10

Probabilistic primality testing

In this chapter, we discuss some simple and efficient probabilistic tests for primality.

10.1 Trial division

Suppose we are given an integer $n > 1$, and we want to determine whether n is prime or composite. The simplest algorithm to describe and to program is **trial division**. We simply divide n by 2, 3, and so on, testing if any of these numbers evenly divide n. Of course, we don't need to go any further than $\sqrt{n}$, since if n has any non-trivial factors, it must have one that is no greater than $\sqrt{n}$ (see Exercise 1.1). Not only does this algorithm determine whether n is prime or composite, it also produces a non-trivial factor of n in case n is composite.

Of course, the drawback of this algorithm is that it is terribly inefficient: it requires $\Theta(\sqrt{n})$ arithmetic operations, which is exponential in the binary length of n. Thus, for practical purposes, this algorithm is limited to quite small n. Suppose, for example, that n has 100 decimal digits, and that a computer can perform 1 billion divisions per second (this is much faster than any computer existing today). Then it would take on the order of 10^{33} *years* to perform $\sqrt{n}$ divisions.

In this chapter, we discuss a much faster primality test that allows 100 decimal digit numbers to be tested for primality in less than a second. Unlike the above test, however, this test does not find a factor of n when n is composite. Moreover, the algorithm is probabilistic, and may in fact make a mistake. However, the probability that it makes a mistake can be made so small as to be irrelevant for all practical purposes. Indeed, we can easily make the probability of error as small as 2^{-100} — should one really care about an event that happens with such a miniscule probability?

10.2 The structure of $\mathbb{Z}_n^*$

Before going any further, we have to have a firm understanding of the group $\mathbb{Z}_n^*$, for integer $n > 1$. As we know, $\mathbb{Z}_n^*$ consists of those elements $[a]_n \in \mathbb{Z}_n$ such that a is an integer relatively prime to n. Suppose $n = p_1^{e_1} \cdots p_r^{e_r}$ is the factorization of n into primes. By the Chinese remainder theorem, we have the ring isomorphism

$$\mathbb{Z}_n \cong \mathbb{Z}_{p_1^{e_1}} \times \cdots \times \mathbb{Z}_{p_r^{e_r}}$$

which induces a group isomorphism

$$\mathbb{Z}_n^* \cong \mathbb{Z}_{p_1^{e_1}}^* \times \cdots \times \mathbb{Z}_{p_r^{e_r}}^*.$$

Thus, to determine the structure of the group $\mathbb{Z}_n^*$ for general n, it suffices to determine the structure for $n = p^e$, where p is prime. By Theorem 2.13, we already know the order of the group $\mathbb{Z}_{p^e}^*$, namely, $\phi(p^e) = p^{e-1}(p-1)$.

The main result of this section is the following:

Theorem 10.1. *If p is an odd prime, then for any positive integer e, the group $\mathbb{Z}_{p^e}^*$ is cyclic. The group $\mathbb{Z}_{2^e}^*$ is cyclic for $e = 1$ or 2, but not for $e \geq 3$. For $e \geq 3$, $\mathbb{Z}_{2^e}^*$ is isomorphic to the additive group $\mathbb{Z}_2 \times \mathbb{Z}_{2^{e-2}}$.*

In the case where $e = 1$, this theorem is a special case of Theorem 9.16, which we proved in §9.2.3. Note that for $e > 1$, the ring $\mathbb{Z}_{p^e}$ is *not* a field, and so Theorem 9.16 cannot be used directly. To deal with the case $e > 1$, we need a few simple facts.

Theorem 10.2. *Let p be a prime. For integer $e \geq 1$, if $a \equiv b \pmod{p^e}$, then $a^p \equiv b^p \pmod{p^{e+1}}$.*

Proof. We have $a = b + cp^e$ for some $c \in \mathbb{Z}$. Thus, $a^p = b^p + pb^{p-1}cp^e + dp^{2e}$ for an integer d. It follows that $a^p \equiv b^p \pmod{p^{e+1}}$. □

Theorem 10.3. *Let p be a prime. Let $e \geq 1$ be an integer and assume $p^e > 2$. If $a \equiv 1 + p^e \pmod{p^{e+1}}$, then $a^p \equiv 1 + p^{e+1} \pmod{p^{e+2}}$.*

Proof. By Theorem 10.2, $a^p \equiv (1 + p^e)^p \pmod{p^{e+2}}$. Expanding $(1 + p^e)^p$, we have

$$(1 + p^e)^p = 1 + p \cdot p^e + \sum_{k=2}^{p-1} \binom{p}{k} p^{ek} + p^{ep}.$$

By Exercise 1.12, all of the terms in the sum on k are divisible by p^{1+2e}, and $1 + 2e \geq e + 2$ for all $e \geq 1$. For the term p^{ep}, the assumption that $p^e > 2$ means that either $p \geq 3$ or $e \geq 2$, which implies $ep \geq e + 2$. □

Now consider Theorem 10.1 in the case where p is odd. As we already know that $\mathbb{Z}_p^*$ is cyclic, assume $e > 1$. Let $x \in \mathbb{Z}$ be chosen so that $[x]_p$ generates $\mathbb{Z}_p^*$. Suppose the multiplicative order of $[x]_{p^e} \in \mathbb{Z}_{p^e}^*$ is m. Then as $x^m \equiv 1 \pmod{p^e}$ implies $x^m \equiv 1 \pmod{p}$, it must be the case that $p-1$ divides m, and so $[x^{m/(p-1)}]_{p^e}$ has multiplicative order exactly $p-1$. By Theorem 8.38, if we find an integer y such that $[y]_{p^e}$ has multiplicative order p^{e-1}, then $[x^{m/(p-1)}y]_{p^e}$ has multiplicative order $(p-1)p^{e-1}$, and we are done. We claim that $y := 1 + p$ does the job. Any integer between 0 and $p^e - 1$ can be expressed as an e-digit number in base p; for example, $y = (0 \cdots 0\,1\,1)_p$. If we compute successive pth powers of y modulo p^e, then by Theorem 10.3 we have

$$\begin{aligned}
y \bmod p^e &= (0 \quad \cdots \quad 0\,1\,1)_p, \\
y^p \bmod p^e &= (* \quad \cdots \quad *\,1\,0\,1)_p, \\
y^{p^2} \bmod p^e &= (* \quad \cdots \quad *\,1\,0\,0\,1)_p, \\
&\vdots \\
y^{p^{e-2}} \bmod p^e &= (1\,0 \quad \cdots \quad 0\,1)_p, \\
y^{p^{e-1}} \bmod p^e &= (0 \quad \cdots \quad 0\,1)_p.
\end{aligned}$$

Here, "$*$" indicates an arbitrary digit. From this table of values, it is clear (see Theorem 8.37) that $[y]_{p^e}$ has multiplicative order p^{e-1}. That proves Theorem 10.1 for odd p.

We now prove Theorem 10.1 in the case $p = 2$. For $e = 1$ and $e = 2$, the theorem is easily verified. Suppose $e \geq 3$. Consider the subgroup $G \subseteq \mathbb{Z}_{2^e}^*$ generated by $[5]_{2^e}$. Expressing integers between 0 and $2^e - 1$ as e-digit binary numbers, and applying Theorem 10.3, we have

$$\begin{aligned}
5 \bmod 2^e &= (0 \quad \cdots \quad 0\,1\,0\,1)_2, \\
5^2 \bmod 2^e &= (* \quad \cdots \quad *\,1\,0\,0\,1)_2, \\
&\vdots \\
5^{2^{e-3}} \bmod 2^e &= (1\,0 \quad \cdots \quad 0\,1)_2, \\
5^{2^{e-2}} \bmod 2^e &= (0 \quad \cdots \quad 0\,1)_2.
\end{aligned}$$

So it is clear (see Theorem 8.37) that $[5]_{2^e}$ has multiplicative order 2^{e-2}. We claim that $[-1]_{2^e} \notin G$. If it were, then since it has multiplicative order 2, and since any cyclic group of even order has precisely one element of order 2 (see Theorem 8.31), it must be equal to $[5^{2^{e-3}}]_{2^e}$; however, it is clear from the above calculation that $5^{2^{e-3}} \not\equiv -1 \pmod{2^e}$. Let $H \subseteq \mathbb{Z}_{2^e}^*$ be the subgroup generated by $[-1]_{2^e}$. Then from the above, $G \cap H = \{[1]_{2^e}\}$, and hence by Theorem 8.28, $G \times H$ is isomorphic to the subgroup $G \cdot H$ of $\mathbb{Z}_{2^e}^*$.

But since the orders of $G \times H$ and $\mathbb{Z}_{2^e}^*$ are equal, we must have $G \cdot H = \mathbb{Z}_{2^e}^*$. That proves the theorem.

EXERCISE 10.1. Show that if n is a positive integer, the group $\mathbb{Z}_n^*$ is cyclic if and only if

$$n = 1, 2, 4, p^e, \text{ or } 2p^e,$$

where p is an odd prime and e is a positive integer.

EXERCISE 10.2. Let $n = pq$, where p and q are distinct primes such that $p = 2p' + 1$ and $q = 2q' + 1$, where p' and q' are themselves prime. Show that the subgroup $(\mathbb{Z}_n^*)^2$ of squares is a cyclic group of order $p'q'$.

EXERCISE 10.3. Let $n = pq$, where p and q are distinct primes such that $p \nmid (q-1)$ and $q \nmid (p-1)$.

(a) Show that the map that sends $[a]_n \in \mathbb{Z}_n^*$ to $[a^n]_{n^2} \in (\mathbb{Z}_{n^2}^*)^n$ is a group isomorphism.

(b) Consider the element $\alpha := [1+n]_{n^2} \in \mathbb{Z}_{n^2}^*$; show that for any non-negative integer k, $\alpha^k = [1+kn]_{n^2}$, and conclude that α has multiplicative order n.

(c) Show that the map from $\mathbb{Z}_n \times \mathbb{Z}_n^*$ to $\mathbb{Z}_{n^2}^*$ that sends $([k]_n, [a]_n)$ to $[(1+kn)a^n]_{n^2}$ is a group isomorphism.

10.3 The Miller–Rabin test

We describe in this section a fast (polynomial time) test for primality, known as the **Miller–Rabin test**. The algorithm, however, is probabilistic, and may (with small probability) make a mistake.

We assume for the remainder of this section that the number n we are testing for primality is an odd integer greater than 1.

Several probabilistic primality tests, including the Miller–Rabin test, have the following general structure. Define $\mathbb{Z}_n^+$ to be the set of non-zero elements of $\mathbb{Z}_n$; thus, $|\mathbb{Z}_n^+| = n - 1$, and if n is prime, $\mathbb{Z}_n^+ = \mathbb{Z}_n^*$. Suppose also that we define a set $L_n \subseteq \mathbb{Z}_n^+$ such that:

- there is an efficient algorithm that on input n and $\alpha \in \mathbb{Z}_n^+$, determines if $\alpha \in L_n$;
- if n is prime, then $L_n = \mathbb{Z}_n^*$;
- if n is composite, $|L_n| \leq c(n-1)$ for some constant $c < 1$.

To test n for primality, we set an "error parameter" t, and choose random elements $\alpha_1, \ldots, \alpha_t \in \mathbb{Z}_n^+$. If $\alpha_i \in L_n$ for all $i = 1, \ldots, t$, then we output *true*; otherwise, we output *false*.

It is easy to see that if n is prime, this algorithm always outputs *true*, and if n is composite this algorithm outputs *true* with probability at most c^t. If $c = 1/2$ and t is chosen large enough, say $t = 100$, then the probability that the output is wrong is so small that for all practical purposes, it is "just as good as zero."

We now make a first attempt at defining a suitable set L_n. Let us define

$$L_n := \{\alpha \in \mathbb{Z}_n^+ : \alpha^{n-1} = 1\}.$$

Note that $L_n \subseteq \mathbb{Z}_n^*$, since if $\alpha^{n-1} = 1$, then α has a multiplicative inverse, namely, α^{n-2}. Using a repeated-squaring algorithm, we can test if $\alpha \in L_n$ in time $O(\mathrm{len}(n)^3)$.

Theorem 10.4. *If n is prime, then $L_n = \mathbb{Z}_n^*$. If n is composite and $L_n \subsetneq \mathbb{Z}_n^*$, then $|L_n| \leq (n-1)/2$.*

Proof. Note that L_n is the kernel of the $(n-1)$-power map on $\mathbb{Z}_n^*$, and hence is a subgroup of $\mathbb{Z}_n^*$.

If n is prime, then we know that $\mathbb{Z}_n^*$ is a group of order $n-1$. Since the order of a group element divides the order of the group, we have $\alpha^{n-1} = 1$ for all $\alpha \in \mathbb{Z}_n^*$. That is, $L_n = \mathbb{Z}_n^*$.

Suppose that n is composite and $L_n \subsetneq \mathbb{Z}_n^*$. Since the order of a subgroup divides the order of the group, we have $|\mathbb{Z}_n^*| = m|L_n|$ for some integer $m > 1$. From this, we conclude that

$$|L_n| = \frac{1}{m}|\mathbb{Z}_n^*| \leq \frac{1}{2}|\mathbb{Z}_n^*| \leq \frac{n-1}{2}. \quad \square$$

Unfortunately, there are odd composite numbers n such that $L_n = \mathbb{Z}_n^*$. Such numbers are called **Carmichael numbers**. The smallest Carmichael number is

$$561 = 3 \cdot 11 \cdot 17.$$

Carmichael numbers are extremely rare, but it is known that there are infinitely many of them, so we can not ignore them. The following theorem puts some constraints on Carmichael numbers.

Theorem 10.5. *A Carmichael number n is of the form $n = p_1 \cdots p_r$, where the p_i are distinct primes, $r \geq 3$, and $(p_i - 1) \mid (n-1)$ for $i = 1, \ldots, r$.*

Proof. Let $n = p_1^{e_1} \cdots p_r^{e_r}$ be a Carmichael number. By the Chinese remainder theorem, we have an isomorphism of $\mathbb{Z}_n^*$ with the group

$$\mathbb{Z}_{p_1^{e_1}}^* \times \cdots \times \mathbb{Z}_{p_r^{e_r}}^*,$$

and we know that each group $\mathbb{Z}_{p_i^{e_i}}^*$ is cyclic of order $p_i^{e_i-1}(p_i - 1)$. Thus, the power $n - 1$ kills the group $\mathbb{Z}_n^*$ if and only if it kills all the groups $\mathbb{Z}_{p_i^{e_i}}^*$, which happens if and only if $p_i^{e_i-1}(p_i - 1) \mid (n - 1)$. Now, on the one hand, $n \equiv 0 \pmod{p_i}$. On the other hand, if $e_i > 1$, we would have $n \equiv 1 \pmod{p_i}$, which is clearly impossible. Thus, we must have $e_i = 1$.

It remains to show that $r \geq 3$. Suppose $r = 2$, so that $n = p_1 p_2$. We have

$$n - 1 = p_1 p_2 - 1 = (p_1 - 1)p_2 + (p_2 - 1).$$

Since $(p_1 - 1) \mid (n - 1)$, we must have $(p_1 - 1) \mid (p_2 - 1)$. By a symmetric argument, $(p_2 - 1) \mid (p_1 - 1)$. Hence, $p_1 = p_2$, a contradiction. □

To obtain a good primality test, we need to define a different set L'_n, which we do as follows. Let $n - 1 = 2^h m$, where m is odd (and $h \geq 1$ since n is assumed odd), and define

$$L'_n := \{\alpha \in \mathbb{Z}_n^+ : \ \alpha^{m2^h} = 1 \ \text{ and } \\ \text{for } j = 0, \ldots, h-1, \ \alpha^{m2^{j+1}} = 1 \text{ implies } \alpha^{m2^j} = \pm 1\}.$$

The Miller–Rabin test uses this set L'_n, in place of the set L_n defined above. It is clear from the definition that $L'_n \subseteq L_n$.

Testing whether a given $\alpha \in \mathbb{Z}_n^+$ belongs to L'_n can be done using the following procedure:

$\beta \leftarrow \alpha^m$
if $\beta = 1$ then return *true*
for $j \leftarrow 0$ to $h - 1$ do
    if $\beta = -1$ then return *true*
    if $\beta = +1$ then return *false*
    $\beta \leftarrow \beta^2$
return *false*

It is clear that using a repeated-squaring algorithm, this procedure runs in time $O(\text{len}(n)^3)$. We leave it to the reader to verify that this procedure correctly determines membership in L'_n.

Theorem 10.6. *If n is prime, then $L'_n = \mathbb{Z}_n^*$. If n is composite, then $|L'_n| \leq (n - 1)/4$.*

The rest of this section is devoted to a proof of this theorem. Let $n-1 = m2^h$, where m is odd.

Case 1: n is prime. Let $\alpha \in \mathbb{Z}_n^*$. Since $\mathbb{Z}_n^*$ is a group of order $n-1$, and the order of a group element divides the order of the group, we know that $\alpha^{m2^h} = \alpha^{n-1} = 1$. Now consider any index $j = 0, \ldots, h-1$ such that $\alpha^{m2^{j+1}} = 1$, and consider the value $\beta := \alpha^{m2^j}$. Then since $\beta^2 = \alpha^{m2^{j+1}} = 1$, the only possible choices for β are ± 1—this is because $\mathbb{Z}_n^*$ is cyclic of even order and so there are exactly two elements of $\mathbb{Z}_n^*$ whose multiplicative order divides 2, namely ± 1. So we have shown that $\alpha \in L'_n$.

Case 2: $n = p^e$, where p is prime and $e > 1$. Certainly, L'_n is contained in the kernel K of the $(n-1)$-power map on $\mathbb{Z}_n^*$. By Theorem 8.31, $|K| = \gcd(\phi(n), n-1)$. Since $n = p^e$, we have $\phi(n) = p^{e-1}(p-1)$, and so

$$|L'_n| \leq |K| = \gcd(p^{e-1}(p-1), p^e - 1) = p - 1 = \frac{p^e - 1}{p^{e-1} + \cdots + 1} \leq \frac{n-1}{4}.$$

Case 3: $n = p_1^{e_1} \cdots p_r^{e_r}$ is the prime factorization of n, and $r > 1$. For $i = 1, \ldots, r$, let R_i denote the ring $\mathbb{Z}_{p_i^{e_i}}$, and let

$$\theta : R_1 \times \cdots \times R_r \to \mathbb{Z}_n$$

be the ring isomorphism provided by the Chinese remainder theorem. Also, let $\phi(p_i^{e_i}) = m_i 2^{h_i}$, with m_i odd, for $i = 1, \ldots, r$, and let $\ell := \min\{h, h_1, \ldots, h_r\}$. Note that $\ell \geq 1$, and that each R_i^* is a cyclic group of order $m_i 2^{h_i}$.

We first claim that for any $\alpha \in L'_n$, we have $\alpha^{m2^\ell} = 1$. To prove this, first note that if $\ell = h$, then by definition, $\alpha^{m2^\ell} = 1$, so suppose that $\ell < h$. By way of contradiction, suppose that $\alpha^{m2^\ell} \neq 1$, and let j be the largest index in the range $\ell, \ldots, h-1$ such that $\alpha^{m2^{j+1}} = 1$. By the definition of L'_n, we must have $\alpha^{m2^j} = -1$. Since $\ell < h$, we must have $\ell = h_i$ for some particular index $i = 1, \ldots, r$. Writing $\alpha = \theta(\alpha_1, \ldots, \alpha_r)$, we have $\alpha_i^{m2^j} = -1$. This implies that the multiplicative order of α_i^m is equal to 2^{j+1} (see Theorem 8.37). However, since $j \geq \ell = h_i$, this contradicts the fact that the order of a group element (in this case, α_i^m) must divide the order of the group (in this case, R_i^*).

From the claim in the previous paragraph, and the definition of L'_n, it follows that $\alpha \in L'_n$ implies $\alpha^{m2^{\ell-1}} = \pm 1$. We now consider an experiment in which α is chosen at random from $\mathbb{Z}_n^*$ (that is, with a uniform distribution), and show that $\mathsf{P}[\alpha^{m2^{\ell-1}} = \pm 1] \leq 1/4$, from which the theorem will follow.

Write $\alpha = \theta(\alpha_1, \ldots, \alpha_r)$. As α is uniformly distributed over $\mathbb{Z}_n^*$, each α_i is uniformly distributed over R_i^*, and the collection of all the α_i is a mutually independent collection of random variables.

For $i = 1, \ldots, r$ and $j = 0, \ldots, h$, let $G_i(j)$ denote the image of the $(m2^j)$-power map on R_i^*. By Theorem 8.31, we have

$$|G_i(j)| = \frac{m_i 2^{h_i}}{\gcd(m_i 2^{h_i}, m2^j)}.$$

Because $\ell \leq h$ and $\ell \leq h_i$, a simple calculation shows that

$$|G_i(h)| \text{ divides } |G_i(\ell)| \text{ and } 2|G_i(\ell)| = |G_i(\ell - 1)|.$$

In particular, $|G_i(\ell - 1)|$ is even and is no smaller than $2|G_i(h)|$. The fact that $|G_i(\ell - 1)|$ is even implies that $-1 \in G_i(\ell - 1)$.

The event $\alpha^{m2^{\ell-1}} = \pm 1$ occurs if and only if either

(E_1) $\alpha_i^{m2^{\ell-1}} = 1$ for $i = 1, \ldots, r$, or

(E_2) $\alpha_i^{m2^{\ell-1}} = -1$ for $i = 1, \ldots, r$.

Since the events E_1 and E_2 are disjoint, and since the values $\alpha_i^{m2^{\ell-1}}$ are mutually independent, with each value $\alpha_i^{m2^{\ell-1}}$ uniformly distributed over $G_i(\ell - 1)$ (see part (a) of Exercise 8.22), and since $G_i(\ell - 1)$ contains ± 1, we have

$$\mathsf{P}[\alpha^{m2^{\ell-1}} = \pm 1] = \mathsf{P}[E_1] + \mathsf{P}[E_2] = 2 \prod_{i=1}^{r} \frac{1}{|G_i(\ell - 1)|},$$

and since $|G_i(\ell - 1)| \geq 2|G_i(h)|$, we have

$$\mathsf{P}[\alpha^{m2^{\ell-1}} = \pm 1] \leq 2^{-r+1} \prod_{i=1}^{r} \frac{1}{|G_i(h)|}. \tag{10.1}$$

If $r \geq 3$, then (10.1) directly implies that $\mathsf{P}[\alpha^{m2^{\ell-1}} = \pm 1] \leq 1/4$, and we are done. So suppose that $r = 2$. In this case, Theorem 10.5 implies that n is not a Carmichael number, which implies that for some $i = 1, \ldots, r$, we must have $G_i(h) \neq \{1\}$, and so $|G_i(h)| \geq 2$, and (10.1) again implies that $\mathsf{P}[\alpha^{m2^{\ell-1}} = \pm 1] \leq 1/4$.

That completes the proof of Theorem 10.6.

Exercise 10.4. Show that an integer $n > 1$ is prime if and only if there exists an element in $\mathbb{Z}_n^*$ of multiplicative order $n - 1$.

Exercise 10.5. Let p be a prime. Show that $n := 2p + 1$ is a prime if and only if $2^{n-1} \equiv 1 \pmod{n}$.

EXERCISE 10.6. Here is another primality test that takes as input an odd integer $n > 1$, and a positive integer parameter t. The algorithm chooses $\alpha_1, \ldots, \alpha_t \in \mathbb{Z}_n^+$ at random, and computes

$$\beta_i := \alpha_i^{(n-1)/2} \quad (i = 1, \ldots, t).$$

If $(\beta_1, \ldots, \beta_t)$ is of the form $(\pm 1, \pm 1, \ldots, \pm 1)$, but is not equal to $(1, 1, \ldots, 1)$, the algorithm outputs *true*; otherwise, the algorithm outputs *false*. Show that if n is prime, then the algorithm outputs *false* with probability at most 2^{-t}, and if n is composite, the algorithm outputs *true* with probability at most 2^{-t}.

In the terminology of §7.2, the algorithm in the above exercise is an example of an "Atlantic City" algorithm for the language of prime numbers (or equivalently, the language of composite numbers), while the Miller–Rabin test is an example of a "Monte Carlo" algorithm for the language of *composite* numbers.

10.4 Generating random primes using the Miller–Rabin test

The Miller–Rabin test is the most practical algorithm known for testing primality, and because of this, it is widely used in many applications, especially cryptographic applications where one needs to generate large, random primes (as we saw in §7.8). In this section, we discuss how one uses the Miller–Rabin test in several practically relevant scenarios where one must generate large primes.

10.4.1 Generating a random prime between 2 and M

Suppose one is given an integer $M \geq 2$, and wants to generate a random prime between 2 and M. We can do this by simply picking numbers at random until one of them passes a primality test. We discussed this problem in some detail in §7.5, where we assumed that we had a primality test *IsPrime*. The reader should review §7.5, and §7.5.1 in particular. In this section, we discuss aspects of this problem that are specific to the situation where the Miller–Rabin test is used to implement *IsPrime*.

To be more precise, let us define the following algorithm $MR(n, t)$, which takes as input integers n and t, with $n > 1$ and $t \geq 1$, and runs as follows:

Algorithm $MR(n, t)$**:**

if $n = 2$ then return *true*
if n is even then return *false*

repeat t times
    $\alpha \leftarrow_R \{1, \ldots, n-1\}$
    if $\alpha \notin L'_n$ return *false*

return *true*

So we shall implement *IsPrime*$(\cdot)$ as $MR(\cdot, t)$, where t is an auxiliary parameter. By Theorem 10.6, if n is prime, the output of $MR(n, t)$ is always *true*, while if n is composite, the output is *true* with probability at most 4^{-t}. Thus, this implementation of *IsPrime* satisfies the assumptions in §7.5.1, with $\epsilon = 4^{-t}$.

Let $\gamma(M, t)$ be the probability that the output of Algorithm RP in §7.5—using this implementation of *IsPrime*—is composite. Then as we discussed in §7.5.1,

$$\gamma(M, t) \leq 4^{-t} \frac{M-1}{\pi(M)} = O(4^{-t}k), \tag{10.2}$$

where $k = \mathrm{len}(M)$. Furthermore, if the output of Algorithm RP is prime, then every prime is equally likely; that is, conditioning on the event that the output is prime, the conditional output distribution is uniform over all primes.

Let us now consider the expected running time of Algorithm RP. As was shown in §7.5.1, this is $O(kW'_M)$, where W'_M is the expected running time of *IsPrime* where the average is taken with respect to the random choice of input $n \in \{2, \ldots, M\}$ and the random choices of the primality test itself. Clearly, we have $W'_M = O(tk^3)$, since $MR(n, t)$ executes at most t iterations of the Miller–Rabin test, and each such test takes time $O(k^3)$. This leads to an expected total running time bound of $O(tk^4)$. However, this estimate for W'_M is overly pessimistic. Intuitively, this is because when n is composite, we expect to perform very few Miller–Rabin tests—only when n is prime do we actually perform all t of them. To make a rigorous argument, consider the experiment in which n is chosen at random from $\{2, \ldots, M\}$, and $MR(n, t)$ is executed. Let Y be the number of times the basic Miller–Rabin test is actually executed. Conditioned on any fixed, odd, prime value of n, the value of Y is always t. Conditioned on any fixed, odd, composite value of n, the distribution of Y is geometric with an associated success probability of at least $3/4$; thus, the conditional expectation of Y is at most $4/3$ in this

case. Thus, we have

$$\begin{aligned}\mathsf{E}[Y] &= \mathsf{E}[Y \mid n \text{ prime}]\mathsf{P}[n \text{ prime}] + \mathsf{E}[Y \mid n \text{ composite}]\mathsf{P}[n \text{ composite}] \\ &\leq t\pi(M)/(M-1) + 4/3.\end{aligned}$$

Thus, $\mathsf{E}[Y] \leq 4/3 + O(t/k)$, from which it follows that $W'_M = O(k^3 + tk^2)$, and hence the expected total running time of Algorithm RP is actually $O(k^4 + tk^3)$.

Note that the above estimate (10.2) for $\gamma(M, t)$ is actually quite pessimistic. This is because the error probability 4^{-t} is a worst-case estimate; in fact, for "most" composite integers n, the probability that $MR(n, t)$ outputs *true* is much smaller than this. In fact, $\gamma(M, 1)$ is *very* small for large M. For example, the following is known:

Theorem 10.7. *We have*

$$\gamma(M, 1) \leq \exp[-(1 + o(1)) \log(M) \log(\log(\log(M)))/\log(\log(M))].$$

Proof. Literature—see §10.7. □

The bound in the above theorem goes to zero quite quickly—faster than $(\log M)^{-c}$ for any positive constant c. While the above theorem is asymptotically very good, in practice, one needs explicit bounds. For example, the following *lower* bounds for $-\log_2(\gamma(2^k, 1))$ are known:

k	200	300	400	500	600
	3	19	37	55	74

Given an upper bound on $\gamma(M, 1)$, we can bound $\gamma(M, t)$ for $t \geq 2$ using the following inequality:

$$\gamma(M, t) \leq \frac{\gamma(M, 1)}{1 - \gamma(M, 1)} 4^{-t+1}. \tag{10.3}$$

To prove (10.3), it is not hard to see that on input M, the output distribution of Algorithm RP is the same as that of the following algorithm:

```
repeat
    repeat
        n ←_R {2, ..., M}
    until MR(n, 1)
    n_1 ← n
until MR(n_1, t − 1)
output n_1
```

Consider for a moment a single execution of the outer loop of the above algorithm. Let β be the probability that n_1 is composite, and let α be the conditional probability that $MR(n_1, t-1)$ outputs *true*, given that n_1 is composite. Evidently, $\beta = \gamma(M, 1)$ and $\alpha \leq 4^{-t+1}$.

Now, using *exactly* the same reasoning as was used to derive equation (7.2) in §7.5.1, we find that

$$\gamma(M,t) = \frac{\alpha\beta}{\alpha\beta + (1-\beta)} \leq \frac{\alpha\beta}{1-\beta} \leq \frac{4^{-t+1}\gamma(M,1)}{1-\gamma(M,1)},$$

which proves (10.3).

Given that $\gamma(M, 1)$ is so small, for large M, Algorithm RP actually exhibits the following behavior in practice: it generates a random value $n \in \{2, \ldots, M\}$; if n is odd and composite, then the very *first* iteration of the Miller–Rabin test will detect this with overwhelming probability, and no more iterations of the test are performed on this n; otherwise, if n is prime, the algorithm will perform $t-1$ more iterations of the Miller–Rabin test, "just to make sure."

Exercise 10.7. Consider the problem of generating a random Sophie Germain prime between 2 and M (see §5.5.5). One algorithm to do this is as follows:

```
repeat
    n ←_R {2, ..., M}
    if MR(n, t) then
        if MR(2n + 1, t) then
            output n and halt
forever
```

Assuming Conjecture 5.26, show that this algorithm runs in expected time $O(k^5 + tk^4)$, and outputs a number that is not a Sophie Germain prime with probability $O(4^{-t}k^2)$. As usual, $k := \mathrm{len}(M)$.

Exercise 10.8. Improve the algorithm in the previous exercise, so that under the same assumptions, it runs in expected time $O(k^5 + tk^3)$, and outputs a number that is not a Sophie Germain prime with probability $O(4^{-t}k^2)$, or even better, show that this probability is at most $\gamma(M,t)\pi^*(M)/\pi(M) = O(\gamma(M,t)k)$, where $\pi^*(M)$ is defined as in §5.5.5.

Exercise 10.9. Suppose in Algorithm RFN in §7.7 we implement algorithm *IsPrime*$(\cdot)$ as $MR(\cdot, t)$, where t is a parameter satisfying $4^{-t}(2 + \log M) \leq$

1/2, if M is the input to RFN. Show that the expected running time of Algorithm RFN in this case is $O(k^5 + tk^4 \operatorname{len}(k))$. Hint: use Exercise 7.20.

10.4.2 Trial division up to a small bound

In generating a random prime, most candidates n will in fact be composite, and so it makes sense to cast these out as quickly as possible. Significant efficiency gains can be achieved by testing if a given candidate n is divisible by any small primes up to a given bound s, before we subject n to a Miller–Rabin test. This strategy makes sense, since for a small, "single precision" prime p, we can test if $p \mid n$ essentially in time $O(\operatorname{len}(n))$, while a single iteration of the Miller–Rabin test takes time $O(\operatorname{len}(n)^3)$ steps.

To be more precise, let us define the following algorithm $MRS(n, t, s)$, which takes as input integers n, t, and s, with $n > 1$, $t \geq 1$, and $s > 1$:

Algorithm $MRS(n, t, s)$**:**

for each prime $p \leq s$ do
    if $p \mid n$ then
        if $p = n$ then return *true* else return *false*
repeat t times
    $\alpha \leftarrow_R \{1, \ldots, n-1\}$
    if $\alpha \notin L'_n$ return *false*
return *true*

In an implementation of the above algorithm, one would most likely use the sieve of Eratosthenes (see §5.4) to generate the small primes.

Note that $MRS(n, t, 2)$ is equivalent to $MR(n, t)$. Also, it is clear that the probability that $MRS(n, t, s)$ makes a mistake is no more than the probability that $MR(n, t)$ makes a mistake. Therefore, using MRS in place of MR will not increase the probability that the output of Algorithm RP is a composite—indeed, it is likely that this probability decreases significantly.

Let us now analyze the impact on the running time. To do this, we need to estimate the probability $\tau(M, s)$ that a randomly chosen number between 2 and M is not divisible by any primes up to s. If M is sufficiently large with respect to s, the following heuristic argument can be made rigorous, as we will discuss below. The probability that a random number is divisible by a prime p is about $1/p$, so the probability that it is not divisible by p is about $1 - 1/p$. Assuming that these events are essentially independent for

different values of p (this is the heuristic part), we estimate

$$\tau(M, s) \approx \prod_{p \le s} (1 - 1/p) \sim B_1 / \log s,$$

where $B_1 \approx 0.56146$ is the constant from Exercise 5.14 (see also Theorem 5.21).

Of course, performing the trial division takes some time, so let us also estimate the expected number $\kappa(M, s)$ of trial divisions performed. If $p_1, p_2, \ldots, p_r$ are the primes up to s, then for $i = 1, \ldots, r$, the probability that we perform at least i trial divisions is precisely $\tau(M, p_i - 1)$. From this, it follows (see Theorem 6.8) that

$$\kappa(M, s) = \sum_{p \le s} \tau(M, p - 1) \approx \sum_{p \le s} B_1 / \log p.$$

Using Exercise 5.9 and the Prime number theorem, we obtain

$$\kappa(M, s) \approx \sum_{p \le s} B_1 / \log p \sim B_1 \pi(s) / \log s \sim B_1 s / (\log s)^2.$$

If $k = \operatorname{len}(M)$, for a random $n \in \{2, \ldots, M\}$, the expected amount of time spent within $MRS(n, t, s)$ performing the Miller–Rabin test is now easily seen to be $O(k^3 / \operatorname{len}(s) + tk^2)$. Further, assuming that each individual trial division step takes time $O(\operatorname{len}(n))$, the expected running time of trial division up to s is $O(ks / \operatorname{len}(s)^2)$. This estimate does not take into account the time to generate the small primes using the sieve of Eratosthenes. These values might be pre-computed, in which case this time is zero, but even if we compute them on the fly, this takes time $O(s \operatorname{len}(\operatorname{len}(s)))$, which is dominated by $O(ks / \operatorname{len}(s)^2))$ for any reasonable value of s (in particular, for $s \le k^{O(1)}$).

So provided $s = o(k^2 \operatorname{len}(k))$, the running time of MRS will be dominated by the Miller–Rabin test, which is what we want, of course — if we spend as much time on trial division as the time it would take to perform a single Miller–Rabin test, we might as well just perform the Miller–Rabin test. In practice, one should use a very conservative bound for s, probably no more than k^2, since getting s arbitrarily close to optimal does not really provide that much benefit, while if we choose s too large, it can actually do significant harm.

From the above estimates, we can conclude that with $k \le s \le k^2$, the expected running time W'_M of $MRS(n, t, s)$, with respect to a randomly chosen n between 2 and M, is

$$W'_M = O(k^3 / \operatorname{len}(k) + tk^2). \tag{10.4}$$

From this, it follows that the expected running time of Algorithm RP on input M is $O(k^4/\operatorname{len}(k) + tk^3)$. Thus, we effectively reduce the running time by a factor proportional to $\operatorname{len}(k)$, which is a very real and noticeable improvement in practice.

> The reader may have noticed that in our analysis of *MRS*, we assumed that computing $n \bmod p$ for a "small" prime p takes time $O(\operatorname{len}(n))$. However, if we strictly followed the rules established in Theorem 3.3, we should charge time $O(\operatorname{len}(n)\operatorname{len}(p))$ for this division step. To answer this charge that we have somehow "cheated," we offer the following remarks.
>
> First, in practice the primes p are so small that they surely will fit into a single digit in the underlying representation of integers as vectors of digits, and so estimating the cost as $O(\operatorname{len}(n))$ rather than $O(\operatorname{len}(n)\operatorname{len}(p))$ seems more realistic.
>
> Second, even if one uses the bound $O(\operatorname{len}(n)\operatorname{len}(p))$, one can carry out a similar analysis, obtaining the same result (namely, a speedup by a factor proportional to $\operatorname{len}(k)$) except that one should choose s from a slightly smaller range (namely, $s = o(k^2)$).

As we already mentioned, the above analysis is heuristic, but the results are correct. We shall now discuss how this analysis can be made rigorous; however, we should remark that any such rigorous analysis is mainly of theoretical interest only—in any practical implementation, the optimal choice of the parameter s is best determined by experiment, with the analysis being used only as a rough guide. Now, to make the analysis rigorous, we need prove that the estimate $\tau(M, s) \approx \prod_{p \le s}(1 - 1/p)$ is sufficiently accurate. Proving such estimates takes us into the realm of "sieve theory." The larger M is with respect to s, the easier it is to prove such estimates. We shall prove only the simplest and most naive such estimate, but it is still good enough for our purposes, if we do not care too much about hidden big-O constants.

Before stating any results, let us restate the problem slightly. For real $y \ge 0$, let us call a positive integer "y-rough" if it is not divisible by any prime p up to y. For real $x \ge 0$, let us define $R(x, y)$ to be the number of y-rough integers up to x. Thus, since $\tau(M, s)$ is the probability that a random integer between 2 and M is s-rough, and 1 is by definition s-rough, we have $\tau(M, s) = (R(M, s) - 1)/(M - 1)$.

Theorem 10.8. *For any real $x \ge 0$ and $y \ge 0$, we have*

$$\left| R(x, y) - x \prod_{p \le y}(1 - 1/p) \right| \le 2^{\pi(y)}.$$

Proof. To simplify the notation, we shall use the Möbius function μ (see

§2.6). Also, for a real number u, let us write $u = \lfloor u \rfloor + \{u\}$, where $0 \leq \{u\} < 1$. Let P be the product of the primes up to the bound y.

Now, there are $\lfloor x \rfloor$ positive integers up to x, and of these, for each prime p dividing P, precisely $\lfloor x/p \rfloor$ are divisible by p, for each pair p, p' of distinct primes dividing P, precisely $\lfloor x/pp' \rfloor$ are divisible by pp', and so on. By inclusion/exclusion (see Exercise 6.3), we have

$$R(x,y) = \sum_{d|P} \mu(d)\lfloor x/d \rfloor = \sum_{d|P} \mu(d)(x/d) - \sum_{d|P} \mu(d)\{x/d\}.$$

Moreover,

$$\sum_{d|P} \mu(d)(x/d) = x \sum_{d|P} \mu(d)/d = x \prod_{p \leq y} (1 - 1/p),$$

and

$$\left| \sum_{d|P} \mu(d)\{x/d\} \right| \leq \sum_{d|P} 1 = 2^{\pi(y)}.$$

That proves the theorem. □

This theorem only says something non-trivial when y is quite small. Nevertheless, using Chebyshev's theorem on the density of primes, along with Mertens' theorem, it is not hard to see that this theorem implies that $\tau(M,s) = O(1/\log s)$ when $s = O(\log M \log\log M)$, which implies the estimate (10.4) above. We leave the details as an exercise for the reader.

Exercise 10.10. Prove the claim made above that $\tau(M,s) = O(1/\log s)$ when $s = O(\log M \log\log M)$. More precisely, show that there exist constants c, d, and s_0, such that for all M and d satisfying $s_0 \leq s \leq c \log M \log\log M$, we have $\tau(M,s) \leq d/\log s$. From this, derive the estimate (10.4) above.

Exercise 10.11. Let f be a polynomial with integer coefficients. For real $x \geq 0$ and $y \geq 0$, define $R_f(x,y)$ to be the number of integers m up to x such that $f(m)$ is y-rough. For positive integer M, define $\omega_f(M)$ to be the number of integers $m \in \{0, \ldots, M-1\}$ such that $f(m) \equiv 0 \pmod{M}$. Show that

$$\left| R_f(x,y) - x \prod_{p \leq y} (1 - \omega_f(p)/p) \right| \leq \prod_{p \leq y} (1 + \omega_f(p)).$$

Exercise 10.12. Consider again the problem of generating a random Sophie Germain prime, as discussed in Exercises 10.7 and 10.8. A useful idea is to

first test if *either* n *or* $2n+1$ are divisible by any small primes up to some bound s, before performing any more expensive tests. Using this idea, design and analyze an algorithm that improves the running time of the algorithm in Exercise 10.8 to $O(k^5/\operatorname{len}(k)^2 + tk^3)$—under the same assumptions, and achieving the same error probability bound as in that exercise. Hint: first show that the previous exercise implies that the number of positive integers m up to x such that both m and $2m+1$ are y-rough is at most

$$x \cdot \frac{1}{2} \prod_{2<p\leq y} (1 - 2/p) + 3^{\pi(y)}.$$

Exercise 10.13. Design an algorithm that takes as input a prime q and a bound M, and outputs a random prime p between 2 and M such that $p \equiv 1 \pmod{q}$. Clearly, we need to assume that M is sufficiently large with respect to q. Analyze your algorithm assuming Conjecture 5.24 (and using the result of Exercise 5.22). State how large M must be with respect to q, and under these assumptions, show that your algorithm runs in time $O(k^4/\operatorname{len}(k) + tk^3)$, and that its output is incorrect with probability $O(4^{-t}k)$. As usual, $k := \operatorname{len}(M)$.

10.4.3 Generating a random k-bit prime

In some applications, we want to generate a random prime of fixed size—a random 1024-bit prime, for example. More generally, let us consider the following problem: given integer $k \geq 2$, generate a random k-bit prime, that is, a prime in the interval $[2^{k-1}, 2^k)$.

Bertrand's postulate (Theorem 5.7) implies that there exists a constant $c > 0$ such that $\pi(2^k) - \pi(2^{k-1}) \geq c2^{k-1}/k$ for all $k \geq 2$.

Now let us modify Algorithm RP so that it takes as input integer $k \geq 2$, and repeatedly generates a random n in the interval $\{2^{k-1}, \ldots, 2^k - 1\}$ until *IsPrime*(n) returns *true*. Let us call this variant Algorithm RP$'$. Further, let us implement *IsPrime*$(\cdot)$ as $MR(\cdot, t)$, for some auxiliary parameter t, and define $\gamma'(k,t)$ to be the probability that the output of Algorithm RP$'$—with this implementation of *IsPrime*—is composite.

Then using exactly the same reasoning as above,

$$\gamma'(k,t) \leq 4^{-t} \frac{2^{k-1}}{\pi(2^k) - \pi(2^{k-1})} = O(4^{-t}k).$$

As before, if the output of Algorithm RP$'$ is prime, then every k-bit prime is equally likely, and the expected running time is $O(k^4 + tk^3)$. By doing some trial division as above, this can be reduced to $O(k^4/\operatorname{len}(k) + tk^3)$.

The function $\gamma'(k,t)$ has been studied a good deal; for example, the following is known:

Theorem 10.9. *For all $k \geq 2$, we have*

$$\gamma'(k,1) \leq k^2 4^{2-\sqrt{k}}.$$

Proof. Literature—see §10.7. □

Upper bounds for $\gamma'(k,t)$ for specific values of k and t have been computed. The following table lists some known *lower* bounds for $-\log_2(\gamma'(k,t))$ for various values of k and t:

$t\backslash k$	200	300	400	500	600
1	11	19	37	56	75
2	25	33	46	63	82
3	34	44	55	70	88
4	41	53	63	78	95
5	47	60	72	85	102

Using exactly the same reasoning as the derivation of (10.3), one sees that

$$\gamma'(k,t) \leq \frac{\gamma'(k,1)}{1-\gamma'(k,1)} 4^{-t+1}.$$

10.5 Perfect power testing and prime power factoring

Consider the following problem: we are given a integer $n > 1$, and want to determine if n is a **perfect power**, which means that $n = d^e$ for integers d and e, both greater than 1. Certainly, if such d and e exist, then it must be the case that $2^e \leq n$, so we can try all possible candidate values of e, running from 2 to $\lfloor \log_2 n \rfloor$. For each such candidate value of e, we can test if $n = d^e$ for some d as follows. Suppose n is a k-bit number, that is, $2^{k-1} \leq n < 2^k$. Then $2^{(k-1)/e} \leq n^{1/e} < 2^{k/e}$. So any integer eth root of n must lie in the set $\{u, \ldots, v-1\}$, where $u := 2^{\lfloor (k-1)/e \rfloor}$ and $v := 2^{\lceil k/e \rceil}$. Using u and v as starting values, we can perform a binary search:

```
repeat
    w ← ⌊(u + v)/2⌋
    z ← w^e
    if z = n then
        declare than n = w^e is an a perfect eth power, and stop
    else if z < n then
        u ← w + 1
    else
        v ← w
until u ≥ v
declare that n is not a perfect eth power
```

If $n = d^e$ for some integer d, then the following invariant holds (verify): at the beginning of each loop iteration, we have $u \leq d < v$. Thus, if n is a perfect eth power, this will be discovered. That proves the correctness of the algorithm.

As to its running time, note that with each loop iteration, the length $v - u$ of the search interval decreases by a factor of at least 2 (verify). Therefore, after t iterations the interval will be of length at most $2^{k/e+1}/2^t$, so after at most $k/e + 2$ iterations, the interval will be of length less than 1, and hence of length zero, and the algorithm will halt. So the number of loop iterations is $O(k/e)$. The power w^e computed in each iteration is no more than $2^{(k/e+1)e} = 2^{k+e} \leq 2^{2k}$, and hence can be computed in time $O(k^2)$ (see Exercise 3.22). Hence the overall cost of testing if n is an eth power using this algorithm is $O(k^3/e)$.

Trying all candidate values of e from 1 to $\lfloor \log_2 n \rfloor$ yields an overall running time for perfect power testing of $O(\sum_e k^3/e)$, which is $O(k^3 \operatorname{len}(k))$. To find the largest possible value of e for which n is an eth power, we should examine the candidates from highest to lowest.

Using the above algorithm for perfect power testing and an efficient primality test, we can determine if an integer n is a prime power p^e, and if so, compute p and e: we find the largest positive integer e (possibly 1) such that $n = d^e$ for integer d, and test if d is a prime using an efficient primality test.

10.6 Factoring and computing Euler's phi function

In this section, we use some of the ideas developed to analyze the Miller–Rabin test to prove that the problem of factoring n and the problem of computing $\phi(n)$ are equivalent. By equivalent, we mean that given an effi-

cient algorithm to solve one problem, we can efficiently solve the other, and *vice versa*.

Clearly, one direction is easy: if we can factor n into primes, so

$$n = p_1^{e_1} \cdots p_r^{e_r}, \tag{10.5}$$

then we can simply compute $\phi(n)$ using the formula

$$\phi(n) = p_1^{e_1-1}(p_1 - 1) \cdots p_r^{e_r-1}(p_r - 1).$$

For the other direction, first consider the special case where $n = pq$, for distinct primes p and q. Suppose we are given n and $\phi(n)$, so that we have two equations in the unknowns p and q:

$$n = pq \quad \text{and} \quad \phi(n) = (p-1)(q-1).$$

Substituting n/p for q in the second equation, and simplifying, we obtain

$$p^2 + (\phi(n) - n - 1)p + n,$$

which can be solved using the quadratic formula.

For the general case, it is just as easy to prove a stronger result: given any non-zero multiple of the *exponent* of $\mathbb{Z}_n^*$, we can efficiently factor n. In particular, this will show that we can efficiently factor Carmichael numbers.

Before stating the algorithm in its full generality, we can convey the main idea by considering the special case where $n = pq$, where p and q are distinct primes, with $p \equiv q \equiv 3 \pmod 4$. Suppose we are given such an n, along with $f \neq 0$ that is a common multiple of $p-1$ and $q-1$. The algorithm works as follows: let $f = 2^h m$, where m is odd; choose a random, non-zero element α of $\mathbb{Z}_n$; test if either $\gcd(\mathrm{rep}(\alpha), n)$ or $\gcd(\mathrm{rep}(\alpha^m) + 1, n)$ splits n (recall that $\mathrm{rep}(\alpha)$ denotes the canonical representative of α).

The assumption that $p \equiv 3 \pmod 4$ means that $(p-1)/2$ is an odd integer, and since f is a multiple of $p-1$, it follows that $\gcd(m, p-1) = (p-1)/2$, and hence the image of $\mathbb{Z}_p^*$ under the m-power map is the subgroup of $\mathbb{Z}_p^*$ of order 2, which is $\{\pm 1\}$. Likewise, the image of $\mathbb{Z}_q^*$ under the m-power map is $\{\pm 1\}$. Let $\theta : \mathbb{Z}_p \times \mathbb{Z}_q \to \mathbb{Z}_n$ be the ring isomorphism from the Chinese remainder theorem. Now, if α in the above algorithm does not lie in $\mathbb{Z}_n^*$, then certainly $\gcd(\mathrm{rep}(\alpha), n)$ splits n. Otherwise, condition on the event that $\alpha \in \mathbb{Z}_n^*$. In this conditional probability distribution, α is uniformly distributed over $\mathbb{Z}_n^*$, and $\beta := \alpha^m$ is uniformly distributed over $\theta(\pm 1, \pm 1)$. Let us consider each of these four possibilities:

- $\beta = \theta(1, 1)$ implies $\beta + 1 = \theta(2, 2)$, and so $\gcd(\mathrm{rep}(\beta) + 1, n) = 1$;
- $\beta = \theta(-1, -1)$ implies $\beta + 1 = \theta(0, 0)$, and so $\gcd(\mathrm{rep}(\beta) + 1, n) = n$;

- $\beta = \theta(-1, 1)$ implies $\beta + 1 = \theta(0, 2)$, and so $\gcd(\mathrm{rep}(\beta) + 1, n) = p$;
- $\beta = \theta(1, -1)$ implies $\beta + 1 = \theta(2, 0)$, and so $\gcd(\mathrm{rep}(\beta) + 1, n) = q$.

Thus, if $\beta = \theta(-1, 1)$ or $\beta = \theta(1, -1)$, which happens with probability $1/2$, then $\gcd(\mathrm{rep}(\beta) + 1, n)$ splits n. Therefore, the overall probability that we split n is at least $1/2$.

We now present the algorithm in its full generality. We first introduce some notation; namely, let $\lambda(n)$ denote the exponent of $\mathbb{Z}_n^*$. If the prime factorization of n is as in (10.5), then by the Chinese remainder theorem, we have

$$\lambda(n) = \mathrm{lcm}(\lambda(p_1^{e_1}), \ldots, \lambda(p_r^{e_r})).$$

Moreover, for any prime power p^e, by Theorem 10.1, we have

$$\lambda(p^e) = \begin{cases} p^{e-1}(p-1) & \text{if } p \neq 2 \text{ or } e \leq 2, \\ 2^{e-2} & \text{if } p = 2 \text{ and } e \geq 3. \end{cases}$$

In particular, if $m \mid n$, then $\lambda(m) \mid \lambda(n)$.

Now, returning to our factorization problem, we are given n and a non-zero multiple f of $\lambda(n)$, and want to factor n. We may as well assume that n is odd; otherwise, we can pull out all the factors of 2, obtaining n' such that $n = 2^e n'$, where n' is odd and f is a multiple of $\lambda(n')$, thus, reducing to the odd case.

So now, assume n is odd and f is a multiple of $\lambda(n)$. Assume that f is of the form $f = 2^h m$, where m is odd. Our factoring algorithm, which we describe recursively, runs as follows.

```
if n is a prime power p^e then
        output e copies of p and return
generate a random, non-zero element α of Z_n
d_1 ← gcd(rep(α), n)
if d_1 ≠ 1, then recursively factor d_1 and n/d_1 (using the same f),
        and return
α ← α^m
for j ← 0 to h − 1 do
        d_2 ← gcd(rep(α) + 1, n)
        if d_2 ∉ {1, n}, then recursively factor d_2 and n/d_2
                (using the same f), and return
        α ← α^2
recursively factor n (using the same f)
```

It is clear that when the algorithm terminates, its output consists of the

list of all primes (including duplicates) dividing n, assuming the primality test does not make a mistake.

To analyze the running time of the algorithm, assume that the prime factorization of n is as in (10.5). By the Chinese remainder theorem, we have a ring isomorphism

$$\theta : \mathbb{Z}_{p_1^{e_1}} \times \cdots \times \mathbb{Z}_{p_r^{e_r}} \to \mathbb{Z}_n.$$

Let $\lambda(p_i^{e_i}) = m_i 2^{h_i}$, where m_i is odd, for $i = 1, \ldots, r$, and let $\ell := \max\{h_1, \ldots, h_r\}$. Note that since $\lambda(n) \mid f$, we have $\ell \leq h$.

Consider one execution of the body of the recursive algorithm. If n is a prime power, this will be detected immediately, and the algorithm will return. Here, even if we are using probabilistic primality test, such as the Miller–Rabin test, that always says that a prime is a prime, the algorithm will certainly halt. So assume that n is not a prime power, which means that $r \geq 2$. If the chosen value of α is not in $\mathbb{Z}_n^*$, then d_1 will be a non-trivial divisor of n. Otherwise, conditioning on the event that $\alpha \in \mathbb{Z}_n^*$, the distribution of α is uniform over $\mathbb{Z}_n^*$. Consider the value $\beta := \alpha^{m2^{\ell-1}}$.

We claim that with probability at least $1/2$, $\gcd(\mathrm{rep}(\beta) + 1, n)$ is a non-trivial divisor of n. To prove this claim, let us write

$$\beta = \theta(\beta_1, \ldots, \beta_r),$$

where $\beta_i \in \mathbb{Z}_{p_i^{e_i}}^*$ for $i = 1, \ldots, r$. Note that for those i with $h_i < \ell$, the $m2^{\ell-1}$-power map kills the group $\mathbb{Z}_{p_i^{e_i}}^*$, while for those i with $h_i = \ell$, the image of $\mathbb{Z}_{p_i^{e_i}}^*$ under the $m2^{\ell-1}$-power map is $\{\pm 1\}$. Without loss of generality, assume that the indices i such that $h_i = \ell$ are numbered $1, \ldots, r'$, where $1 \leq r' \leq r$. The values β_i for $i = 1, \ldots, r'$ are uniformly and independently distributed over $\{\pm 1\}$, while for all $i > r'$, $\beta_i = 1$. Thus, the value of $\gcd(\mathrm{rep}(\beta) + 1, n)$ is the product of all prime powers $p_i^{e_i}$, with $\beta_i = -1$, which will be non-trivial unless either (1) all the β_i are 1, or (2) $r' = r$ and all the β_i are -1. Consider two cases. First, if $r' < r$, then only event (1) is possible, and this occurs with probability $2^{-r'} \leq 1/2$. Second, if $r' = r$, then each of events (1) and (2) occurs with probability 2^{-r}, and so the probability that either occurs is $2^{-r+1} \leq 1/2$. That proves the claim.

From the claim, it follows that with probability at least $1/2$, we will obtain a non-trivial divisor d_2 of n when $j = \ell - 1$ (if not before).

So we have shown that with probability at least $1/2$, one execution of the body will succeed in splitting n into non-trivial factors. After at most $\log_2 n$ such successes, we will have completely factored n. Therefore, the expected number of recursive invocations of the algorithm is $O(\mathrm{len}(n))$.

Exercise 10.14. Suppose you are given an integer n of the form $n = pq$, where p and q are distinct, ℓ-bit primes, with $p = 2p' + 1$ and $q = 2q' + 1$, where p' and q' are themselves prime. Suppose that you are also given an integer m such that $\gcd(m, p'q') \neq 1$. Show how to efficiently factor n.

Exercise 10.15. Suppose there is a probabilistic algorithm A that takes as input an integer n of the form $n = pq$, where p and q are distinct, ℓ-bit primes, with $p = 2p' + 1$ and $q = 2q' + 1$, where p' and q' are prime. The algorithm also takes as input $\alpha, \beta \in (\mathbb{Z}_n^*)^2$. It outputs either "failure," or integers x, y, not both zero, such that $\alpha^x \beta^y = 1$. Furthermore, assume that A runs in strict polynomial time, and that for all n of the above form, and for randomly chosen $\alpha, \beta \in (\mathbb{Z}_n^*)^2$, A succeeds in finding x, y as above with probability $\epsilon(n)$. Here, the probability is taken over the random choice of α and β, as well as the random choices made during the execution of A. Show how to use A to construct another probabilistic algorithm A' that takes as input n as above, runs in expected polynomial time, and that satisfies the following property:

> if $\epsilon(n) \geq 0.001$, then A' factors n with probability at least 0.999.

10.7 Notes

The Miller–Rabin test is due to Miller [63] and Rabin [75]. The paper by Miller defined the set L'_n, but did not give a probabilistic analysis. Rather, Miller showed that under a generalization of the Riemann hypothesis, for composite n, the least positive integer a such that $[a]_n \in \mathbb{Z}_n \setminus L'_n$ is at most $O((\log n)^2)$, thus giving rise to a deterministic primality test whose correctness depends on the above unproved hypothesis. The later paper by Rabin re-interprets Miller's result in the context of probabilistic algorithms.

Bach [10] gives an explicit version of Miller's result, showing that under the same assumptions, the least positive integer a such that $[a]_n \in \mathbb{Z}_n \setminus L'_n$ is at most $2(\log n)^2$; more generally, Bach shows the following holds under a generalization of the Riemann hypothesis:

> For any positive integer n, and any proper subgroup $G \subsetneq \mathbb{Z}_n^*$, the least positive integer a such that $[a]_n \in \mathbb{Z}_n \setminus G$ is at most $2(\log n)^2$, and the least positive integer b such that $[b]_n \in \mathbb{Z}_n^* \setminus G$ is at most $3(\log n)^2$.

The first efficient probabilistic primality test was invented by Solovay and Strassen [94] (their paper was actually submitted for publication in 1974).

Later, in Chapter 22, we shall discuss a recently discovered, deterministic, polynomial-time (though not very practical) primality test, whose analysis does not rely on any unproved hypothesis.

Carmichael numbers are named after R. D. Carmichael, who was the first to discuss them, in work published in the early 20th century. Alford, Granville, and Pomerance [7] proved that there are infinitely many Carmichael numbers.

Exercise 10.6 is based on Lehmann [55].

Theorem 10.7, as well as the table of values just below it, are from Kim and Pomerance [53]. In fact, these bounds hold for the weaker test based on L_n.

Our analysis in §10.4.2 is loosely based on a similar analysis in §4.1 of Maurer [61]. Theorem 10.8 and its generalization in Exercise 10.11 are certainly not the best results possible in this area. The general goal of "sieve theory" is to prove useful upper and lower bounds for quantities like $R_f(x, y)$ that hold when y is as large as possible with respect to x. For example, using a technique known as Brun's pure sieve, one can show that for $\log y < \sqrt{\log x}$, there exist β and β', both of absolute value at most 1, such that

$$R_f(x, y) = (1 + \beta e^{-\sqrt{\log x}})x \prod_{p \le y} (1 - \omega_f(p)/p) + \beta' \sqrt{x}.$$

Thus, this gives us very sharp estimates for $R_f(x, y)$ when x tends to infinity, and y is bounded by any fixed polynomial in $\log x$. For a proof of this result, see §2.2 of Halberstam and Richert [42] (the result itself is stated as equation 2.16). Brun's pure sieve is really just the first non-trivial sieve result, developed in the early 20th century; even stronger results, extending the useful range of y (but with larger error terms), have subsequently been proved.

Theorem 10.9, as well as the table of values immediately below it, are from Damgård, Landrock, and Pomerance [32].

The algorithm presented in §10.6 for factoring an integer given a multiple of $\phi(n)$ (or, for that matter, $\lambda(n)$) is essentially due to Miller [63]. However, just as for his primality test, Miller presents his algorithm as a deterministic algorithm, which he analyzes under a generalization of the Riemann hypothesis. The probabilistic version of Miller's factoring algorithm appears to be "folklore."