** ℓ **times**
2 **if** MiRa-Test$(n) = 1$ **then return** 1;
3 **return** 0;

Diesen Test wollen wir kurz IterMiRa(n, ℓ) nennen. – Wir erhalten zusammenfassend:

Proposition 5.39
Algorithmus 5.7 benötigt $O(\ell \cdot \log n)$ arithmetische Operationen auf Zahlen, die kleiner als n^2 sind, und $O(\ell \cdot (\log n)^3)$ Bitoperationen. Wenn n eine Primzahl ist, ist die Ausgabe 0, wenn n zusammengesetzt ist, ist die Wahrscheinlichkeit, dass 0 ausgegeben wird, kleiner als $4^{-\ell}$. □

5.3 Die Erzeugung von Primzahlen

Für kryptographische Anwendungen (zum Beispiel für die Erzeugung von Schlüsselpaaren für das RSA-Public-Key-Kryptosystem) werden vielziffrige Primzahlen benötigt. Eine typische Aufgabe in diesem Zusammenhang könnte lauten:

Finde eine zufällige Primzahl mit μ Bits!

Dabei hängt μ von der Anwendung ab; wir können uns z. B. $\mu = 512$ oder $\mu = 2048$ sein.

Man erinnere sich an den Primzahlsatz und an den Satz von Finsler, die angeben, wie viele Primzahlen es in $[m, 2m)$ gibt.

Ein naheliegender Ansatz zur Erzeugung einer zufälligen Primzahl in $[m, 2m)$ ist folgender: Man wählt wiederholt eine (ungerade) Zahl aus $[m, 2m)$ zufällig und wendet auf sie den Miller-Rabin-Test an. Dies wiederholt man, bis eine Zahl gefunden wurde, die die Ausgabe 0 liefert.

Algorithmus 5.8 ***GetPrime – Randomisierte Primzahlerzeugung***
EINGABE: Zahl $m \geq 2$, Zahl $\ell \geq 1$ // Zuverlässigkeitsparameter .
METHODE:
1 **repeat**
2 $\mathtt{n} \leftarrow$ zufällige ungerade Zahl in $[m, 2m)$;
3 **until** IterMiRa$(\mathtt{n}, \ell) = 0$; // Algorithmus 5.7
4 **return** n.

Die Ausgabe ist korrekt, wenn er eine Primzahl zurückgibt, ein Fehler tritt auf, wenn eine zusammengesetzte Zahl zurückgegeben wird. Auf den ersten Blick könnte man meinen (aufgrund von Proposition 5.39), dass die Fehlerwahrscheinlichkeit höchstens $1/4^\ell$ ist – eben die Fehlerwahrscheinlichkeit des iterierten MiRa-Tests. Wir werden sehen, dass wir auf der Basis der Analyse des Miller-Rabin-Tests nur ein etwas schwächeres Ergebnis bekommen.[7]

Wir definieren: $\text{Prim}_m := \{p \in [m, 2m) \mid p \text{ ist Primzahl}\}$.

Satz 5.40
Bei der Anwendung von Algorithmus 5.8 auf eine gerade Zahl m gilt:

(a) Für jedes $p \in \text{Prim}_m$ gilt:

$$\Pr(\text{GetPrime}(m, \ell) = p \mid \text{GetPrime}(m, \ell) \in \text{Prim}_m) = \frac{1}{|\text{Prim}_m|}.$$

In Worten: Jede Primzahl in $[m, 2m)$ hat dieselbe Wahrscheinlichkeit, als Ergebnis zu erscheinen.

(b)

$$\Pr(\text{GetPrime}(m, \ell) \notin \text{Prim}_m) \leq \frac{3\ln(2m)}{2 \cdot 4^\ell} = O\left(\frac{\log m}{4^\ell}\right).$$

((!) Nicht $1/4^\ell$, wie naiv vermutet.*)*

(c) Die erwartete Rundenzahl ist $\leq \frac{3}{2}\ln(2m)$, der erwartete Rechenaufwand ist $O((\log n)^2)$ arithmetische Operationen und $O((\log n)^4)$ Bitoperationen.

Beweis: (a) Eine Primzahl wird als Resultat genau dann geliefert, wenn keine zusammengesetzte Zahl fälschlicherweise vom Miller-Rabin-Test akzeptiert wird, bevor (in Zeile 2) die erste echte Primzahl gewählt wird. Jede der Primzahlen in $[m, 2m)$ hat dieselbe Wahrscheinlichkeit, diese erste gewählte Primzahl zu sein.

[7] Tatsächlich ist die Fehlerwahrscheinlichkeit durch $1/4^\ell$ beschränkt, für genügend große m. Dies kann man aber nur durch fortgeschrittene zahlentheoretische Untersuchungen über das Verhalten einer zufälligen ungeraden Zahl beim Miller-Rabin-Test beweisen [P. Beauchemin, G. Brassard, C. Crépeau, C. Goutier, C. Pomerance: The generation of random numbers that are probably prime. *J. Cryptology* **1**(1): 53-64 (1988)].

(b)

$$
\begin{aligned}
&\Pr(\text{GetPrime}(m,\ell) \notin \text{Prim}_m) \\
&= \Pr\Big(\exists i \geq 1\text{: in Runden } j = 1,\ldots,i-1 \text{ wird eine} \\
&\qquad\qquad \text{zusammengesetzte Zahl gewählt } \textit{und erkannt } \wedge \\
&\qquad\qquad \text{in Runde } i \text{ wird zusammengesetzte Zahl } n \text{ gewählt} \\
&\qquad\qquad \text{und der iterierte MiRa-Test auf } n \text{ liefert } 0\Big) \\
&\leq \sum_{i\geq 1} \left(1 - \frac{|\text{Prim}_m|}{m/2}\right)^{i-1} \cdot \frac{1}{4^\ell} \\
&= \frac{m/2}{|\text{Prim}_m|} \cdot \frac{1}{4^\ell} \\
&\leq \frac{3\ln(2m)}{2\cdot 4^\ell}.
\end{aligned}
$$

Wir haben benutzt, dass es in $[m, 2m)$ genau $m/2$ ungerade Zahlen gibt (m ist gerade). Die letzte Ungleichung folgt aus der Ungleichung von Finsler (Satz 5.28).

(c) Da man in jeder Runde mit Wahrscheinlichkeit mindestens $\frac{|\text{Prim}_m|}{m/2}$ eine Primzahl wählt, ist die erwartete Rundenzahl nicht größer als $\frac{m/2}{|\text{Prim}_m|} = O(\log m)$. □

Bemerkung: Bei der Erzeugung zufälliger Primzahlen für kryptographische Zwecke wird man aus Effizienzgründen nicht unseren Algorithmus anwenden, der $(\log n)^2$ Multiplikationen benötigt, sondern eine Kombination zweier verschiedener Primzahltests (z. B. Miller-Rabin und „Lucas Strong Probable Prime Test“) mit sehr wenigen Iterationen und einem Test auf Teilbarkeit durch sehr kleine Primteiler. Dies erfordert nur $O(\log n)$ Multiplikationen. Die Dichte der zusammengesetzten Zahlen, die von einem solchen Test nicht erkannt werden, ist als sehr gering einzuschätzen. Über eine interessante experimentelle Untersuchung hierzu berichtet die kurze Notiz
`people.csail.mit.edu/rivest/Rivest-FindingFourMillionLargeRandomPrimes.ps`
von Ron Rivest (bekannt von „RSA“ und von „Cormen, Leiserson, Rivest und Stein“).

5.4 Quadratische Reste und Quadratwurzeln modulo p

Wir betrachten hier den Begriff des quadratischen Rests und der Quadratwurzeln modulo p, für Primzahlen p.

Beispiel: Für $p = 13$ und $p = 19$ betrachten wir die Quadrate der Zahlen in $\mathbb{Z}_p^*$.

a	1	2	3	4	5	6	7	8	9	10	11	12
a^2	1	4	9	16	25	36	49	64	81	100	121	144
$a^2 \bmod 13$	1	4	9	3	12	10	10	12	3	9	4	1

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$a^2 \bmod 19$	1	4	9	16	6	17	11	7	5	5	7	11	17	6	16	9	4	1

Wir bemerken, dass in beiden Fällen jede Zahl entweder keinmal oder zweimal als Quadrat vorkommt, dass infolgedessen genau $\frac{1}{2}(p-1)$ Zahlen in $\mathbb{Z}_p^*$ als Quadratzahlen auftauchen. Im Fall $p = 13$ ist $12 = -1 \bmod 13$ eine Quadratzahl, im Fall $p = 19$ ist $18 = -1 \bmod 19$ keine Quadratzahl. Es stellt sich die Frage, wie man zu gegebenem p und a ermitteln kann, ob a in $\mathbb{Z}_p$ ein Quadrat ist (einfach, wenn man weiß, wie es geht) und wie man gegebenenfalls ein c mit $c^2 \bmod p = a$ finden kann (eventuell nicht ganz so einfach).

Bemerkung: In $\mathbb{Z}_p$ spielt $p-1$ die Rolle von -1, dem additiven Inversen von 1.

Definition 5.41
Für eine ungerade Primzahl p sei

$$\mathrm{QR}_p := \{a \in \mathbb{Z}_p^* \mid \exists b \in \mathbb{Z}_p^* : b^2 \bmod p = a\}$$

die Menge der „quadratischen Reste“ in $\mathbb{Z}_p^*$.

Wir stellen einige auch für sich wichtige Hilfsaussagen bereit.

Fakt 5.42
Wenn p eine ungerade Primzahl ist und $s \geq 2$, dann gilt:

(a) Wenn $d \geq 1$ ist, dann hat ein Polynom $f(X) = X^d + a_{d-1}X^{d-1} + \cdots + a_1X + a_0$ in $\mathbb{Z}_p$ höchstens d Nullstellen.

(b) Für $s \geq 1$ gibt es in $\mathbb{Z}_p^*$ höchstens s Elemente b mit $b^s \bmod p = 1$.

Beweis: (a) Dies ist eine Tatsache, die in Körpern allgemein gilt. Man kann sich recht leicht überlegen, dass man für Nullstellen $b_1, \ldots, b_k$ aus $f(X)$ den Faktor $(X - b_1) \cdots (X - b_k)$ ausklammern kann, d. h. $f(X) = (X - b_1) \cdots (X - b_k) \cdot g(X)$ schreiben kann. Dann kann k nicht größer als der Grad d von $f(X)$ sein.
(b) Wenn $b_1, \ldots, b_{s+1}$ verschiedene Lösungen wären, hätte das Polynom $X^s - 1$ Grad s, aber $> s$ Nullstellen, was (a) widerspricht. □

Lemma 5.43
Wenn p eine ungerade Primzahl ist, dann gilt $b^2 \bmod p = 1$, $b \in \mathbb{Z}_p$ genau für $b \in \{1, -1\} = \{1, p-1\}$.

Beweis: Offensichtlich gilt für *jedes beliebige* $m \geq 2$, dass $1^2 \bmod m = 1$ und $(m-1)^2 \bmod m = (m(m-2)+1) \bmod m = 1$ ist. Nach Fakt 5.42(b) kann das Polynom $X^2 - 1$ nicht mehr als diese beiden Nullstellen haben. □

Allgemeiner gilt:

Lemma 5.44
Wenn p eine ungerade Primzahl ist und $a \in \mathrm{QR}_p$, dann gibt es genau zwei Zahlen $b \in \mathbb{Z}_p$ mit $b^2 \bmod p = a$.

Folgerung: $|\mathrm{QR}_p| = \frac{1}{2}(p-1)$.

Beweis: Nach Definition von QR_p gibt es ein solches b. Dann gilt auch $(-b)^2 \bmod p = b^2 \bmod p = a$. Weil p ungerade ist, gilt $b - (-b) \bmod p = 2b \bmod p \neq 0$, also $b \neq -b$. Nach Fakt 5.42 kann das Polynom $X^2 - a$ nicht mehr als diese beiden Nullstellen haben. □

Die folgende Behauptung erlaubt es zu testen, ob a ein quadratischer Rest ist.

Proposition 5.45 ***Eulersches Kriterium***
Für jede ungerade Primzahl p und jedes $a \in \mathbb{Z}_p^*$ gilt:

(a) $a^{(p-1)/2} \bmod p \in \{1, -1\}$.

(b) $a \in \mathrm{QR}_p \;\Leftrightarrow\; a^{(p-1)/2} \bmod p = 1$.

Beweis von Proposition 5.45:

(a) Nach dem kleinen Satz von Fermat (Fakt 5.16) gilt $a^{p-1} \bmod p = 1$ für jedes $a \in \mathbb{Z}_p^*$. Daraus:

$$\left(a^{(p-1)/2} \bmod p\right)^2 \bmod p = 1 \text{ für jedes } a \in \mathbb{Z}_p^*.$$

Mit Lemma 5.43 folgt $a^{(p-1)/2} \bmod p \in \{1, -1\} \subseteq \mathbb{Z}_p^*$.

(b) Wir definieren $A := \{a \in \mathbb{Z}_p^* \mid a^{(p-1)/2} \bmod p = 1\}$. Unser Ziel ist zu zeigen, dass $\mathrm{QR}_p = A$ ist. Die eine Inklusion ist leicht zu sehen:

Beh. 1: $\mathrm{QR}_p \subseteq A$. (Wenn $a \in \mathrm{QR}(p)$, also $a = b^2 \bmod p$ für ein b, dann gilt

$$(a^{(p-1)/2} \bmod p) \bmod p = ((b^2)^{(p-1)/2} \bmod p) \bmod p = b^{p-1} \bmod m = 1,$$

also ist $a \in A$, wieder nach dem kleinen Satz von Fermat.)

Für die Umkehrung benutzen wir ein Kardinalitätsargument.

Beh. 2: $|\mathrm{QR}_p| = \frac{1}{2}(p-1)$. Dies gilt, weil nach Lemma 5.44 jeder quadratische Rest genau zwei verschiedene Quadratwurzeln hat und natürlich jede der $p-1$ Zahlen in $\mathbb{Z}_p^*$ Quadratwurzel ist.

Beh. 3: $|A| \leq \frac{1}{2}(p-1)$. (Nach Fakt 5.42 (mit $s = \frac{1}{2}(p-1)$) gilt, dass höchstens $\frac{1}{2}(p-1)$ viele Elemente von $\mathbb{Z}_p^*$ die Gleichung $a^{(p-1)/2} \bmod p = 1$ erfüllen.)

Aus Beh. 1, 2 und 3 folgt sofort $\mathrm{QR}_p = A$, wie gewünscht. □

Mit diesem Kriterium kann man mit einer schnellen Exponentiation testen, ob eine Zahl a quadratischer Rest modulo p ist oder nicht.

Algorithmus 5.9 ***Quadratzahltest***
Input: Primzahl p, Zahl a mit $1 < a < p$
Methode:
(1) Berechne $z = a^{(p-1)/2} \bmod p$; // Schnelle Exponentiation
(2) **if** $z = 1$ **then return** „Quadrat“ **else return** „Nichtquadrat“.

Wir rechnen versuchsweise in $\mathbb{Z}_{19}$, für $18 \notin \mathrm{QR}_{19}$ und $17 \in \mathrm{QR}_{19}$:

$18^9 \bmod 19 = (((-1)^8 \bmod 19) \cdot 18) \bmod 19 = 18 \neq 1$;
$17^9 \bmod 19 = (((-2)^8 \bmod 19) \cdot (-2)) \bmod 19 = (256 \cdot (-2)) \bmod 19 = (9 \cdot (-2)) \bmod 19 = 1$.

Wir notieren eine leichte Folgerung aus dem Eulerschen Kriterium:

Korollar 5.46
Sei p eine ungerade Primzahl. Dann gilt:

(a) Für $a, b \in \mathbb{Z}_p^*$ und das Produkt $c = ab \bmod p$:

$$\begin{aligned} a, b \in \mathrm{QR}_p &\Rightarrow c \in \mathrm{QR}_p; \\ a, b \notin \mathrm{QR}_p &\Rightarrow c \in \mathrm{QR}_p; \\ a \in \mathrm{QR}_p, b \notin \mathrm{QR}_p &\Rightarrow c \notin \mathrm{QR}_p. \end{aligned}$$

(b) Für $a \in \mathbb{Z}_p^*$ und a^{-1} in $\mathbb{Z}_p^*$, das multiplikative Inverse von a:

$$a \in \mathrm{QR}_p \quad \Leftrightarrow \quad a^{-1} \in \mathrm{QR}_p.$$

((a) folgt durch Anwendung von Exponentiationsregeln und Proposition 5.45; (b) folgt aus $a \cdot a^{-1} = 1 \in \mathrm{QR}_p$ und (a).)

Wenn das so einfach geht – kann man dann zu einer Quadratzahl a auch leicht eine Quadratwurzel b modulo p berechnen?

Erstaunlicherweise gibt es hier zwei sehr verschiedene Fälle. Wir betrachten das Beispiel $p = 19$. Die Quadrate in $\mathbb{Z}_p^*$ sind die 9 Zahlen $1, 4, 5, 6, 7, 9, 11, 16, 17$. Wir berechnen $a^5 \bmod 19$ für diese Quadratzahlen:

a	1	4	5	6	7	9	11	16	17
$b = a^5 \bmod 19$	1	17	9	5	11	16	7	4	6
$b^2 \bmod 19$	1	4	5	6	7	9	11	16	17

Wir sehen: a^5 ist für jede der Quadratzahlen a eine Quadratwurzel. (Die andere ist natürlich dann $(19 - a^5) \bmod 19$.) Dies ist kein Zufall. Die Zahl 5 ergibt sich aus $p = 19$ als $\frac{1}{4}(p+1)$. Diese Operation ist für alle Primzahlen möglich, die um 1 unter einem Vielfachen von 4 liegen.

Proposition 5.47
Wenn p eine Primzahl ist derart, dass $p + 1$ durch 4 teilbar ist, dann ist für jedes $a \in \mathrm{QR}_p$ die Zahl $a^{(p+1)/4} \bmod p$ eine Quadratwurzel von a.

Beweis:

$$((a^{(p+1)/4}) \bmod p)^2 \bmod p = (a^{(p+1)/2}) \bmod p = (((a^{(p-1)/2}) \bmod p) \cdot a) \bmod p = a,$$

wobei wir Proposition 5.45(b) benutzt haben. □

Mit anderen Worten: Für die Primzahlen $p \in \{3, 7, 11, 19, 23, 31, 43, 47, 59, \dots\}$ läuft das Ziehen von Quadratwurzeln auf eine schnelle Exponentiation mit Exponent $\frac{1}{4}(p+1)$ hinaus. (Bitkomplexität $O((\log p)^3)$.)

Wie steht es mit den anderen Primzahlen $5, 13, 17, 29, 37, 41, 53, \dots$? Gibt es einen ähnlich effizienten Algorithmus zum Finden von Quadratwurzeln? Hier gibt es nochmals zwei Unterfälle. Wenn man leicht einen beliebigen nichtquadratischen Rest modulo p finden kann, dann kann man mit Hilfe eines deterministischen Algorithmus, der in Anhang **??** beschrieben ist, mit $O((\log p)^2)$ arithmetischen Operationen Quadratwurzeln finden. Dies ist der Fall, wenn $p - 5$ durch 8 teilbar ist, weil dann stets $2 \notin \mathrm{QR}_p$ gilt (Ergebnis der Zahlentheorie im Zusammenhang mit dem „Quadratischen Reziprozitätsgesetz“). Es bleiben die Primzahlen p, für die $p - 1$ durch 8 teilbar

ist, beispielsweise $17, 41, 73 \ldots$. Interessanterweise kennt man für solche Primzahlen keinen *deterministischen* Algorithmus, der Quadratwurzeln modulo p berechnet und dazu Zeit $(\log p)^{O(1)}$ braucht. Wir betrachten hier einen ***randomisierten*** Algorithmus für diese Aufgabe. Er funktioniert für alle Primzahlen der Form p, für die $p-1$ durch 4 teilbar ist.

Idee: Gegeben ein $a \in \mathrm{QR}_p$, bezeichne eine Quadratwurzel von a mit $\diamond$. Die andere Quadratwurzel von a ist dann $-\diamond$. Rechne mit $\diamond$ (in $\mathbb{Z}_p$) wie mit einer Unbekannten. Es entstehen Ausdrücke $\alpha + \beta \cdot \diamond$, die man addieren, subtrahieren und multiplizieren kann. Höhere Potenzen von $\diamond$ entstehen nicht, wenn $\diamond^2$ durch a ersetzt wird. Es gelten folgende Regeln:

$$\begin{aligned} (\alpha + \beta \cdot \diamond) \pm (\gamma + \delta \cdot \diamond) &= (\alpha \pm \gamma) + (\beta \pm \delta) \cdot \diamond; \\ (\alpha + \beta \cdot \diamond) \cdot (\gamma + \delta \cdot \diamond) &= (\alpha\gamma + \beta\delta \cdot a) + (\alpha\delta + \beta\gamma) \cdot \diamond. \end{aligned} \tag{6}$$

Wenn man diese Operationen *implementieren* will, stellt man ein Element $\alpha + \beta \cdot \diamond$ als (α, β) dar und rechnet wie in (6). Mit den in (6) angegebenen Operationen bildet die Menge der Paare (a, b) in $\mathbb{Z}_p \times \mathbb{Z}_p$ einen Ring mit 1. Insbesondere besitzt die Multiplikation ein neutrales Element, nämlich $(1, 0)$, und die Multiplikation ist kommutativ und assoziativ (prüft man leicht nach). Daher kann man das schnelle Exponentiationsverfahren aus Algorithmus 5.3 anwenden, um „symbolisch“ die Potenz $(\alpha + \beta \cdot \diamond)^i$ in $\log i$ Runden mit jeweils höchstens 10 Multiplikationen modulo p zu berechnen.

Wesentlich ist die folgende Grundeigenschaft. Wenn man in der abstrakten Notation eine Gleichheit ausgerechnet hat, dann gilt sie für beide Wurzeln von a:

Wenn sich mit den Regeln von (6) $(\alpha + \beta \cdot \diamond)^i$ zu $\gamma + \delta \cdot \diamond$ berechnet und wenn $w^2 \bmod p = a$ ist, dann gilt $(\alpha + \beta \cdot w)^i = \gamma + \delta \cdot w$.

(Man setzt w für $\diamond$ ein und verfolgt die symbolische Rechnung, aber nun in $\mathbb{Z}_p$.)

Algorithmus 5.10 ***Quadratwurzeln, randomisiert***
Input: Primzahl p, wobei $p-1$ durch 4 teilbar ist;
$a \in \{1, \ldots, p-1\}$ mit $a^{(p-1)/2} \bmod p = 1$.
Methode:
// 1–4: Finde b mit $b^2 = a$ oder $b^2 - a \notin \mathrm{QR}_p$:

1 **repeat**
2 $b :=$ Zufallszahl in $\{1, \ldots, p-1\}$;
3 **if** $b^2 \bmod p = a$ **then return** b; // Glück gehabt!
4 **until** $(b^2 - a)^{(p-1)/2} \bmod p = p - 1$; // $b^2 - a$ nicht-quadratischer Rest
5 $(c + d \cdot \diamond) \leftarrow (b + 1 \cdot \diamond)^{(p-1)/2}$;
6 // Schnelle Exponentiation mit Ausdrücken $\alpha + \beta \cdot \diamond$, Regeln (6)
7 Berechne $u = d^{-1}$ // im Körper $\mathbb{Z}_p$;
8 // Erweiterter Euklidischer Algorithmus
9 **return** $\{u, -u\}$.

Korrektheit: w sei eine Quadratwurzel von a; die andere ist dann $-w$. Nach der obigen Vorüberlegung gilt (in $\mathbb{Z}_p$ gerechnet), weil sowohl $w^2 = a$ als auch $(-w)^2 = a$ gilt:

$$\begin{aligned} c + dw &= (b + w)^{(p-1)/2} \in \{1, -1\} \text{ und} \\ c - dw &= (b - w)^{(p-1)/2} \in \{1, -1\}. \end{aligned} \tag{7}$$

Wir addieren diese beiden Gleichungen und erhalten:

$$2 \cdot c = (b + w)^{(p-1)/2} + (b - w)^{(p-1)/2}. \tag{8}$$

Andererseits ist $(b^2 - a)^{(p-1)/2} = -1$ (wegen Zeile 4 im Algorithmus). Wir setzen $a = w^2$ ein und erhalten

$$-1 = (b^2 - w^2)^{(p-1)/2} = (b + w)^{(p-1)/2} \cdot (b - w)^{(p-1)/2}.$$

Wegen (7) muss einer der Faktoren gleich 1 und der andere gleich -1 sein:

$$\{(b + w)^{(p-1)/2}, (b - w)^{(p-1)/2}\} = \{1, -1\}.$$

Mit (8) folgt $2 \cdot c = 0$, und damit $c = 0$, weil in $\mathbb{Z}_p$ gilt, dass $2 = 1 + 1 \neq 0$ ist. Subtraktion der Gleichungen in (7) ergibt

$$2dw = (b + w)^{(p-1)/2} - (b - w)^{(p-1)/2}.$$

Die beiden Terme in der Differenz sind 1 und -1 (in irgendeiner Reihenfolge), also ist $2dw \in \{2, -2\}$, oder (wieder weil $2 \neq 0$):

$$dw \in \{1, -1\}, \text{ d.h.: } w \in \{d^{-1}, -d^{-1}\},$$

wobei das multiplikative Inverse in $\mathbb{Z}_p$ berechnet wird. Weil nach der Ausgangssituation w und $-w$ die Quadratwurzeln von a sind, sind auf jeden Fall $u = d^{-1}$ und $-u = -d^{-1}$ diese Wurzeln.

Rechenzeit: In der Laufzeit des Algorithmus gibt es drei entscheidende Komponenten: (i) Das Finden eines Elements b derart dass b Quadratwurzel von a (sehr unwahrscheinlich) oder $b^2 - a \notin \mathrm{QR}_p$ (ein „nicht-quadratischer Rest“) ist; (ii) die Exponentiation zum Finden von $(a+1\cdot\diamond)^{(p-1)/2}$ (die höchstens $\log p$ Schleifendurchläufe mit je höchstens 10 modularen Multiplikationen erfordert); (iii) der Aufwand für den erweiterten Euklidischen Algorithmus. Teile (ii) und (iii) haben Kosten, die einer festen Anzahl modularer Exponentiationen entspricht, also $O(\log p)$ Multiplikationen und Additionen. Für den ersten Teil ist zu überlegen, was die Dichte

$$\frac{|G|}{p-1} = \frac{|\{b \in \mathbb{Z}_p^* \mid b^2 = a \ \vee \ b^2 - a \notin \mathrm{QR}_p\}|}{p-1}$$

der Menge G der „guten“ Elemente in $\mathbb{Z}_p^*$ ist.

Behauptung: $|G| = \frac{1}{2}(p+3)$.
Beweis der Behauptung: Die Elemente w und $-w$ sind verschiedene und jedenfalls gute Elemente. Es geht also um die Anzahl der guten Elemente in der Menge („*Nicht-Wurzeln*“)

$$NW := \mathbb{Z}_p^* - \{w, -w\}.$$

Wir definieren eine Funktion $f\colon NW \to \mathbb{Z}_p$ durch

$$f(b) := \frac{b+w}{b-w} \quad \text{(Arithmetik in } \mathbb{Z}_p\text{)}.$$

Diese Funktion ist wohldefiniert, da $b \neq w$, und sie nimmt den Wert 0 nicht an, da $b \neq -w$. Weiterhin wird der Wert 1 nicht angenommen (wenn $b + w = b - w$, wäre $2w = 0$, also $w = 0$), und der Wert -1 auch nicht (wenn $b + w = -(b - w)$, wäre $2b = 0$, also $b = 0$, was $b \in \mathbb{Z}_p^*$ widerspricht). Also gilt: $f\colon NW \to \{2, \ldots, p-2\}$. Man sieht leicht, dass f injektiv ist:

$$\begin{aligned}
\frac{b_1+w}{b_1-w} = \frac{b_2+w}{b_2-w} \quad &\Rightarrow \quad (b_1+w)(b_2-w) = (b_2+w)(b_1-w) \\
&\Rightarrow \quad b_1b_2 - wb_1 + wb_2 - w^2 = b_1b_2 + wb_1 - wb_2 - w^2 \\
&\Rightarrow \quad 2w(b_2-b_1) = 0 \\
&\Rightarrow \quad b_1 = b_2,
\end{aligned}$$

wobei wir $2 \neq 0$ und $w \neq 0$ benutzt haben. Aus der Injektivität folgt, dass f eine Bijektion zwischen NW und $\{2, \ldots, p-2\}$ ist. In der letzteren Menge gibt es $(p-1)/2 - 2$ quadratische Reste (weil 1 und $p-1$ quadratische Reste sind: $(p-1)/2$ ist gerade, daher ist $(-1)^{(p-1)/2} = 1$), also $p - 3 - ((p-1)/2 - 2) = (p-1)/2$ nicht-quadratische Reste. Bleibt zu zeigen:

$$b^2 - a \in \mathrm{QR}_p \quad \Leftrightarrow \quad \frac{b+w}{b-w} \in \mathrm{QR}_p.$$

Das gilt, weil $b^2 - a = (b+w)(b-w)$ ist, also $(b+w)/(b-w) = (b^2-a)/(b-w)^2$, und nach Korollar 5.46 die Menge der (nicht-)quadratischen Reste in $\mathbb{Z}_p$ unter Division mit quadratischen Resten (hier $(b-w)^2$) abgeschlossen ist. Wegen dieser Eigenschaft von f ist

$$|G| = 2 + |\{u \in NW \mid f(u) \notin \mathrm{QR}_p\}| = 2 + (p-1)/2 = (p+3)/2.$$

Damit ist die Behauptung bewiesen. – Die Wahrscheinlichkeit, in Zeile 2 ein Element von $|G|$ zu wählen, ist also $(p+3)/(2(p-1)) > \frac{1}{2}$. Damit ist die erwartete Anzahl von Versuchen, bis ein geeignetes b gefunden wird, kleiner als 2. □

Bemerkung: Es gibt einen anderen randomisierten Algorithmus zur Ermittlung von Quadratwurzeln modulo p, der mit $O((\log p)^2)$ Multiplikationen modulo p (entsprechend einer logarithmischen Anzahl von schnellen Exponentiationen) auskommt. Dieser Algorithmus benutzt Randomisierung, um einen beliebigen nichtquadratischen Rest zu finden; mit diesem Element wird dann deterministisch weiter gearbeitet.

Bemerkung: Für Zahlen der Form $n = p \cdot q$ für Primzahlen p, q ist kein effizienter Algorithmus zur Ermittlung von Quadratwurzeln modulo n bekannt (auch kein randomisierter!), wenn nur n und die Quadratzahl a (modulo n), nicht aber die Faktoren p und q Teil der Eingabe sind. Diese Tatsache ist die Grundlage der Sicherheit des *Rabin-Kryptosystems*.

Wir fassen unsere Ergebnisse im folgenden Satz zusammen:

Satz 5.48
Für jede ungerade Primzahl p mit $p \equiv 1 \pmod 4$ und einem quadratischen Rest a modulo p berechnet Algorithmus 5.10 die beiden Quadratwurzeln von a. Die erwartete Anzahl von Runden im ersten Teil ist $O(1)$; der erwartete Aufwand ist $O((\log p)^3)$ Bitoperationen.