1 Fermat Pseudoprimes

A primality test is an algorithm which, given an integer \( n \), decides whether \( n \) is prime or not. The most naive algorithm, trial division, is hopelessly inefficient when \( n \) is very large. Fortunately, there exist much more efficient algorithms for determining whether \( n \) is prime. The most common such algorithms are probabilistic; they give the correct answer with very high probability. All efficient primality testing algorithms are based, in one way or another, on Fermat’s Little Theorem.

**Theorem 1.1 (Fermat).** If \( p \) is prime, then for all \( b \in \{1, \ldots, p - 1\} \),

\[
    b^{p-1} \equiv 1 \pmod{p}.
\]

**Definition (Fermat pseudoprime).** Let \( n \geq 2 \) and \( b \in \{1, \ldots, n - 1\} \). We say that the number \( n \) passes the Fermat pseudoprime test at base \( b \) if \( b^{n-1} \equiv 1 \pmod{n} \). A number \( n \) is called a Fermat pseudoprime if it passes the Fermat pseudoprime test for all \( b \in \mathbb{Z}_n^* \).

By Fermat’s Little Theorem, every prime number is a Fermat pseudoprime. Unfortunately, the converse does not hold. There are Fermat pseudoprimes which are not prime. Such numbers are called Carmichael numbers. The first few Carmichael numbers are

\[
    561, 1105, 1729, \ldots
\]

Nevertheless, the notion of a Fermat pseudoprime is a useful notion, not least because there is a very efficient probabilistic algorithm for checking whether a given number \( n \) is a Fermat pseudoprime.

**Proposition 1.2.** If \( n \) is not a Fermat pseudoprime, then \( n \) fails the Fermat pseudoprime test at base \( b \) for at least half of the elements \( b \in \{1, \ldots, n - 1\} \).

**Proof.** Suppose \( n \) is not a Fermat pseudoprime, and let

\[
    G = \{b \in \mathbb{Z}_n \mid b^{n-1} \equiv 1 \pmod{n}\} \subseteq \mathbb{Z}_n^*.
\]

Then \( G \) is a subgroup of \( \mathbb{Z}_n^* \), thus \( |G| \leq |\mathbb{Z}_n^*| \). Since \( n \) is not a Fermat pseudoprime, there exists some \( b \in \mathbb{Z}_n^* \) with \( b \not\in G \), thus \( |G| < |\mathbb{Z}_n^*| \). It follows that \( |G| \leq \frac{1}{2}|\mathbb{Z}_n^*| \leq \frac{n-1}{2} \). Finally, whenever \( b \in \{1, \ldots, n - 1\} \) and \( b \not\in G \), then \( b \) fails the test; there are at least \( \frac{n-1}{2} \) such elements.

**Algorithm 1.3 (Fermat pseudoprime test).**

**Input:** Integers \( n \geq 2 \) and \( t \geq 1 \).

**Output:** If \( n \) is prime, output “yes”. If \( n \) is not a Fermat pseudoprime, output “no” with probability at least \( 1 - 1/2^t \), “yes” with probability at most most \( 1/2^t \).

**Algorithm:** Pick \( t \) independent, uniformly distributed random numbers \( b_1, \ldots, b_t \in \{1, \ldots, n - 1\} \). If \( b_i^{n-1} \equiv 1 \pmod{n} \) for all \( i \), output “yes”, else output “no”.

**Proof.** We prove that the output of the algorithm is as specified. If \( n \) is prime, then the algorithm outputs “yes” by Fermat’s Little Theorem. If \( n \) is not a Fermat pseudoprime, then by Proposition 1.2, \( n \) passes the test at base \( b_i \) with probability at most \( \frac{1}{2} \). Hence the probability that \( n \) passes all \( t \) tests is at most \( 1/2^t \).\qed

Algorithm 1.3 can distinguish prime numbers from non-Fermat pseudoprimes. We did not specify its behavior if the input is a Carmichael number. As a matter of fact, if the input is a Carmichael number, the algorithm will usually output “yes”, but will output “no” with a small probability (namely, when \( n \) has a common prime factor with one of the \( b_i \)).

2 Carmichael numbers

Before describing an improved version of the primality testing algorithm, we prove some useful properties of Carmichael numbers, i.e., non-prime Fermat pseudoprimes.

**Lemma 2.1.** Let \( p^e \) be a prime power with \( e \geq 2 \). Then the group \( \mathbb{Z}_{p^e}^* \) has an element of order \( p \).

**Proof.** Consider \( G = \{1 + p^{e-1}x \mid x \in \mathbb{Z}_{p^e}\} \). Clearly \( G \) is a subgroup of \( \mathbb{Z}_{p^e}^* \) with \( p \) elements. Since \( p \) is prime, each element \( g \in G \) has order 1 or \( p \). The only element of \( G \) of order 1 is 1, hence e.g. \( g = 1 + p^{e-1} \) has order \( p \).\qed

**Proposition 2.2.** Let \( n \) be a Carmichael number. Then \( n \) is odd, and we can factor \( n = m_1 m_2 \), where \( m_1, m_2 \geq 3 \) and \( \gcd(m_1, m_2) = 1 \).
Proof. To show that \( n \) is odd, assume on the contrary that it is even. Then \( n \geq 4 \), since 2 is not a Carmichael number. Moreover, \( n - 1 \) is odd, so we have \((-1)^{n-1} \equiv -1 (\mod n)\). It follows that \( n \) fails the Fermat pseudoprime test at base \( b = 1 \).

To show that \( n \) has the desired factorization, it suffices to show that two distinct primes occur in the prime factorization of \( n \). Since \( n \) is not itself prime, this is equivalent to proving that \( n \) is not of the form \( p^e \), for some prime \( p \) and \( e \geq 2 \). Suppose, for contradiction, that \( n = p^e \). Then, by Lemma 2.1, there is an element \( x \in \mathbb{Z}^*_n \) of order \( p \). Since \( n \) is a Fermat pseudoprime, we also have \( x^{n-1} \equiv 1 (\mod n) \), hence \( p | n - 1 \). But this is impossible since \( p | n \).

\( \Box \)

### 3 Strong Pseudoprimes

**Definition (Strong pseudoprime).** Let \( n \) be odd and write \( n - 1 = 2^r l \), where \( l \) is odd. Given \( b \), compute the following elements of \( \mathbb{Z}^*_n \):

\[
b^l, \ b^{2l}, \ b^{4l}, \ldots, b^{2^{r-1}l}, \ b^{2^rl} = b^{n-1}.
\]

We say that \( n \) passes the strong pseudoprime test at base \( b \) if either \( b^l \equiv 1 (\mod n) \) or \( b^{2^rl} \equiv -1 (\mod n) \) for some \( 0 \leq r < s \).

Note that in the sequence \( b^l, b^{2l}, b^{4l}, \ldots, b^{2^{r-1}l}, b^{2^rl} \), each element is the square of the preceding element. Thus if one of these elements is 1 or \(-1\), then all the following elements are equal to 1.

**Remark 3.1.** If \( n \) passes the strong pseudoprime test at base \( b \), then it also passes the Fermat pseudoprime test at base \( b \). In particular, any strong pseudoprime is a Fermat pseudoprime. Proof: If \( n \) passes the strong pseudoprime test at \( b \), then either \( b^l \equiv 1 (\mod n) \) or \( b^{2^rl} \equiv -1 (\mod n) \) for some \( r < s \). In either case, \( b^{2^rl} \equiv 1 (\mod n) \), and hence \( b^{n-1} \equiv 1 (\mod n) \).

**Remark 3.2.** Any prime is a strong pseudoprime. Proof: If \( n \) is prime, then \( \mathbb{Z}^*_n \) is a field. It follows that the polynomial \( x^{2^r} - 1 \) has at most two roots in \( \mathbb{Z}^*_n \). These roots are \( \pm 1 \). By Fermat’s Little Theorem, \( b^{2^rl} = b^{n-1} = 1 (\mod n) \). If \( b^l \not\equiv 1 (\mod n) \), then let \( r \) be maximal such that \( b^{2^rl} \not\equiv 1 \). Then \((b^{2^rl})^2 = 1\) implies \( b^{2^rl} = -1 \), so \( n \) passes the test at \( b \).

**Proposition 3.3.** If \( n \) is not prime, then \( n \) fails the strong pseudoprime test at base \( b \) for at least half of the elements \( b \in \{1, \ldots, n-1\} \).

**Proof.** Let \( n - 1 = 2^s l \) as before. If \( n \) is not a Fermat pseudoprime, then the result follows from Proposition 1.2 and Remark 3.1. So let us consider the case where \( n \) is a Carmichael number. By Proposition 2.2, we can write \( n = m_1 m_2 \), where \( m_1, m_2 \geq 3 \) and \( \gcd(m_1, m_2) = 1 \). Since \( l \) is odd, we have \((-1)^l \not\equiv 1 (\mod n) \). Let \( r \) be the maximal integer such that there exists some \( b \in \mathbb{Z}^*_n \) with \( b^{2^rl} \not\equiv 1 (\mod n) \). Note that \( 0 \leq r < s \). Let

\[
G = \{ b \in \mathbb{Z}^*_n | b^{2^rl} \equiv \pm 1 (\mod n) \}.
\]

Clearly, \( G \) is a subgroup of \( \mathbb{Z}^*_n \). Hence \( |G| \) divides \( |\mathbb{Z}^*_n| \). We now show that \( G \) is a strict subset of \( \mathbb{Z}^*_n \). By definition of \( r \), there exists some \( b \in \mathbb{Z}^*_n \) with \( b^{2^rl} \not\equiv 1 (\mod n) \). Then either \( b \not\in G \), or else \( b^{2^rl} \equiv -1 (\mod n) \). In the latter case, use the Chinese Remainder Theorem to define \( b' \in \mathbb{Z}^*_n \) such that \( b' \equiv b (\mod m_1) \) and \( b' \equiv 1 (\mod m_2) \). Then \( b^{2^rl} \equiv -1 (\mod m_1) \) and \( b^{2^rl} \equiv 1 (\mod m_2) \). This implies \( b^{2^rl} \not\equiv \pm 1 (\mod n) \), hence \( b' \not\in G \). In either case, \( G \neq \mathbb{Z}^*_n \). Thus, \( |G| < |\mathbb{Z}^*_n| \), hence \( |G| < \frac{1}{2} |\mathbb{Z}^*_n| < \frac{n-1}{2} \).

Finally, we claim that for all \( b \in \{1, \ldots, n-1\} \) with \( b \not\in G \), \( n \) fails the strong pseudoprime test at \( b \). Indeed, either \( b \) is not a unit, in which case \( b^{n-1} \not\equiv 1 (\mod n) \). Or else, \( b^{2^rl} \equiv 1 (\mod n) \) but \( b^{2^rl} \not\equiv \pm 1 (\mod n) \), causing the test to fail. As there are at least \( \frac{n-1}{2} \) elements in \( \{1, \ldots, n-1\} \setminus G \), we are done. \( \Box \)

As a result of Remark 3.2 and Proposition 3.3, we obtain an efficient probabilistic algorithm for primality testing. This algorithm is known as the Miller-Rabin algorithm. Notice that the algorithm is correct for all numbers; there is no equivalent of Carmichael numbers with respect to strong pseudoprimes. A number is a strong pseudoprime if and only if it is prime, which is the case if and only if it passes (with probability as close to 1 as desired) the Miller-Rabin primality test. We finish by summarizing the algorithm:

**Algorithm 3.4 (Miller-Rabin primality test).**

*Input:* Integers \( n \geq 2 \) and \( t \geq 1 \).

*Output:* If \( n \) is prime, output “yes”. If \( n \) is not prime, output “no” with probability at least \( 1 - \frac{1}{2^t} \), and “yes” with probability at most \( \frac{1}{2^t} \).

*Algorithm:* Let \( n - 1 = 2^s l \), where \( l \) is odd. Pick \( t \) independent, uniformly distributed random numbers \( b_1, \ldots, b_t \in \{1, \ldots, n-1\} \). For each \( i \), check that one of the following conditions hold: either \( b_i^l \equiv 1 (\mod n) \) or \( b_i^{2^rl} \equiv -1 (\mod n) \) for some \( 0 \leq r < s \). If this is the case for all \( b_i \), output “yes”, else “no”. \( \Box \)