Université du Luxembourg

Faculté des Sciences de la Technologie
et de la Communication

A thesis submitted in
partial fulfillment for the degree of

Bachelor en sciences
et ingénierie

Primes in P

Student project by
Luca Notarnicola

under the supervision of
Prof. Dr. Gabor Wiese

Luxembourg
Summer semester 2015

Abstract

It is a known result from high school that prime numbers are numbers that only have two divisors, 1 an themselves. But there is much more beyond the theory of prime numbers, which has actually become an important component of number theory. For different purposes, it is of great use to know whether a given number is prime or not, but this task results to be even more difficult, the larger the number is. But the enormous growth of computer science can help us make giant steps forward in this direction.

Contents

List of Algorithms

1 Introduction

Determining whether a given number is a prime number or not is referred to as a *primality test*. The best cryptographers can no longer think prime numbers away since their daily use in cryptography makes it numbers like no other. Efficient cryposystems often make use of prime numbers and the larger[1] these numbers are, the more difficult it is to break the system. But breaking the system would mean having access to all sort of data, and this is why the study of prime numbers is a serious task. One of the most famous cryptosystems is the RSA public key algorithm, developped by Ronald Rivest, Adi Shamir and Leonard Adleman[2] in 1977. The algorithm uses two large prime numbers, which makes it very hard to break. Nowadays many security applications are based on the RSA algorithm.

Many primality tests are known, but whether they are efficient or not is another important question. The efficiency of an algorithm is best measured by its running time, which is a basic component of complexity theory of an algorithm. For instance to check whether an integer n is prime, the standard trial division primality test, known since the time of the ancient Greeks, consists in trying to divide n by every integer k in the range $2 \leq k \leq \sqrt{n}$. If there is any integer k that divides n, then n is composite, otherwise n is prime. In the case that n is prime, it takes $\lfloor \sqrt{n} \rfloor - 1$ steps to see that n is prime and hence results to be extremely slow when applied to large integers n. This primality test is the basis of the famous *Sieve of Erathosthenes*[3] which generates all the prime numbers up to n.

A central result that many primality algorithms use, is Fermat's Little Theorem, which states that for any prime number p, and any integer a relatively prime to p, one has the congruence $a^{p-1} \equiv 1 \pmod{p}$. Based on this formulation, Fermat's primality test suggests to verify the above equality for coprime integers a and n (to be checked for primality and hence playing the role of p). If the equality does not hold, then n cannot be a prime number. On the other hand, if the equality holds, then n is probably prime. *Probably*? Yes, probably since there are composite numbers n that nevertheless satisfy the equality. Such numbers are called *Carmichael numbers*, but we should discuss them later on. Due to this, Fermat's primality test is a *probabilistic* primality test. In contrast to such probabilistic tests, there are *deterministic* tests, which yield proved results.

It has been shown that there are efficient probabilistic primality tests that run in polynomial time, which means that their run time is bounded from above by a polynomial in the size of their input. We say that algorithms of this type of running time belong to the class **P**. They run in time at most $O(n^k)$ for some $k \in \mathbb{N}$ for an input of size n. Denoting by $\mathcal{T}(n)$ the maximum amount of time that takes an algorithm on an input of size n, we obtain that $\mathbf{P} = \cup_{k \geq 1} \mathcal{T}(n^k)$. It can be shown[4] that $O(n \log(n))$ is as well in **P**, where $\log(n)$ stands[5] for the logarithm of n in base 2. It is proven that there are primality tests of which the run time is bounded by a polynomial in $\log(n)$ (meaning that there is some $m \in \mathbb{N}$ such that the run time is bounded by $\log^m(n)$).

[1]By large numbers we mean numbers of more than 600 decimal digits.

[2]Ronald Rivest (born in 1947) is an American crytographer, Adi Shamir (born in 1962) is an Israeli mathematician and cryptographer, and Leonard Adleman (born in 1945) is an American theoretical computer scientist.

[3]Eratosthenes of Cyrene (ca. 284 - ca. 200 BC) was a Greek mathematician.

[4]Using the definition that we recall: Let f, g be two functions. Then $f(n) = O(g(n))$ if and only if there exist two constants n_0 and $C > 0$ such that for all $n \geq n_0 : |f(n)| \leq C|g(n)|$, and the fact that $\log(n) \leq n$ implies that $n \log(n) \leq n^2$.

[5]We will use this notation all the time, instead of explicitely writing $\log_2(n)$.

In this paper, we are going to study in depth two primality tests: The *Miller-Rabin primality test* and the quite recent *AKS primality test*. Both tests are in class **P** and the first one results to be probabilistic. Apart giving the algorithm, we will study their complexity and understand how and why both work. The paper is divided into three main parts. The first part presents the Miller-Rabin primality test, including a pseudocode of the algorithm. A special focus is made on the fact that this is a probabilistic test, finding out how reliable its results are. In a second part, we will expose the AKS primality test, with pseudocode too. Both sections are described and the corresponding theory is worked out. Finally, the last part of the paper gathers possible implementations of both algorithms in SAGE, a free software, that allows to perform all sort of computations and programs. There are more reasons for choosing SAGE to implement our algorithms; as for instance its simplicity. It has many functions already implemented and uses a very basic code (for instance without need to specify delimiters or declare variable types). And that it is free, of course. Included in this part, we also discuss some examples of compilations.

2 The Miller-Rabin primality test

In this section, we are going to present with full details the Miller-Rabin primality test, due first to Gary Lee Miller (1976), an American computer scientist, and later (1980) modified by Michael Oser Rabin, an Israeli computer scientist. The algorithm presented by Miller is a deterministic test, but relies on the unproven *Generalized Riemann Hypothesis*, which we will discuss at some point. Rabin modified this algorithm in order to turn it into a probabilistic test, unconditionally of the generalized Riemann hypothesis. Today the algorithm is linked to both scientists.

Before discussing the algorithm, we present some theory in order to understand it better.

2.1 Preliminaries

In this section we are going through some materia that will occur at several points in the following part.

2.1.1 Fermat's Little Theorem

We start by recalling a famous result, namely Fermat's Little Theorem. We state in the next theorem, which we will prove by using some basic group theory.

Theorem 2.1 (Little Fermat). *Let p be a prime number. Then it holds*

$$a^{p-1} \equiv 1 \pmod{p}. \tag{1}$$

for all $a \in \mathbb{Z}$ with $\gcd(a,p) = 1$.

Proof. Consider the set $G = \{1, 2, \ldots, p-1\}$ which under multiplication modulo p forms a group. For simplicity, we can assume that $1 \leq a \leq p-1$, since $\gcd(a,p) = 1$. This means that $a \in G$. Define

$$k := \operatorname{ord}(a) = \min\{j \in \mathbb{N} : a^j \equiv 1 \pmod{p}\}.$$

Using a corollary of Lagrange's theorem[6], we see that k divides $|G| = p-1$, so that there exists some integer n such that $p-1 = kn$. It follows that

$$a^{p-1} \equiv a^{kn} \equiv (a^k)^n \equiv 1^n \equiv 1 \pmod{p}. \qquad \square$$

Let us view this result on an example.

Example 2.2. Take $p = 7$ and choose $a = 12$ (indeed $\gcd(12,7) = 1$). Then $12^{7-1} \equiv 2985984 \pmod 7 \equiv 426569 \cdot 7 + 1 \equiv 1 \pmod 7$.

This leads us to make the link with *Fermat's primality test*, which is based on this idea. If we want to check whether an integer p is prime, we randomly pick a such that $\gcd(a,p) = 1$. If (1) does not hold, then, by the contrapositive of Theorem 2.1, we are sure that p cannot be prime.

Let us now think about the converse of Theorem 2.1. Let $n \in \mathbb{N}, n \geq 2$. The question that arises reads as follows: replacing p by n in (1), are we ensured that n is prime, whenever (1) holds for some integers a and n? This can be formulated as follows:

$$\big(\forall a \in \mathbb{Z} \text{ with } \gcd(a,n) = 1 \text{ and } a^{n-1} \equiv 1 \pmod{n}\big) \overset{?}{\Longrightarrow} n \text{ is a prime number}$$

[6]We recall the corollary here: Given a finite group G, then the order of any element $a \in G$ divides the cardinality of the group G.

One may think so, but this is wrong. Indeed, there are integers n that satisfy the modular equality $a^{n-1} \equiv 1 \pmod{n}$ for all $a \in \mathbb{Z}$ with $\gcd(a,n) = 1$, but that are not prime. This brings us to define the so called *Carmichael numbers* in the next section.

Due to this argument, we can conclude that Fermat's primality test is only deterministic in the case for composite numbers. But this is not sufficient to state that the whole test is deterministic. Hence this is a probabilistic test, since it does not prove whether the integer given as input is prime or not. Summarizing, we conclude that Fermat's primality test can only prove compositenenss of a number, but not primality.

2.1.2 Carmichael numbers

In this section we will first define *Carmichael numbers*[7] and study some of their most important properties afterwards.

Definition 2.3. Let $n \in \mathbb{N}, n \geq 2$ be composite. Then n is called a *Carmichael number* if $a^{n-1} \equiv 1 \pmod{n}$ for all $a \in \mathbb{Z}$ with $\gcd(a,n) = 1$.

Remark 2.4. Carmichael numbers are sometimes called *Fermat pseudoprimes.* If n satisfies $a^{n-1} \equiv 1 \pmod{n}$ only for special a, then n is called *pseudoprime* in base a. It follows that Carmichal numbers are pseudoprimes in any base a.

Let us see some examples of such integers.

Example 2.5. We are looking for an integer n that satisfies Little Fermat, though it is not prime. A classical example is $n = 561$, since it is the smallest such number. It results that for all $a \in \mathbb{Z}$ with $\gcd(a, 561) = 1$, one has the modular congruence $a^{560} \equiv 1 \pmod{561}$, but $561 = 3 \cdot 11 \cdot 17$ is composite. This does not yet give a formal proof that 561 is a Carmichael number, but we will soon develop a tool to check directly whether a number is a Carmichael number or not. Other Carmichael numbers are for instance $1104, 1728$ or 2464. There are infinitely many.

Definition 2.6. Let $n \in \mathbb{N}$. Then n is called *squarefree* if n is not divisble by the square of any prime number.

Example 2.7. For instance, $238 = 2 \cdot 7 \cdot 17$ is squarefree, but $275 = 5^2 \cdot 11$ is not, since $5^2 \mid 275$.

Let us now start to formulate an important property of Carmichael numbers, which will provide us a characterization of them and thus help us recognize them. We have the following theorem, often referred to as KORSELT's criterion[8].

Theorem 2.8. *Let $n \in \mathbb{N}, n \geq 2$ be composite. Then n is a Carmichael number if and only if n is squarefree and $p \mid n$ implies $(p-1) \mid (n-1)$, for each prime p.*

Proof. ($\Longrightarrow$) Write $n = p_1^{\alpha_1} \cdot \ldots \cdot p_r^{\alpha_r}$, with $p_1, \ldots, p_r$ distict primes and $\alpha_1, \ldots, \alpha_r \geq 1$. We first want to show that n is squarefree, i.e. that $\alpha_i = 1$ for all $1 \leq i \leq r$. For each $1 \leq i \leq r$, let a_i be a primitive root modulo $p_i^{\alpha_i}$.[9] Then, by the Chinese Remainder Theorem, there exists $a \in \mathbb{Z}$ such that $a \equiv a_i \pmod{p_i^{\alpha_i}}$ for all i and such that $\gcd(a,n) = 1$. Therefore, and since we assume that n is a Carmichael number, we have that $a^{n-1} \equiv 1 \pmod{n}$, and thus $a^{n-1} \equiv a_i^{n-1} \equiv 1 \pmod{p_i^{\alpha_i}}$, for all $1 \leq i \leq r$. Moreover, since a_i is a primitive root modulo $p_i^{\alpha_i}$, it has order $\varphi(p_i^{\alpha_i}) = p_i^{\alpha_i - 1}(p_i - 1)$ in $\mathbb{Z}/p_i^{\alpha_i}\mathbb{Z}$, and together with $a_i^{n-1} \equiv 1 \pmod{p_i^{\alpha_i}}$, we obtain that $p_i^{\alpha_i - 1}(p_i - 1)$ must divide $n - 1$, for all i. But since $p_i \mid n$, we

[7]named after Robert Daniel Carmichael (1879 – 1967), an American mathematician.

[8]named after Alwin Reinhold Korselt (1864 – 1947), a German mathematician.

[9]We recall the definition here: g is a primitive root modulo p, if for all $0 < i < p-1$, one has $g^i \not\equiv 1 \pmod{p}$ and $g^{p-1} \equiv 1 \pmod{p}$. Equivalently, one could define g to be a primitive root modulo p, if $p - 1 = \min\{j \in \mathbb{N} : g^j \equiv 1 \pmod{p}\}$, i.e. if g has order $p-1$ in $\mathbb{Z}/p\mathbb{Z}$.

have $\gcd(p_i, n-1) = 1$, and thus $\alpha_i = 1$, for all i, which shows that n is squarefree. Then (since $\alpha_i = 1$) it is also clear that $p_i - 1$ divides $n - 1$.
($\Longleftarrow$) Suppose that $n = p_1 p_2 \cdot \ldots \cdot p_r$ and $(p_i - 1) \mid (n-1)$ for all $1 \leq i \leq r$. So there exist integers k_i such that $n - 1 = k_i(p_i - 1)$, for $1 \leq i \leq r$. We want to see that n is a Carmichael number, i.e. that $a^{n-1} \equiv 1 \pmod{n}$ for $a \in \mathbb{Z}$ coprime to n. The idea is that we first prove that $a^{n-1} \equiv 1 \pmod{p_i}$ for each i. If a is a multiple of p_i, then clearly $a^n \equiv a \pmod{p_i}$. If a is not a multiple of p_i, i.e. $\gcd(a, p_i) = 1$, then by Little Fermat, $a^{p_i - 1} \equiv 1 \pmod{p_i}$. Then

$$a^n \equiv a^{(n-1)+1} \equiv a^{k_i(p_i-1)} a \equiv \left(a^{p_i-1}\right)^{k_i} a \equiv 1^{k_i} a \equiv a \pmod{p_i},$$

which shows that for all i, $p_i \mid (a^n - a)$, and hence $n = p_1 p_2 \cdot \ldots \cdot p_r \mid (a^n - a)$, since n is squarefree. So $a^n \equiv a \pmod{n}$ for all a with $\gcd(a, n) = 1$, thus $a^{n-1} \equiv 1 \pmod{n}$. This achieves the proof. □

Let us come back to Example 2.5. We are now in a position to check that 561 is a Carmichael number using the theorem above. Since 561 is $3 \cdot 11 \cdot 17$ when decomposed in prime factors, we see that all prime factor only occurs once (i.e. with multiplicity 1), which means that 561 is squarefree. Moreover, we note that

$$\begin{array}{rcl}(3-1) \mid (561-1) & \text{since} & 560 = 2 \cdot 280 \\ (11-1) \mid (561-1) & \text{since} & 560 = 10 \cdot 56 \\ (17-1) \mid (561-1) & \text{since} & 560 = 16 \cdot 35,\end{array}$$

which by the theorem above shows that 561 is a Carmichael number.

Corollary 2.9. *Let n be a Carmichael number. Then n is odd and has at least three prime factors.*

Proof. Let n be a Carmichael number. Suppose that n is even, so $n - 1$ is odd. Let $p \mid n$. Then $(p-1) \mid (n-1)$ would imply that only 2 can be a prime factor of n, and since n is squarefree by definition, this means that $n = 2$, thus prime. This contradicts the fact that n is a Carmichael number.
To show that n has at least three prime factors, we write $n = pq$ with p, q two different prime numbers (since n is composite and squarefree), and see that this will lead to a contradiction. Since n is a Carmichael number, we know that $p - 1$ and $q - 1$ both divide

$$n - 1 = pq - 1 = (p-1)q + (q-1) = (q-1)p + (p-1),$$

whence $(p-1) \mid (q-1)$, and $(q-1) \mid (p-1)$, implying that $p = q$, which contradicts the fact that p and q are distinct primes. This means that n must have at least three prime factors. □

2.2 Key points and description of the algorithm

This section serves us as a preparation for the Miller-Rabin primality test. We go through some key points that are used in the algorithm, that we will provide later on.

Square roots of 1. We start by recalling the Chinese Remainder Theorem. Given a prime factorization of n, say $n = p_1^{\alpha_1} \cdot \ldots \cdot p_r^{\alpha_r}$, the theorem induces the ring isomorphism

$$\begin{array}{ccc}\mathbb{Z}/n\mathbb{Z} & \xrightarrow{\simeq} & \mathbb{Z}/p_1^{\alpha_1}\mathbb{Z} \times \cdots \times \mathbb{Z}/p_r^{\alpha_r}\mathbb{Z} \\ a + (n) & \longmapsto & (a + (p_1^{\alpha_1}), \ldots, a + (p_r^{\alpha_r}))\end{array}.$$

We are now interested in finding all the square roots of 1. Let n be an odd integer. Square roots of 1 are solutions of the equation $X^2 \equiv 1 \pmod{n}$, or equivalently of $X^2 - 1 \equiv 0$

(mod n). We now use the factorization that we get by Chinese Remainder Theorem. Since the unit group of $\mathbb{Z}/p_i^{\alpha_i}\mathbb{Z}$ is cyclic[10] for all $1 \leq i \leq r$, it follows that the only solutions are -1 and 1 in $\mathbb{Z}/p_i^{\alpha_i}\mathbb{Z}$. Hence we get a total of $|\{\pm 1\}|^r = 2^r$ solutions in $\mathbb{Z}/n\mathbb{Z}$, namely the elements $(a_1, \dots, a_r)$ with $a_i \in \{-1, 1\}$ for all $1 \leq i \leq r$.

Example 2.10. Let us fnd out all elements of order 2 in $\mathbb{Z}/15\mathbb{Z}$. Note that 15 is odd and hence we can apply the reasoning above. According to what is stated above, the equation $X^2 - 1 \equiv 0 \pmod{15}$ has exactly $2^2 = 4$ solutions since in its prime factorization, 15 is written as $3 \cdot 5$ (ie. with two factors). To see this, we draw a table (see Table 2.1) of squares in $\mathbb{Z}/15\mathbb{Z}$, and we observe that there are indeed 4 solutions, namely $1, 4, 11$ and 14.

x	x^2	x	x^2	x	x^2
0	0	5	10	10	10
1	**1**	6	6	11	**1**
2	4	7	4	12	9
3	9	8	4	13	4
4	**1**	9	6	14	**1**

Table 2.1: Squares in $\mathbb{Z}/15\mathbb{Z}$

This brings us to formulate an easy primality criterion, that we state in the following lemma. The Miller-Rabin primality test is based on a variation of this criterion, which we will soon discover.

Lemma 2.11. *Let $n \in \mathbb{N}$ be squarefree. Then n is prime if and only if the equation $X^2 - 1 \equiv 0 \pmod{n}$ only has solutions $\{-1, 1\}$ in $\mathbb{Z}/n\mathbb{Z}$.*

Proof. It follows from the reasoning above.
($\Longrightarrow$) Assume that n is prime. Then $\mathbb{Z}/n\mathbb{Z}$ is a field and hence an integral domain. It follows that in $\mathbb{Z}/n\mathbb{Z}$, we obtain

$$X^2 - 1 = 0 \iff (X-1)(X+1) = 0 \iff X - 1 = 0 \text{ or } X + 1 = 0,$$

so that the only solutions are -1 and 1.
($\Longleftarrow$) Assume that n is composite and squarefree, say $n = p_1 \cdot \ldots \cdot p_r$ with $r \geq 2$, in its prime factorization (we can assume that there are no exponents since n is squarefree). By the Chinese Remainder Theorem, we have that $\mathbb{Z}/n\mathbb{Z} \simeq \mathbb{Z}/p_1\mathbb{Z} \times \cdots \times \mathbb{Z}/p_r\mathbb{Z}$, which suggests us to consider the elements $(-1, 1, \dots, 1), (1, -1, 1, \dots, 1), 1 = (1, \dots, 1)$ and $-1 = (-1, \dots, -1)$ in $\mathbb{Z}/n\mathbb{Z}$. The squares of these elements are all 1 and thus they are solutions of the equation $X^2 - 1 \equiv 0 \pmod{n}$, which contradicts the fact that its only solutions are 1 and -1. Therefore n must be prime. $\square$

Fast exponentiation. Since good algorithms are those that have minimal running time, it is important to implement fast methods of calculations. In the Miller-Rabin test, as we will see later on, it can occur that large power calculations have to be computed. Let $x \in$

[10]For $p > 2$ prime and $m \in \mathbb{N}, m \geq 1$, there is a group isomorphism $(\mathbb{Z}/p^m\mathbb{Z})^\times \simeq \mathbb{Z}/(p-1)\mathbb{Z} \times \mathbb{Z}/p^{m-1}\mathbb{Z}$. To see this, it suffices to take a generator, say $a + (p)$ in $(\mathbb{Z}/p\mathbb{Z})^\times$ (thus has order $p - 1$) and compute its order inside $(\mathbb{Z}/p^m\mathbb{Z})^\times$, which turns out to be the same. Via the congruence $(1+p)^{p^n} \equiv 1 + p^{n+1} \pmod{p^{n+2}}$, for all $n \in \mathbb{N}, n \geq 0$, one shows that the order of $1+p+(p^m)$ inside $(\mathbb{Z}/p^m\mathbb{Z})^\times$ is p^{m-1}. Finally, since p and p^m are coprime, it follows that the order of $a(1+p)+(p^m)$ is $(p-1)p^{m-1} = \varphi(p^m)$, and thus $(\mathbb{Z}/p^m\mathbb{Z})^\times$ is cyclic. To obtain the factorization in the right hand side of the the group isomorphism, one uses the Chinese Remainder Theorem.

$\mathbb{Z}, n \in \mathbb{N}$ and assume we wish to compute x^n minimalizing the number of multiplications. The idea of the *fast exponentiation method* is to write n in its binary decomposition, say

$$n = \sum_{i=0}^{r} a_i 2^i,$$

with $a_i \in \{0, 1\}$ for each i. We then compute x^n by

$$x^n = x^{(\sum_{i=0}^{r} a_i 2^i)} = (x^{2^0})^{a_0} \cdot \ldots \cdot (x^{2^r})^{a_r} = \prod_{i=0}^{r} (x^{2^i})^{a_i},$$

which takes at most $2r$ multiplications[11]. For instance, since $99 = (1100011)_2$, this method needs at most $2 \cdot 6 = 12$ multiplications in order to compute x^{99}, versus the 98 multiplications that standard multiplication would require.

Towards a probabilistic primality test. Before we describe the criterion onto which the Miller-Rabin algorithm will be based, we should shortly discuss another very simple idea that will arise in the algorithm. It is that all integer n can be decomposed as

$$n = 2^r k \text{ with } r = \max\{j \in \mathbb{N} : 2^j \mid n\} \text{ and } k = \frac{n}{2^r} \text{ is odd.} \tag{2}$$

In this decomposition, r and k are clearly integers, r is the greatest power of 2 in n and k is the odd part in n. For instance, by doing so, 584 is decomposed as $584 = 2^3 \cdot 73$.

We already have encountered two primality criteria, that are Fermat's test and Lemma 2.11. These criteria result to be basic reference points for the Miller-Rabin primality test. Let $n \in \mathbb{N}, n > 1$ a prime number, and decompose $n - 1$ as $2^r k$ accordingly to (2). Let $a \in \mathbb{Z}$ with $\gcd(a, n) = 1$. We conclude by Fermat's Little Theorem that

$$a^{n-1} \equiv a^{2^r k} \equiv 1 \pmod{n}.$$

Now the idea is to work inductively with square roots of 1 and make use of Lemma 2.11. We can write the latter congruence as

$$a^{2^r k} \equiv (a^{2^{r-1} k})^2 \equiv 1 \pmod{n}$$

and therefore by Lemma 2.11, we have that either $a^{2^{r-1}k} \equiv 1 \pmod{n}$ or $a^{2^{r-1}k} \equiv -1 \pmod{n}$ if n is prime. Taking square root in the first case, we get again two possibilities, that are either $a^{2^{r-2}k} \equiv 1 \pmod{n}$ or $a^{2^{r-2}k} \equiv -1 \pmod{n}$. We see that continuing on this way, if we always get as result 1 when reducing modulo n, then we come to $a^{2^0 k} \equiv a^k \equiv 1 \pmod{n}$. Note that the sequence of numbers

$$a^{2^0 k} = a^k, a^{2^1 k} = a^{2k}, \ldots, a^{2^{r-2}k}, a^{2^{r-1}k}, \tag{3}$$

is very simple to construct, in the sense that any term (except for the first one) of the sequence is the square of the previous term.

This reasoning suggests a good idea for primality testing: Given an input n, we first decompose it as in (2). Then we can randomly pick an integer a relatively prime to n and check whether either $a^k \equiv 1 \pmod{n}$ or $a^{2^i k} \equiv -1 \pmod{n}$, for some $0 \leq i \leq r - 1$.

However, it results that this is a probabilistic test, in the sense that it cannot prove primality. Clearly, the only case that is proven is when n is composite, which occurs when any of the two conditions fail. We should later on focus on the question of how sure we are of the output. As we already mentioned, the test could be turned into a deterministic version, under the assumption of the Generalized Riemann Hypothesis. Roughly spoken, this consists in choosing the values of a in an interval bounded by a polynomial in $\log(n)$. This makes the test run in polynomial time.

[11]The product has exactly $r+1$ factors $(x^{2^i})^{a_i}, i = 1\ldots, r$. In the worst case, when all the terms appear, we need r multiplications to multiply the $r+1$ factors together. Then to compute each of the x^{2^i}, for $i = 1, \ldots r$, we need r more multiplications. Thus a total of $2r$ multiplications.

2.3 The algorithm

After having studied the main ingredients, we are now in a position to present the algorithm. We will not assume the Generalized Riemann hypothesis, and hence present the probabilistic version of the test. In order to simplify, we give a pseudocode of it, and discuss its complexity afterwards.

Algorithm 2.1 Miller-Rabin primality test

```
Require: $n, t \in \mathbb{N}, n \geq 3$
Ensure: $n$ is Probably Prime or $n$ is Composite
 1: function MILLER-RABIN($n, t$)
 2:     if $n$ is a multiple of $2, 3$ or $5$ then
 3:         return Composite
 4:     end if
 5:     Decompose $n - 1 = 2^r k$ with $r, k \in \mathbb{N}$, and $k$ odd
 6:     for $i = 1$ to $t$ do
 7:         Choose $a \in (\mathbb{Z}/n\mathbb{Z})^\times$ randomly
 8:         Compute $b = a^k \mod n$
 9:         if $b \neq 1$ then
10:             $j \leftarrow 0$
11:             while $j \leq r - 1$ and $b \neq -1$ do
12:                 $b \leftarrow b^2 \mod n$
13:                 $j \leftarrow j + 1$
14:             end while
15:             if $b \neq -1$ then
16:                 return Composite
17:             end if
18:         end if
19:     end for
20:     return Probably Prime
21: end function
```

Main steps. Let us analyze and describe the main steps of this algorithm. One may ask what role the input t has in the algorithm. This is just a bound for the number of a's we want to consider in $(\mathbb{Z}/n\mathbb{Z})^\times$ (see line 7). One could also decide not to give this value as input, but woluld then need to run the program t times, which is not as comfortable.

step	lines	description
I	$2-4$	Clearly, the input n is composite if it is divisible by $2, 3$ or 5.
II	5	The algorithm decomposes $n-1$ accordingly to (2).
III	$6-8$	All what follows is repeated t times, where t is the bound chosen in the input. Firstly, the algorithm chooses at rondom a number in $(\mathbb{Z}/n\mathbb{Z})^\times$ and raises it to the power k by fast exponentiation method, and is then reduced modulo n. This is the content of b.
IV	$9-20$	This step is the main test. If b computed above is not equal to 1, then we have to see whether there is some $0 \leq j \leq r-1$ so that $a^{2^j k}$ equals -1 modulo n. This sequence is set up by squaring previous terms and incrementing j. If no such j is detected, then n is composite. If $a^k \mod n$ equals 1, then n is likely prime.

2.4 Correctness and complexity

First of all we will argue why the algorithm works, using part of the theory we studied so far, whereas later, we will discuss time complexity of the algorithm.

2.4.1 Correctness of the algorithm

We start by proving the main argument of the primality test, that is the following proposition.

Proposition 2.12. *Let $n \in \mathbb{N}$, $n \geq 3$. Let the decomposition $n-1 = 2^r k$ with $r, k \in \mathbb{N}$ and k odd. Then the following statements are equivalent.*

(i) n is a prime number

(ii) For all $a \in (\mathbb{Z}/n\mathbb{Z})^\times$, one of the following conditions holds:

(α) $a^k \equiv 1 \pmod{n}$

(β) $\exists\, i \in \{0, \dots, r-1\}$ such that $a^{2^i k} \equiv -1 \pmod{n}$

Proof. ($\Longrightarrow$) This part is based on Fermat's Little Theorem. Since, by assumption n is prime, and a is in $(\mathbb{Z}/n\mathbb{Z})^\times$, the conditions of Theorem 2.1 are fulfilled. Thus, $a^{n-1} \equiv 1 \pmod{n}$. Given the decomposition of $n-1$ as $2^r k$, we obtain that

$$a^{n-1} \equiv a^{2^r k} \equiv \left(a^{2^{r-1}k}\right)^2 \equiv 1 \pmod{n}. \tag{4}$$

By Lemma 2.11, we see that $a^{2^{r-1}k} \equiv \pm 1 \pmod{n}$. Let us discuss both cases separately. Assume first that $a^{2^{r-1}k} \equiv -1 \pmod{n}$. Then condition ($\beta$) holds. In the other case, let us suppose that $a^{2^{r-1}k} \equiv 1 \pmod{n}$. By proceeding exactly as in (4), we get that $\left(a^{2^{r-2}k}\right)^2 \equiv \pm 1 \pmod{n}$. Again, if $\left(a^{2^{r-2}k}\right)^2 \equiv -1 \pmod{n}$, we are left with condition (β). Assuming that we never get a congruence to -1, we repeat the procedure in (4) until we obtain $a^{2^0 k} \equiv a^k \equiv 1 \pmod{n}$, which is condition ($\alpha$).
($\Longleftarrow$) Instead of proving this implication, we show its contrapositive. Therefore, assume that n is composite, say $n = p_1 \cdot \ldots \cdot p_m$ with p_i prime numbers for $i = 1, \dots, m$. Let $a \in (\mathbb{Z}/n\mathbb{Z})^\times$. We will now see that $a^{n-1} \equiv 1 \pmod{n}$ whenever (α) or (β) hold. Indeed, if (α) holds, then

$$a^{n-1} \equiv a^{2^r k} \equiv \left(a^k\right)^{2^r} \equiv 1^{2^r} \equiv 1 \pmod{n},$$

and if (β) holds, then there exists $i \in \{0, \dots, r-1\}$ such that $a^{2^i k} \equiv -1 \pmod{n}$. By repeated squarings we get that $a^{2^r k} \equiv a^{n-1} \equiv 1 \pmod{n}$. Since n is not prime but still satisfies $a^{n-1} \equiv 1 \pmod{n}$ with $a \in (\mathbb{Z}/n\mathbb{Z})^\times$, n must be a Carmichael number. Then by Corollary 2.9, it follows that n has at least three odd prime factors, i.e. that $m \geq 3$. By the Chinese Remainder Theorem, we have the ring isomorphism $\mathbb{Z}/n\mathbb{Z} \simeq \mathbb{Z}/p_1\mathbb{Z} \times \cdots \times \mathbb{Z}/p_m\mathbb{Z}$. Let us now show that ($\alpha$) and ($\beta$) cannot hold, by exhibiting an $a \in (\mathbb{Z}/n\mathbb{Z})^\times$ that does not fit any of the conditions. By the isomporphism, we choose $a = (a_1, \dots, a_m)$ with at least one a_i equal to -1, say a_1, and at least one a_i equal to 1, say a_2, for simplicity. Then a looks like $(-1, 1, \star, \dots, \star)$. Since k is odd in the factorization of $n-1$, we immediately see that $a^k \not\equiv 1 \pmod{n}$ because ${a_1}^k \equiv -1 \pmod{n}$. Hence ($\alpha$) cannot be satisfied. Analogously, it can never occur that $a^{2^r k} \equiv -1 \pmod{n}$, since for any exponent j, we will have ${a_2}^j \equiv 1 \pmod{n}$. So ($\beta$) fails as well, and hence (ii) is not satisfied. $\square$

Remark 2.13. We note that in (β), the sequence built up with i running through integers 1 up to $r-1$ is set up such that each element is the square of its previous element (except for the first one).

Let us apply this to a concrete example.

Example 2.14. Let us check the first implication of this criterion on the example $n = 7$. We write $6 = 2 \cdot 3$, so that $r = 1$ and $k = 3$. Since 7 is prime, we need to check that for all $a \in (\mathbb{Z}/7\mathbb{Z})^\times = \{0, \ldots, 6\}$, either $a^3 \equiv 1 \pmod 7$, or $a^{2^0 \cdot 3} = a^3 \equiv -1 \pmod 7$ (since $i \in \{1, \ldots, r-1\}$, we have the only case $i = 0$). Indeed, for $a \in \{1, 2, 4\}$, we have that $a^3 \equiv 1 \pmod 7$, whereas for $a \in \{3, 5, 6\}, a^3 \equiv -1 \pmod 7$. Hence one of the conditions holds for all $a \in (\mathbb{Z}/7\mathbb{Z})^\times$.

Let us now move on to the question related to the probability estimate of a correct output.

How sure are we about the output? Let $n \in \mathbb{N}$. The only case where the correctness of the output is guaranteed, is the case when *(ii)* is not satisfied for any of its conditions. In this case we know for sure that n must be composite. But assuming that we are not in this case, we now focus on the question of how sure we are about the result. If n is not prime, there could exist some $a \in (\mathbb{Z}/n\mathbb{Z})^\times$ such that condition *(ii)* is satisfied anyway. Put another way, there exist some values of a that fulfill condition *(ii)* without n being prime. Thus, for composite n, it is convenient to define the set

$$A_n := \left\{a \in (\mathbb{Z}/n\mathbb{Z})^\times : (\alpha) \text{ or } (\beta) \text{ holds}\right\}, \tag{5}$$

that is the set

$$\left\{a \in (\mathbb{Z}/n\mathbb{Z})^\times : a^k \equiv 1 \pmod n \text{ or } \exists\ i \in \{0, \ldots, r-1\}, a^{2^i k} \equiv -1 \pmod n\right\},$$

where k (odd) and r are the integers in the decomposition $n - 1 = 2^r k$. The next proposition will lead us towards an answer to this question, by estimating the size of A_n, i.e. the number of a's that satisfy both conditions although n is composite.

Proposition 2.15. *Let $n \in \mathbb{N}$ be composite such that n is not divisible by 2 and 3. Write the decomposition $n - 1 = 2^r k$ with $r, k \in \mathbb{N}$ and k odd. Define A_n as in (5). Then*

$$|A_n| \leq \frac{\varphi(n)}{4},$$

where φ denotes Euler's totient function[12].

Proof. We consider the prime factorization $n = p_1^{\alpha_1} \cdot \ldots \cdot p_m^{\alpha_m}$ with $p_1, \ldots, p_m$ distinct odd primes. Let $0 \leq s \leq r-1$ denote the maximum such that there exists some $a \in (\mathbb{Z}/n\mathbb{Z}^\times)$ that satisfies $a^{2^s k} \equiv -1 \pmod n$ (thus $a^{2^{s+1}k} \equiv 1 \pmod n$). From this it follows that for all $1 \leq i \leq m$, one has that $2^{s+1} \mid (p_i - 1)$. Indeed $a^{2^s k} \equiv -1 \pmod n$ implies $a^{2^s k} \equiv -1 \pmod{p_i}$, and therefore $\operatorname{ord}(a^k) = 2^{s+1}$ in $\mathbb{Z}/p_i\mathbb{Z}^\times$ (by construction of s). Therefore 2^{s+1} divides the order of $\mathbb{Z}/p_i\mathbb{Z}^\times$, which is $p_i - 1$.
Let us now consider the composite group homomorphism $\psi = \pi \circ f$ defined by

$$\begin{array}{rccccc} \psi: & (\mathbb{Z}/n\mathbb{Z})^\times & \xrightarrow{f} & (\mathbb{Z}/n\mathbb{Z})^\times & \xrightarrow{\pi} & (\mathbb{Z}/n\mathbb{Z})^\times/\{\pm 1\}, \\ & a & \longmapsto & a^{2^s k} & \longmapsto & a^{2^s k} \mod \{\pm 1\} \end{array}$$

where π represents the natural projection. Then $\ker \psi$ is the set of all elements $a \in (\mathbb{Z}/n\mathbb{Z})^\times$ such that $\psi(a) = \pi(f(a)) = \pi(a^{2^s k}) = \pm 1$, which shows that $A_n \subset \ker \psi$. So in order to bound the cardinality of A_n, it suffices to find a bound for the cardinality of $\ker \psi$. This is done by giving a lower bound of the cardinality of $\operatorname{im}\psi$. By the Chinese Remainder

[12]We briefly recall Euler's totient function. Let $n \in \mathbb{N}$, then $\varphi(n)$ counts the positive integers less or equal to n that are coprime with n. An important consequence for us is the fact that $\varphi(n)$ counts units in the group $\mathbb{Z}/n\mathbb{Z}$, ie. $\varphi(n) = |(\mathbb{Z}/n\mathbb{Z})^\times|$.

Theorem, we have the factorization $\mathbb{Z}/n\mathbb{Z} \simeq \mathbb{Z}/p_1^{\alpha_1}\mathbb{Z} \times \cdots \times \mathbb{Z}/p_m^{\alpha_m}\mathbb{Z}$, and we can define the element

$$w_i := (1, \ldots, 1, g_i, 1, \ldots, 1) \in (\mathbb{Z}/n\mathbb{Z})^\times,$$

where $g_i \in (\mathbb{Z}/p_i^{\alpha_i}\mathbb{Z})^\times$ is a generator for $1 \leq i \leq m$, since the groups $\mathbb{Z}/p_i^{\alpha_i}\mathbb{Z}$ are all cyclic. From this definition, it clearly follows that w_i and g_i have same order. We should now discuss different possible cases for the value of m.

(i) If $m = 1$, then $n = p_1^{\alpha_1}$ and thus $\alpha_1 > 1$, otherwise n would be prime. Since n is not divisible by 2 and 3, we have that $p_1 \geq 5$. We observe that $p_1 \mid \operatorname{ord}(g_1) = \operatorname{ord}(w_1)$. But $p_1 \nmid k$ (since $p_1 \nmid (n-1) = p_1^{\alpha_1} - 1$), hence $p_1 \mid \operatorname{ord}(w_1^{2^s k}) = \operatorname{ord}(\psi(w_1))$. Thus $\operatorname{ord}(\psi(w_1)) \geq p_1 \geq 5 > 4$, which gives $|\mathrm{im}\psi| \geq 4$.

(ii) If $m = 2$, then $n = p_1^{\alpha_1} p_2^{\alpha_2}$. Again, for $i = 1, 2$, we see that $p_i \nmid (n-1)$. Let us first assume that there is $j \in \{1, 2\}$ such that $\alpha_j > 1$, then again $p_j \mid \operatorname{ord}(w_j^{2^s k}) = \operatorname{ord}(g_j^{2^s k})$, and consequently $\operatorname{ord}(w_j^{2^s k}) \geq p_j \geq 5 > 4$. So we get again that $|\mathrm{im}\psi| \geq 4$. Assume now that $\alpha_1 = \alpha_2 = 1$, so that $n = p_1 p_2$. For $i = 1, 2$, we decompose $p_i - 1 = 2^{r_i} k_i$, with $r_i, k_i \in \mathbb{N}$ and k_i odd. We observe that

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

and substituting $p_i = 2^{r_i} k_i$ for $i = 1, 2$,

$$\begin{aligned} n - 1 = 2^r k &= 2^{r_1} k_1 \cdot 2^{r_2} k_2 + 2^{r_1} k_1 + 2^{r_2} k_2 \\ &= 2^{r_1 + r_2} k_1 k_2 + 2^{r_1} k_1 + 2^{r_2} k_2. \end{aligned} \tag{6}$$

Let us suppose that both $k_i \mid k$, for $i = 1, 2$. Since $k_1 \mid k$, we get that $k_1 \mid 2^r k$, and by (6) that $k_1 \mid (2^{r_1+r_2} k_1 k_2 + 2^{r_1} k_1 + 2^{r_2} k_2)$. It follows that $k_1 \mid k_2$. By symmetry, since we additionnally assume that $k_2 \mid k$, we obtain that also $k_2 \mid k_1$, implying that $k_1 = k_2$. Hence (6) can be rewritten as

$$2^r k = 2^{r_1+r_2} {k_1}^2 + (2^{r_1} + 2^{r_2}) k_1.$$

Dividing both sides by $k_1 \geq 2$, and assuming, without loss of generality, that $r_1 > r_2$, we find

$$2^r \frac{k}{k_1} = 2^{r_1+r_2} k_1 + 2^{r_1} + 2^{r_2} = 2^{r_2}(2^{r_1} k_1 + 2^{r_1 - r_2} + 1).$$

Since the number $2^{r_1} k_1 + 2^{r_1 - r_2} + 1$ is odd, we conclude, from the above, that $r = r_2$, and consequently we get the estimation $k/k_1 \geq 4 + 2 + 1 = 7$, whence $\operatorname{ord}(w_1^{2^s k}) = \operatorname{ord}(g_1^{2^s k}) \geq 7$.
We now treat the case where for $i = 1, 2$, one of the $k_i \nmid k$, say k_1. The other case is analogous. Then we trivially observe that $\gcd(k_1, k) < k_1$ and since both k and k_1 are odd, $\gcd(k_1, k) = k_1/a$ for some $a \geq 3$, so that $\gcd(k_1, k) \leq k_1/3$. From the formula $\operatorname{ord}(g^z) = \frac{\operatorname{ord}(g)}{\gcd(\operatorname{ord}(g), z)}$[13], it follows that $\operatorname{ord}(w_1^{2^s k}) = \operatorname{ord}(g_1^{2^s k}) = \frac{\operatorname{ord}(g_1^{2^s})}{\gcd(\operatorname{ord}(g_1^{2^s}), k)} \geq 6$, since $2k_1 \mid \operatorname{ord}(g_1^{2^s k})$ because it divides the numerator but not the denominator of $\operatorname{ord}(g_1^{2^s k})$. Therefore $\operatorname{ord}(g_1^{2^s k}) \geq 2k_1 \geq 6$ since $k_1 \geq 3$ by the fact that $k_1 \geq 2$ is odd. Now since $\operatorname{ord}(w_2^{2^s k}) = \operatorname{ord}(g_2^{2^s k}) \geq 2$ (again as before, by construction of w_i), the elements $w_1^{2^s k}$ and $w_2^{2^s k}$ generate a subgroup of order at least $12 = 2 \cdot 6$ of $(\mathbb{Z}/n\mathbb{Z})^\times$, and since the projection map π in ψ can remove at most a factor of 2, we conclude that $|\mathrm{im}\psi| \geq 12/2 = 6$.

[13]This gives the order of an element g raised to the power z. Indeed, denoting $a = \operatorname{ord}(g)$ and $b = \operatorname{ord}(g^z)$, it suffices to see that $\langle g^z \rangle \subset \langle g \rangle$ and thus by Lagrange's Theorem, $b \mid a$, hence there exists $\alpha \in \mathbb{Z}^*$ such that $b = a/\alpha$ and thus $\alpha \mid a = \operatorname{ord}(g)$. But since $e = g^{zb} = g^{za/\alpha}$ (e denotes the neutral element in the group) and since $a = \operatorname{ord}(g)$, a must divide za/α. Therefore, since α divides both a and z, $\alpha \mid \gcd(a, z)$, and a calculation shows that $\alpha = \gcd(a, z)$ is convenient.

(iii) If $m \geq 3$, then the elements $w_i^{2^s k}$ have order at least 2 (as before, by their construction) for $1 \leq i \leq m$. These elements generate a subgroup of order at least 2^m of $(\mathbb{Z}/n\mathbb{Z})^\times$. The projection map π in ψ can at most remove a factor of 2, so that $|\text{im}\psi| \geq 2^{m-1} \geq 4$, since $m \geq 3$.

Putting all cases together, we get that $|\text{im}\psi| \geq 4$. By the isomorpism theorem, we obtain $(\mathbb{Z}/n\mathbb{Z})^\times / \ker \psi \simeq \text{im}\psi$, and thus they have the same cardinality, that is

$$\frac{|(\mathbb{Z}/n\mathbb{Z})^\times|}{|\ker \psi|} = \frac{\varphi(n)}{|\ker \psi|} = |\text{im}\psi| \geq 4 \Longrightarrow |A_n| \leq |\ker \psi| \leq \frac{\varphi(n)}{4},$$

what we wanted to show. □

Interpretation. Roughly spoken, Proposition 2.15 tells us that each fourth value of a in $(\mathbb{Z}/n\mathbb{Z})^\times$ could lead to a wrong output. Hence we see that the more values of a we consider, the more sure we are about the output, and in this way, the probability of an incorrect output can become extremely small. The probability that n is composite when a satisfies one of both conditions, is less than $1/4$. So, if we repeat the test t times (picking t different values for a in $(\mathbb{Z}/n\mathbb{Z})^\times$), we can make this probability less than $(1/4)^t$. Consequently, if t tends to $\varphi(n)$, and n is large, then this probability tends to 0.

The Generalized Riemann Hypothesis. In 1976, Miller proved that the algorithm could be turned into a *deterministic* primality test, under the assumption of the *Generalized Riemann Hypothesis* (GRH), that we will not prove[14]. Four years later (1980), Rabin reformulated the test in order to avoid the Generalized Riemann Hypothesis. The next proposition (without proof) gives the analogue to the implication $(ii) \implies (i)$ of Proposition 2.12, under the assumption of GRH.

Proposition 2.16 (assuming GRH). *Let $n \in \mathbb{N}$ odd with the decomposition $n - 1 = 2^r k$ for $r, k \in \mathbb{N}$ and k odd. If for all integers $1 < a < 2(\log_2 n)^2$, one either has*

(α) $a^k \equiv 1 \pmod{n}$ *or*

(β) $\exists i \in \{0, \ldots, r-1\}$ *such that* $a^{2^i k} \equiv -1 \pmod{n}$,

then n is a prime number.

We observe the slight difference with the criterion seen in Proposition 2.12; instead of considering random integers $a \in (\mathbb{Z}/n\mathbb{Z})^\times$, we consider $a \in \mathbb{N}$ satisfying $1 < a < 2(\log_2 n)^2$. Taking hold of this assumption, the Miller-Rabin test will actually prove that n is prime.

Now that we have stated and proved the latter propositions, we can formulate the following theorem and prove correctness of Algorithm 2.1.

Theorem 2.17. *Let $n \in \mathbb{N}$ with $n \geq 3$ odd.*

(*i*) *If* Miller-Rabin(n, t) *returns* `Composite`*, then n is composite.*

(*ii*) *If* Miller-Rabin(n, t) *returns* `Probably Prime`*, then n is prime with probability at least $1 - (1/4)^t$.*

Proof. (*i*) This follows from Proposition 2.12. If $a^k \not\equiv 1 \pmod{n}$ and $a^{2^i k} \not\equiv -1 \pmod{n}$ for all $i \in \{1, \ldots, r-1\}$, then n is a composite number.
(*ii*) This follows from Proposition 2.15. It tells us that, whenever a satisfies one of both conditions of Proposition 2.12 (*ii*), then n is composite with probability at most $1/4$, and in this case Miller-Rabin(n, t) returns `Probably Prime` with probability at most $(1/4)^t$. Therefore, the probability that n is prime and Miller-Rabin(n, t) returns `Probably Prime` is at least $1 - (1/4)^t$. □

[14]Note that no formal proof has confirmed the result so far, but it is most often believed to be true.

2.4.2 Running time

The running time of an algorithm is its asymptotic time complexity, that is the time required by the longest part of the algorithm. It is clear that if one step of the algorithm takes more time to run, then this time is also the algorithm's running time, since all the other steps are bounded by this time. Running time is expressed in terms of the big O notation, that we already encountered in the introduction. Let us shortly recall this notation.

Let f, g be two functions. We say that $f(n) = O(g(n))$ as $n \to +\infty$ if and only if there exist two constants n_0 and $C > 0$ such that for all $n \geq n_0 : |f(n)| \leq C|g(n)|$.

In this subsection we will analyze the running time of the Miller-Rabin primality algorithm described in Algorithm 2.1. The result is stated in the following theorem.

Theorem 2.18. *Let $n, t \in \mathbb{N}, n \geq 3$. The running time of* Miller-Rabin*(n, t) is $O(\log^3(n))$.*

Proof. By using fast exponentiation, it takes $O(\log(n))$ modular multiplications to compute $a^k \pmod{n}$, where one modular multiplication again takes time $O(\log^2(n))$. Once computed $a^k \pmod{n}$, the remaining terms $a^{2k}, a^{4k}, \ldots, a^{2^r k}$ are obtained by $r \leq \log(n)$ repeated squarings modulo n, each of which requiring again $O(\log(n))$ modular multiplications. All the other steps in the algorithm need less time, so that the running time is $O(\log^3(n))$, by adding these times up. $\square$

3 The AKS primality test

In this section, we will go through the AKS primality test with special focus on complexity. We are talking of a quite recent algorithm, that surprisingly appeared in 2002, formalized by three Indian scientists, Manindra Agrawal, Neeraj Kayal and Nitin Saxena (therefore the abbreviation AKS). The big difference with the Miller-Rabin primality test consists in the fact that this is a deterministic test and hence yields proved results.

Analougously to the previous section, we should first start preparing the necessary ingredients and present the algorithm later on.

3.1 Preliminaries

In this preliminary section, we present (resp. recall) some important basic results.

3.1.1 Roots of unity

We recall the definitions of roots of unity and primitive roots of unity and study some of their properties.

Definition 3.1. Let $n \in \mathbb{N}$. A complex number ζ is called a *nth root of unity* if $\zeta^n = 1$.

It is clear that there are exactly n nth roots of unity. These are precisely the complex numbers

$$\zeta_k = e^{\frac{2\pi i}{n} k} \text{ for } k = 1, \ldots, n,$$

which are the roots of the polynomial $X^n - 1$.

We can extend this definition to define the primitive roots of unity.

Definition 3.2. Let $n \in \mathbb{N}$. A complex number ζ is called a *primitive nth root of unity* if n is the smallest positive integer such that $\zeta^n = 1$, i.e. $\operatorname{ord}(\zeta) = n$.

Remark 3.3. We note that the complex number $e^{\frac{2\pi i}{n}}$ is always a primitive nth root of unity since $\left(e^{\frac{2\pi i}{n}}\right)^n = e^{2\pi i} = 1$ and the order of $e^{\frac{2\pi i}{n}}$ is n.

Lemma 3.4. *Let $n \in \mathbb{N}$, and ζ a primitive nth root of unity. Then all the nth roots of unity are the numbers $1, \zeta, \zeta^2, \dots, \zeta^{n-1}$.*

Proof. For every $0 \leq k < n$, ζ^k is an nth root of unity by the fact that $(\zeta^k)^n = (\zeta^n)^k = 1^k = 1$. Therefore, the numbers $\zeta^0 = 1, \zeta^1 = \zeta, \zeta^2, \dots, \zeta^{n-1}$ are all distinct, hence these must be all the nth roots of unity, since there are exactly n such roots. □

Example 3.5. Let us look at 7th roots of unity. By the above, the 7th roots of unity are the numbers $1, \zeta, \dots, \zeta^6$ for $\zeta = e^{\frac{2\pi i}{7}}$, which is a primitive 7th root of unity, by Remark 3.3.

Lemma 3.6. *Let $n, d \in \mathbb{N}$ and ζ a primitive nth root of unity. Then ζ^k is a primitive nth root of unity if and only if* $\gcd(n, k) = 1$.

Proof. ($\Longrightarrow$) We show the contrapositive. Assume that $\gcd(n, k) = g \neq 1$. Then $(\zeta^k)^{n/g} = (\zeta^n)^{k/g} = 1^{k/g} = 1$, but $n/g < n$, so that $\mathrm{ord}(\zeta^k) \neq n$, and therefore ζ^k is not a primitive nth root of unity.
($\Longleftarrow$) Assume that $\gcd(n, k) = 1$ and call $\mathrm{ord}(\zeta^k) = a$. If ζ is a primitive nth root of unity, then $\mathrm{ord}(\zeta) = n$. We want to see that $a = n$. Consider $G = \langle \zeta \rangle$ and $H = \langle \zeta^k \rangle \subset G$. By construction, $|G| = n$ and $|H| = a$, thus it follows from Lagrange's Theorem that $a \mid n$. Moreover, from $\zeta^{ka} = 1$, it follows $n \mid ka$. But $\gcd(n, k) = 1$ implies $n \mid a$. So $a = n$, and ζ^k is a primitive nth root of unity. □

Corollary 3.7. *There are exactly $\varphi(n)$ primitive nth roots of unity.*

Proof. By definition of Euler's function, there are precisely $\varphi(n)$ integers $k \in \{0, \dots, n\}$ that satisfy $\gcd(k, n) = 1$, so there are exactly $\varphi(n)$ primitive nth roots of unity by Lemma 3.6. □

3.1.2 Cyclotomic polynomials

There are several possible ways to define cyclotomic polynomials. One way is to define them by means of primitive roots of unity that we studied above.

Definition 3.8. Let $n \in \mathbb{N}$. The *nth cyclotomic polynomial*, denoted by $\Phi_n(X)$ is the polynomial having exactly the primitive nth roots of unity as roots, that is

$$\Phi_n(X) = \prod_{\mathrm{ord}(\zeta)=n} (X - \zeta) = \prod_{\substack{k=1 \\ \gcd(n,k)=1}}^{n} (X - \zeta^k).$$

Let us work out some examples.

Example 3.9. For $n = 3$, the 3rd roots of unity are the complex numbers $1, \zeta, \zeta^2$ for $\zeta = e^{\frac{2\pi i}{3}}$. That are the numbers $1, -1/2 + i\sqrt{3}/2, -1/2 - i\sqrt{3}/2$, among which only ζ and ζ^2 are primitive ($\gcd(1, 3) = \gcd(2, 3) = 1$). Thus the 3rd cyclotomic polynomial is given by

$$\Phi_3(X) = (X - \zeta)(X - \zeta^2) = X^2 + X + 1.$$

For $n = 4$, we have that the 4th roots of unity are $1, \zeta, \zeta^2, \zeta^3$ where $\zeta = e^{\frac{2\pi i}{4}} = -i$. That are $1, -i, -1$ and i. The primitive 4th roots of unity are $-i$ and i (since they have order 4). Therefore the 4th cyclotomic polynomial is

$$\Phi_4(X) = (X - i)(X + i) = X^2 + 1.$$

Remark 3.10. All cyclotomic polynomials are monic with integer coefficients (we will not prove this latter result). By Corollary 3.7, the nth cyclotomic polynomial has degree $\varphi(n)$.

Let us now come to an important result of cyclotomic polynomials.

Proposition 3.11. *Let $d, n \in \mathbb{N}$. Then*

$$X^n - 1 = \prod_{d \mid n} \Phi_d(X).$$

Proof. Since the right hand side is a product of cyclotomic polynomials, it is monic, and hence the polynomials on both sides are monic. Let ζ be an nth root of unity. By definition of the right hand side, we have that $\mathrm{ord}(\zeta) = d$ for some $d \mid n$, and hence ζ is a primitive dth root of unity. But since $d \mid n$, we have that $(X^d - 1) \mid (X^n - 1)$ and therefore $\zeta^d - 1$ divides $\zeta^n - 1 = 0$, which means that $\zeta^d = 1$, i.e. that ζ is also a root of the polynomial in the right hand side. From this, it follows that polynomials on both sides are monic and have the same roots, so they are equal. $\square$

We will now see two corollaries that immediately follow from this latter proposition. The first one just proves a well-known formula, and the second one gives an expression for the pth cyclotomic polynomial, where p is a prime number.

Corollary 3.12. *Let $n, d \in \mathbb{N}$. Then*

$$\sum_{d \mid n} \varphi(d) = n.$$

Proof. Since the polynomials in Proposition 3.11 are equal, they must have same degree. The left hand side has degree n, so the degree of the right hand side must equal n, too. But since the polynomial on the right hand side is a product of cyclotomic polynomials, it has degree $\sum_{d \mid n} \varphi(d)$. $\square$

Corollary 3.13. *Let p be a prime number. Then*

$$\Phi_p(X) = \sum_{i=0}^{p-1} X^i.$$

Proof. By Proposition 3.11, it results that $X^p - 1 = \Phi_1(X)\Phi_p(X)$. Substituting $\Phi_1(X) = X - 1$, it comes

$$\Phi_p(X) = \frac{X^p - 1}{X - 1} = X^{p-1} + X^{p-2} \ldots + X + 1 = \sum_{i=0}^{p-1} X^i. \qquad \square$$

Let us see another important fact of cyclotomic polynomials. Let $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$ denote the finite field with p elements.

Proposition 3.14. *Let $n, p \in \mathbb{N}$, with p a prime number and $\gcd(n, p) = 1$. Then the polyomial $\Phi_n(X)$ splits in the field $\mathbb{F}_p[X]$ into irreducible factors of same degree $\mathrm{ord}_n(p)$.*[15]

Proof. We show that the irreducible factors all have same degree. Let d denote the degree of some irreducible factor $Q(X)$ of $\Phi_n(X)$, and call $k := \mathrm{ord}_n(p)$. We are to show that $d = k$.

Consider the field $K := \mathbb{F}_p/(Q(X))$. Since $\mathbb{F}_p$ is finite of prime order p and Q is irreducible of degree d, K is a finite field of cardinality p^d. By Lagrange, for all $x \in K^\times$, it holds $x^{p^d - 1} = 1$. Then K contains a root of Q, thus also a root of Φ_n, which is a primitive nth root of unity, say ζ. Then $\zeta^{p^d-1} = 1$ and $\zeta^n = 1$ implies that $n \mid (p^d - 1)$. So write

[15]The notation $\mathrm{ord}_n(p)$ stands for the order of p modulo n. For the precise definition, have a look at Definition 3.31 page 25.

$p^d - 1 = kn$ for some $k \in \mathbb{Z}$ to see that $p^d \equiv 1 \pmod{n}$. But since by definition of k, k is the smallest positive integer such that $p^k \equiv 1 \pmod{n}$, we conclude from this that $d \geq k$.

To show the other inequality, we first observe that ζ is a root of the polynomial $X^{p^k} - X$. Indeed, since $\zeta^n = 1$ and $n \mid (\zeta^k - 1)$, we have that $\zeta^{p^k} = \zeta$. Consider the subfield of K, $L = \{x \in K : x^{p^k} = x\} \subset K$. By the above, $\zeta \in L$. Since ζ is a generator of $K^\times$, we get that $L = K$, and thus $|L| = |K| = p^d \leq p^k$, and hence $d \leq k$.

Putting both observations together, we have shown that $d = k$ and since Q was arbitrarily chosen, this holds for all irreducible factor of $\Phi_n(X)$. $\square$

One could say much more about cyclotomic polynomials, but our task was just to briefly study them in order to understand some passages of the coming theory.

3.1.3 Binomial coefficients

First we remind that, given $n, k \in \mathbb{N}$ with $n, k \geq 0$ and $k \leq n$, the *binomial coefficient* is defined by

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}. \tag{7}$$

From this follows an immediate consequence, namely that

$$\binom{n}{k} = \binom{n}{n-k}. \tag{8}$$

We also recall PASCAL's rule[16]: For $1 \leq k \leq n+1$,

$$\binom{n}{k} + \binom{n}{k-1} = \binom{n+1}{k}. \tag{9}$$

The proof of this formula is straight forward, it suffices to expand the left hand side of the identity using (7).

Let us now explore some properties of the binomial coefficient. The coming lemma gives the iterated version of Pascal's formula.

Lemma 3.15. *Let $n, k \in \mathbb{N}^*$ with $k \leq n$. Then*

$$\sum_{j=k}^{n} \binom{j}{k} = \binom{n+1}{k+1}.$$

Proof. This proof goes by induction on n. For $n = 1$, we have that $k = 1$, and $\binom{1}{1} = \binom{2}{2} = 1$. Assume now that the equality holds for a certain $n \in \mathbb{N}^*$ and show that it still holds for $n+1$. Then using firstly the induction hypothesis (IH) and then Pascal's formula (9), we see that

$$\sum_{j=k}^{n+1} \binom{j}{k} = \sum_{j=k}^{n} \binom{j}{k} + \binom{n+1}{k} \underset{\text{(IH)}}{=} \binom{n+1}{k+1} + \binom{n+1}{k} \underset{(9)}{=} \binom{n+2}{k+1},$$

which terminates the proof. $\square$

This lemma is used to prove the following result.

Lemma 3.16. *Let $n, k \in \mathbb{N}^*$. The number of possibilities to choose n integers such that their sum is less than or equal to $k-1$ is $\binom{n+k-1}{n}$.*

[16]This has been found by Blaise Pascal (1623 – 1662), a French mathematician, among other disciplines.

Proof. We show this by induction on n. If $n = 1$, on the one hand, there are precisely k possibilities to choose one integer such that it is less or equal to $k-1$, and on the other hand, $\binom{n+k-1}{n} = \binom{k}{1} = k$, which proves the base case.
For the induction step, we assume that the result holds for some $n \in \mathbb{N}^*$ and prove that it is still true for $n+1$. We have to count possibilties to choose $n+1$ integers, say $a_0, a_1, \dots, a_n$, such that $\sum_{i=0}^{n} a_i \leq k-1$. If we fix the choice of a_0, it remains to choose only n integers $a_1, \dots, a_n$, and we can thus use our induction hypothesis. For simplicity, call $\sigma(a_0)$ the number of possibilities to choose $n+1$ integers $a_0, a_1, \dots, a_n$ with fixed a_0 and such that $\sum_{i=0}^{n} a_i \leq k-1$. With this notation we get, by induction hypothesis, that

$$\sigma(j) = \binom{n+k-j-1}{n},$$

for $0 \leq j \leq k-1$.[17] Then adding together all the possibilities that we get with different choices for a_0, we are left with

$$\sum_{j=0}^{k-1} \sigma(j) = \sum_{j=0}^{k-1} \binom{n+k-j-1}{n} = \sum_{j=n}^{n+k-1} \binom{j}{n} = \binom{n+k}{n+1}$$

possibilities to choose $a_0, a_1, \dots, a_n$ such that $\sum_{i=0}^{n} a_i \leq k-1$. Note that we obtained the last equality by Lemma 3.15. This is the wanted result for $n+1$, and therefore the result is true for all $n \in \mathbb{N}^*$. □

Lemma 3.17. *Let $k \in \mathbb{N}, k \geq 2$. Then*

$$\binom{2k+1}{k} > 2^{k+1}.$$

Proof. We prove this by induction on k. The base case is clear, since for $k = 2$, we have $\binom{5}{2} = 10 > 8 = 2^{2+1}$. We assume that the inequality holds for a certain $k \geq 2$ and prove that it still holds for $k+1$:

$$\begin{aligned}\binom{2(k+1)+1}{k+1} &= \binom{2k+3}{k+1} = \frac{(2k+3)!}{(k+1)!(k+2)!} \\ &= \frac{(2k+1)!}{k!(k+1)!} \cdot \frac{(2k+2)(2k+3)}{(k+1)(k+2)} \\ &= 2\underbrace{\binom{2k+1}{k}}_{>2^{k+1}} \underbrace{\frac{2k+3}{k+2}}_{>1} > 2^{k+2}.\end{aligned}$$

So this holds for all $k \geq 2$. □

Lemma 3.18. *Let $a, b, c \in \mathbb{N}$, such that $a \geq b$. Then*

$$\binom{a+c}{c} \geq \binom{b+c}{c}.$$

Proof. This follows from the definition of the binomial coefficients. We have

$$\begin{aligned}\binom{a+c}{c} &= \frac{(a+c)!}{a!c!} = \frac{(a+c)(a+c-1)\cdot \ldots \cdot (a+1)}{c!} \\ &\geq \frac{(b+c)(b+c-1)\cdot \ldots \cdot (b+1)}{c!} = \binom{b+c}{c}.\end{aligned}$$

□

[17]For example, if $a_0 = 1$, then $\sigma(1) = \binom{n+k-2}{n}$ since the remaining integers $a_1, \dots, a_n$, need only have sum less than or equal to $k-2$. To write it compactly, fix $a_0 = j$ with $0 \leq j \leq k-1$. Then we have to choose $a_1, \dots, a_n$ so that their sum is less than or equal to $k-1-j$. Therefore by induction hypothesis, $\sigma(j) = \binom{n+k-j-1}{n}$.

3.1.4 Introspective numbers

Before giving the definition of introspective numbers, we recall the definition of the 'double mod' notation.

Definition 3.19. Let K be a ring, $f(X), g(X), h(X) \in K[X]$ and $n \in \mathbb{N}$. Then $f(X) \equiv g(X) \pmod{h(X), n}$ if there exist $u(X), v(X) \in K[X]$ such that

$$f(X) - g(X) = nu(X) + h(X)v(X).$$

Example 3.20. As example, we check the congruence

$$X^3 + X^2 + X + 1 \equiv -X^3 + 6X^2 - 14X - 20 \pmod{X^2 + 1, 13}.$$

Calling $f(X) = X^3 + X^2 + X + 1$ and $g(X) = -X^3 + +6X^2 - 14X - 20$, we compute $f(X) - g(X)$ and obtain

$$f(X) - g(X) = 2X^3 - 5X^2 - 15X + 21,$$

which we need to reduce modulo $h(X) = X^2 + 1$ and $n = 13$. A standard polynomial division gives

$$f(X) - g(X) = (X^2 + 1)(2X - 5) + 13(X + 2),$$

so that we can choose $u(X) = X + 2$ and $v(X) = 2X - 5$ in Definition 3.19.

Let us now come to the definition of introspective numbers.

Definition 3.21. Let K be a ring, $f(X) \in K[X]$. Then $n \in \mathbb{N}$ is called *introspective for* $f(X)$ if

$$f(X)^n \equiv f(X^n) \pmod{X^r - 1, p},$$

for some prime number p and some $r \in \mathbb{N}$.

Remark 3.22. Note however that this definition is not universal, but was introduced by the three computer scientists in their original paper [5] in order to simplify their own theory by giving this property a name. In order to give a more general definition, one could define n to be introspective for $f(X)$ modulo any $h(X) \in K[X]$ and $k \in \mathbb{N}$ if

$$f(X)^n \equiv f(X^n) \pmod{h(X), k}.$$

Let us view this definition on an example.

Example 3.23. Let us see a trivial example, just to make sure how to work with the definition. Let $K = \mathbb{Z}$ and consider the polynomial $f(X) = X + 1$. Then 2 is introspective for $f(X)$ modulo $n = 2$ and $h(X) = 1_{\mathbb{Z}[X]^\times}$ (the constant 1 polynomial) because

$$f(X)^2 \equiv (X+1)^2 \equiv X^2 + 2X + 1 \equiv X^2 + 1 \equiv f(X^2) \pmod{1_{\mathbb{Z}[X]^\times}, 2}.$$

We will now discuss two important properties of introspective numbers.

Lemma 3.24. *Let K be a ring, $f(X) \in K[X]$, $a \in \mathbb{N}$. If a is introspective for $f(X)$, then a is introspective for $f(X^k)$ for all $k \in \mathbb{N}^*$.*

Proof. We want to see that $f(X^k)^a \equiv f((X^k)^a) \pmod{X^r - 1, p}$. Since a is inrospective for $f(X)$, we have by definition that

$$f(X)^a \equiv f(X^a) \pmod{X^r - 1, p}.$$

Writing this out for X^k, we get

$$f(X^k)^a \equiv f((X^k)^a) \pmod{X^{kr} - 1, p},$$

but since $(X^r - 1) \mid (X^{kr} - 1)$, we also have that

$$f(X^k)^a \equiv f((X^k)^a) \pmod{X^r - 1, p},$$

which is what was claimed[18]. □

Lemma 3.25. *Let K be a ring, $f(X) \in K[X]$, $a, b \in \mathbb{N}$. If a and b are introspective for $f(X)$, then ab also is.*

Proof. To see that ab is introspective for $f(X)$, we check the definition, that says $f(X)^{ab} \equiv f(X^{ab}) \pmod{X^r - 1, p}$. Since a is introspective for $f(X)$, we have

$$f(X)^{ab} \equiv [f(X)^a]^b \equiv f(X^a)^b \pmod{X^r - 1, p}.$$

But, by Lemma 3.24, b is introspective for $f(X^a)$, therefore

$$f(X^a)^b \equiv f((X^a)^b) \equiv f(X^{ab}) \pmod{X^r - 1, p}.$$ □

Lemma 3.26. *Let K be a ring, $f(X), g(X) \in K[X]$, $a \in \mathbb{N}$. If a is introspective for $f(X)$ and $g(X)$, then a is introspective for the product $f(X)g(X)$.*

Proof. We check the definition. Since a is introspective for both $f(X)$ and $g(X)$, it results that

$$[f(X)g(X)]^a \equiv f(X)^a g(X)^a \equiv f(X^a)g(X^a) \pmod{X^r - 1, p}.$$ □

Remark 3.27. A reformulation of the above lemmas could be read as follows.

- Lemma 3.25 shows that the set of all introspective numbers for a fixed polynomial $f(X)$ is closed under multiplication.
- Lemma 3.26 shows that the set of all polynomials, for which a fixed integer a is introspective, is closed under multiplication.

3.2 Key points and description of the algorithm

In this section we present the key idea of the AKS algorithm. We state it in the following lemma.

Lemma 3.28. *Let $n \in \mathbb{N}, n \geq 2$. Then n is a prime number if and only if*

$$(X + a)^n \equiv X^n + a \pmod{n}, \tag{10}$$

in $\mathbb{Z}[X]$, for all $a \in \mathbb{Z}$ with $\gcd(a, n) = 1$.

Proof. ($\Longrightarrow$) Assume that n is a prime number. By using the binomial expansion, the left hand side of (10) is

$$\sum_{k=0}^{n} \binom{n}{k} X^k a^{n-k} = X^n + \left(\sum_{k=1}^{n-1} \binom{n}{k} X^k a^{n-k} \right) + a^n. \tag{11}$$

[18]To see the last step, we can write this out by Definition 3.19. If $f(X^k)^a \equiv f((X^k)^a) \pmod{X^{kr} - 1, p}$, there exist $u(X), v(X) \in K[X]$ such that

$$f(X^k)^a - f((X^k)^a) = pu(X) + (X^{kr} - 1)v(X).$$

Now, since $(X^r - 1) \mid (X^{kr} - 1)$, there is $\alpha(X) \in K[X]$ so that $X^{kr} - 1 = \alpha(X)(X^r - 1)$. By substitution, we thus get

$$f(X^k)^a - f((X^k)^a) = pu(X) + (X^r - 1)w(X),$$

for $w(X) = \alpha(X)v(X) \in K[X]$, showing that $f(X^k)^a \equiv f((X^k)^a) \pmod{X^r - 1, p}$.

We claim that since n is prime, $n \mid \binom{n}{k}$ for all $1 \leq k \leq n-1$. Indeed, by the definition of the binomial coefficient, we have that

$$\binom{n}{k} k! = \frac{n!}{(n-k)!} = \prod_{i=0}^{k-1} (n-i). \tag{12}$$

Since n clearly divides the right hand side of (12) (since $k \geq 1$), n must also divide its left hand side. But since n is prime and $n \nmid k!$ (because $k < n$), n must divide $\binom{n}{k}$ for all $1 \leq k < n$. Therefore reducing (11) modulo n, we are left with

$$(X+a)^n \equiv X^n + a^n \pmod{n}.$$

To conclude, we use Fermat's Little Theorem (Theorem 2.1), to see that $a^n \equiv a \pmod{n}$.
($\Longleftarrow$) We show the contrapositive of this implication. Therefore assume that n is composite. Let p be a prime factor of n and assume additionnally that p^k is the largest power of p that divides n. From the expansion of

$$\binom{n}{p} = \frac{n!}{p!(n-p)!} = \frac{n(n-1) \cdot \ldots \cdot (n-(p-1))}{p(p-1) \cdot \ldots \cdot 1},$$

we note that p divides the numerator (because $p \mid n$) and the denominator of $\binom{n}{p}$. Now, since p^k is the largest power of p dividing n, the largest power of p dividing $\binom{n}{p}$ is p^{k-1}. Therefore n does not divide $\binom{n}{p}$ and hence

$$(X+a)^n \not\equiv X^n + a \pmod{n}. \qquad \square$$

Here is an easy example that will help us visualize this result.

Example 3.29. Take $n = 3$ and $a = 4$. We note that we are in conditions of Lemma 3.28, since 3 is prime and $\gcd(3,4) = 1$. Then

$$(X+4)^3 = X^3 + 12X^2 + 48X + 64 \equiv X^3 + 64 \equiv X^3 + 1 \pmod{3},$$

but $1 \equiv 4 \pmod{3}$, and we see that (10) holds.

Towards a primality test. Making use of Lemma 3.28, we could easily formulate a primality criterion: If we wanted to check whether n is prime, this test would consist in computing

$$(X+a)^n - (X^n + a) \tag{13}$$

for any integer a relatively prime to n and then decide whether it is congruent to 0 (if each coefficient in the expansion of (13) is a multiple of n) or is not congruent to 0 (if not all coefficients are multiples of n) modulo n. Though this is a correct test, this is not advised, since it is very slow. Indeed, $(X+a)^n$ has in its expansion n coefficients, which means that this test requires to stock n coefficients to be reduced modulo n. Since our goal is to work with large n, this would clearly take too much time.

However, one may still use this concept, and try to reduce the number of evaluations. This could eventually be done by reducing $(X+a)^n$ modulo some polynomial of the form $X^r - 1$ for small chosen r and in addition modulo n as well, so that we are to verify the modular equality

$$(X+a)^n \equiv X^n + a \pmod{X^r - 1, n}, \tag{14}$$

in the sense of Definition 3.19. By Lemma 3.28, we see that if n is prime then (14) is satisfied for all a and r. But, analogously to what we saw for Fermat's Little Theorem and Carmichael numbers, it could occur that (14) is satisfied for composite n as well for several values of a and r. To avoid this problem, we choose r and the number of values of a appropriately and it will turn out that if (14) is satisfied, then n must be a prime power. We will also see that we can bound, the value of r and the number of integers a we consider, by a polynomial in $\log(n)$, which will then show that the test is deterministic.

Before coming to the main theorem of this section, we should recall two definitions.

Definition 3.30. Let $n \in \mathbb{N}$. Then n is called a *perfect power* if there exist $a, b \in \mathbb{N}$ with $a, b > 1$ such that $n = a^b$.

Definition 3.31. Let $r, n \in \mathbb{N}$ such that $\gcd(r, n) = 1$. Then the *order of n modulo r*, denoted by $\operatorname{ord}_r(n)$, is the smallest positive integer k such that $n^k \equiv 1 \pmod{r}$.[19]

The AKS primality test can be summarised in the following theorem, which we will prove later when studying the correctness of the algorithm.

Theorem 3.32. *Let $n \in \mathbb{N}, n \geq 2$, and $r \in \mathbb{N}$, with $r < n$ and $\gcd(r, n) = 1$ such that* $\operatorname{ord}_r(n) > \log^2(n)$ modulo r. Then n is a prime number if and only if the three conditions are satisfied:

(*i*) n is not a perfect power;

(*ii*) n has no prime factor $p \leq r$;

(*iii*) $(X + a)^n \equiv X^n + a \pmod{X^r - 1, n}$ for all integers $1 \leq a \leq \sqrt{\varphi(r)}\log(n)$.

3.3 The algorithm

As a matter of simplicity, we present a pseudocode of the algorithm, which will be easier to read and help have an overwiev over the different steps.

Algorithm 3.1 AKS primality test

```
Require: $n \in \mathbb{N}, n > 1$
Ensure: $n$ is Prime or $n$ is Composite
1: function AKS($n$)
2:     if $\exists\, a, b \in \mathbb{N}, a, b > 1$ such that $n = a^b$ then
3:         return Composite
4:     end if
5:     Find the smallest $r$ such that $\operatorname{ord}_r(n) > \log^2(n)$
6:     if $1 < \gcd(a, n) < n$ for some $a \leq r$ then
7:         return Composite
8:     end if
9:     if $n \leq r$ then
10:        return Prime
11:    end if
12:    for $a = 1$ to $\lfloor \sqrt{\varphi(r)}\log(n) \rfloor$ do
13:        if $(X + a)^n \not\equiv X^n + a \pmod{X^r - 1, n}$ then
14:            return Composite
15:        end if
16:    end for
17:    return Prime
18: end function
```

[19]Notice the indexed r in the notation, to indicate modulo which integer the order is calculated.

Main steps. This algorithm is set up by six main steps that we describe in the following table.

step	**lines**	**description**
I	$2-4$	The algorithm checks, using Definition 3.30, condition (i) of Theorem 3.32, that is whether n is a perfect power. If n is a perfect power, then n is composite.
II	5	The algorithm looks for the smallest value of r that verifies $\text{ord}_r(n) > \log^2(n)$, according to the assumption of Theorem 3.32.
III	$6-8$	The algorithm should check condition (ii) of Theorem 3.32, testing whether n has a prime factor smaller than r. In order not to be redundant (in the sense, not to use a primality test inside a primality test), the algorithm simply checks whether n has any divisor smaller than r, making no big time change since r is often small.
IV	$9-11$	If the value of r computed in line 5 is greater than or equal to n, then n must be prime.
V	$12-16$	The algorithm tests condition (iii) of Theorem 3.32. It considers all integers $a = 1$ up to $\lfloor\sqrt{\varphi(r)}\log(n)\rfloor$ (which is the bound in $\log(n)$ we described before; note that it is well defined since at this moment, the program knows both n and r). If the modular congruence described in (14) does not hold, then n is composite.
VI	17	The algorithm ends returning `Prime` when none of the conditions are satisfied.

Remark 3.33. Composite numbers are identified as composite either in steps **I**, **III** or **V**. Not all prime numbers need to go to step **VI** to be declared as prime. It could occur that already step **IV** identifies them as prime numbers.

3.4 Correctness and complexity

This is undoubtedly the most important part of this section. Firstly, we will prove why Algorithm 3.1 works. Then, we will discuss its running time and see that it is indeed in **P**, i.e. that it runs in polynomial time.

3.4.1 Proof of correctness

The goal of this section is to prove the main theorem, that is:

Theorem 3.34. *Let $n \in \mathbb{N}, n > 1$. Then* AKS(n) *returns* `Prime` *if and only if n is a prime number.*

The proof of this theorem is quite a bit long, that's why we split it up in two main lemmas that will follow. One part of the proof is easy. That is the following lemma.

Lemma 3.35. *Let $n \in \mathbb{N}, n > 1$ a prime number. Then* AKS(n) *returns* `Prime` .

Proof. Assume that n is a prime number. Then n is neither a perfect power, nor has a prime factor smaller than r, therefore steps **I** and **III** cannot return `Composite`. Moreover, by Lemma 3.28, the congruence in step **V** is not satisfied and hence this step cannot return `Composite`. Therefore AKS(n) will return `Prime` either in steps **IV** or **VI**. □

To accomplish the proof of Theorem 3.34, we need to prove the converse of the latter lemma. It's here where we need to work a little more, and therefore we will soon set up a sequence of claims to simplify. But, nevertheless let us roughly think what happens when AKS(n) returns `Prime`. If AKS(n) returns `Prime`, then this can only happen in steps **IV** or **VI**. Let us first assume that it returns `Prime` in step **IV**. Then n must indeed be prime since otherwise either step **I** or step **III** would have identified n as a composite number, and then the program would not have gone to step **IV**. So we can assume that this is not the case, assuming that $n > r$. Then, the remaining case is when AKS(n) returns `Prime` in step **VI**, that we should analyze now. Therefore we suppose from now on that AKS(n) returns `Prime` in **VI**.

We first show the existence of the integer r that is to be computed in step **II**, that is the smallest value so that $\text{ord}_r(n) > \log^2(n)$. But, before doing so, we first need to recall a property of the least common multiple.

Lemma 3.36. *Let $n \in \mathbb{N}, n \geq 7$. Then*

$$LCM(n) \geq 2^n,$$

where $LCM(n) := \text{lcm}(1, \ldots, n)$ denotes the least common multiple of the first n numbers.

The proof of this result is non trivial, and we will skip it, otherwise it would lead us away from our main object. But we will give an example instead.

Example 3.37. Take $n = 15 \geq 7$. Then $LCM(15)$ is the least common multiple of $1, 2, \ldots, 15$, and results to be 360360, which is indeed greater than $2^{15} = 32768$. In order to see that we need to restrict the result to integers $n \geq 7$, we can compute $LCM(6) = 60$, but $2^6 = 64 > 60$.

The following lemma now gives an upper bound for the value of r.

Lemma 3.38. *Let $n \in \mathbb{N}, n > 1$. There exists $r \leq \max\{3, \lceil \log^5(n) \rceil\}$ such that $\text{ord}_r(n) > \log^2(n)$.*

Proof. If $n = 2$, then we have to choose r smallest possible such that $\text{ord}_r(2) > 1$, and we see that $r = 3$ is convenient. Now assume $n \geq 3$. We give a proof by contradiction. Therefore consider $r_1, r_2, \ldots, r_t$ integers such that for all $1 \leq i \leq t$, r_i satisfies $\text{ord}_{r_i}(n) \leq \log^2(n)$ and $r_i \leq \lceil \log^5(n) \rceil$. Then for all i, r_i divides the product

$$(n-1)(n^2-1) \cdot \ldots \cdot (n^{\lfloor \log^2(n) \rfloor} - 1) = \prod_{i=1}^{\lfloor \log^2(n) \rfloor} (n^i - 1), \tag{15}$$

since for all i, $n^{\text{ord}_{r_i}(n)} \equiv 1 \pmod{r_i}$ and therefore r_i divides $n^{\text{ord}_{r_i}(n)} - 1$, which is a factor of the product in (15), since we assumed that $1 \leq \text{ord}_{r_i}(n) \leq \log^2(n)$, for all i.

The idea now is to bound the product in (15) in two different ways and deduce a contradiction.

On the one hand, the product is smaller than the biggest power of n in the product, that is

$$\begin{aligned}\prod_{i=1}^{\lfloor \log^2(n) \rfloor} (n^i - 1) &< n \cdot n^2 \cdot \ldots \cdot n^{\lfloor \log^2(n) \rfloor} \\ &= n^{1+2+\ldots+\lfloor \log^2(n) \rfloor} \\ &= n^{\frac{1}{2}\lfloor \log^2(n) \rfloor \left(\lfloor \log^2(n) \rfloor + 1\right)},\end{aligned}$$

where for the last equality we used the formula for the sum of an arithmetic progression. Now, we use a reasoning from analysis that says that for $x \geq 1$, $\frac{1}{2}x(x+1) \leq x^2$.[20] We apply this to $\lfloor \log^2(n) \rfloor \geq 1$ (as $n \geq 2$), an thus get

$$n^{\frac{1}{2}\lfloor \log^2(n) \rfloor \left(\lfloor \log^2(n) \rfloor + 1\right)} < n^{\lfloor \log^2(n) \rfloor^2} = n^{\lfloor \log^4(n) \rfloor}.$$

[20]Indeed, $\frac{1}{2}x(x+1) \leq x^2$ is equivalent to $x^2 - x \geq 0$ which is always true for $x \geq 1$.

Writing $n = 2^{\log(n)}$, we obtain $n^{\lfloor \log^4(n) \rfloor} = 2^{\lfloor \log^5(n) \rfloor}$, and thus

$$\prod_{i=1}^{\lfloor \log^2(n) \rfloor} (n^i - 1) < 2^{\lfloor \log^5(n) \rfloor}. \tag{16}$$

On the other hand, as $n \geq 3$, $\lceil \log^5(n) \rceil \geq \lceil \log^5(3) \rceil = 11 \geq 7$, hence we can apply Lemma 3.36 to see that $LCM\left(\lceil \log^5(n) \rceil\right) \geq 2^{\lceil \log^5(n) \rceil}$. For all $r \leq \lceil \log^4(n) \rceil$ such that $\mathrm{ord}_r(n) \leq \log^2(n)$, r divides the product in (15), by the same argument as above with r_i, and in particular $LCM\left(\lceil \log^5(n) \rceil\right)$ divides the product, too. Therefore

$$2^{\lceil \log^5(n) \rceil} \leq LCM\left(\lceil \log^5(n) \rceil\right) \leq \prod_{i=1}^{\lfloor \log^2(n) \rfloor} (n^i - 1), \tag{17}$$

which contradicts (16). Hence there must exist an $r \leq \lceil \log^5(n) \rceil$ such that $\mathrm{ord}_r(n) > \log^2(n)$. Putting this together with the case, $n = 2$, we see that $r \leq \max\{3, \lceil \log^5(n) \rceil\}$, as wished. □

For what comes, let p be a prime factor of n. The goal is to show that n must be equal to p. We show this stepwise by proving different claims.

Claim 1. *$p > r$.*

Proof. Since by assumption AKS(n) returns `Prime`, it cannot return `Composite` in step **III**, which implies the claim (otherwise n would have p as prime factor smaller than r). □

Claim 2. *$p, n \in (\mathbb{Z}/r\mathbb{Z})^\times$.*

Proof. For this, we need to show that p and n are both relatively prime to r. It is easy to see that $\gcd(p, r) = 1$, since $p > r$ and p is a prime number. Let us now understand why also $\gcd(n, r) = 1$. Indeed, if $\gcd(n, r) \neq 1$, then either step **III** would return `Composite` or step **IV** would return `Prime`, and we excluded both cases by assuming that the algorithm returns `Prime` in step **VI**. □

Claim 3. *Let p be a prime number, $r \in \mathbb{N}$ with $\gcd(p, r) = 1$. Then the map*

$$\begin{array}{cccc} \phi: & \mathbb{F}_p[X]/(X^r - 1) & \longrightarrow & \mathbb{F}_p[X]/(X^r - 1) \\ & g(X) & \longmapsto & g(X)^p \end{array}$$

is a ring isomorphism.

Proof. We first show that ϕ is a ring homomorphism. Let $g(X), h(X) \in \mathbb{F}_p[X]/(X^r - 1)$, then

$$\phi\left((g(X)h(X)\right) = (g(X)h(X))^p = g(X)^p h(X)^p = \phi(g(X))\phi(h(X)),$$

and

$$\phi\left((g(X) + h(X)\right) = (g(X) + h(X))^p = g(X)^p + h(X)^p = \phi(g(X)) + \phi(h(X)),$$

where for the second computation, we used the binomial expansion and the fact that $p \mid \binom{p}{k}$ for all $1 \leq k \leq p - 1$, in the same way we observed this in the proof of Lemma 3.28. In addition, $\phi(1) = 1$, if 1 denotes the constant polynomial 1, which shows with the observations above, that ϕ is a ring homomorphism.

In order to show that ϕ is an isomorphism, we show that ϕ is bijective. Since ϕ is a map between two finite sets of the same cardinality, it suffices to show that ϕ is injective. Let $g(X) \in \ker \phi$ and write $g(X) = a_0 + a_1 X + \ldots + a_{r-1} X^{r-1} + (X^r - 1)$ with $a_i \in \mathbb{F}_p$ for $0 \leq i \leq r - 1$. Then $g(X)^p \in (X^r - 1)$. Since $a_i^p = a_i$ for all $0 \leq i \leq r - 1$, it results that

$$g(X)^p = a_0 + a_1 X^p + \ldots + a_{r-1} X^{p(r-1)} + (X^r - 1). \tag{18}$$

Claim 1 implies that $p \nmid r$. Therefore we can write $p = kr + s$ with $0 < s < r$. Combining this with (18) and reducing modulo r, we get

$$g(X)^p = a_0 + a_1 X^s + \ldots + a_{r-1} X^{(r-1)s} + (X^r - 1).$$

But since by Claim 2, p is invertible in $\mathbb{Z}/r\mathbb{Z}$, it follows that if for some a, b, $ap \equiv bp \pmod{r}$, then $a \equiv b \pmod{r}$. So ϕ permutes the coefficients of $g(X)$ and therefore $g(X) = 0$, which shows that ϕ is injective. □

Claim 4. *Both n and n/p are introspective*[21] *for $X+a$ for all $0 \le a \le \lfloor \sqrt{\varphi(r)} \log(n) \rfloor =: \ell$.*

Proof. Since we assumed that AKS(n) returns `Prime`, it is impossible that it returns `Composite` in step **V**. Therefore the condition in step **V** cannot be satisfied, and thus for all $0 \le a \le \ell$,

$$(X+a)^n \equiv X^n + a \pmod{X^r - 1, n}. \tag{19}$$

In fact step **V** considers the values $1 \le a \le \ell$, but for $a = 0$, (19) is trivially satisfied. Now, since $p \mid n$, it follows that for all $0 \le a \le \ell$,

$$(X+a)^n \equiv X^n + a \pmod{X^r - 1, p}, \tag{20}$$

which already shows that n is introspective for $X + a$. Since p is prime, we can apply Lemma 3.28 to see that for all $0 \le a \le \ell$, we have

$$(X+a)^p \equiv X^p + a \pmod{X^r - 1, p}, \tag{21}$$

from which we see that p is introspective for $X + a$. Then from equalities (20) and (21), and by Claim 3 (considering the inverse map of ϕ, that maps $g(X)$ to $g(X)^{1/p}$), it follows that

$$(X+a)^{n/p} \equiv X^{n/p} + a \pmod{X^r - 1, p}. \tag{22}$$

This shows that n/p is introspective for $X + a$, too. □

Claim 5. *The set of numbers*

$$\mathcal{I} = \{(n/p)^i p^j : i, j \ge 0\} \tag{23}$$

is introspective[22] *for the set of polynomials*

$$\mathcal{P} = \left\{ \prod_{a=0}^{\ell} (X+a)^{\lambda_a} : \lambda_a \ge 0 \right\}. \tag{24}$$

Proof. By Claim 4, we know that n and n/p are introspective for $X + a$. By Lemma 3.25, all the products formed by these two numbers, that is the products $\alpha := (n/p)^i p^j$ for some $i, j \ge 0$, are introspective for $X + a$. But now Lemma 3.26 implies that all α are introspective for the products of polynomials,

$$\beta(X) := X^{\lambda_0}(X+1)^{\lambda_1}(X+2)^{\lambda_2} \cdot \ldots \cdot (X+\ell)^{\lambda_\ell}$$

for some $\lambda_1, \lambda_2, \ldots \lambda_\ell \ge 0$, which precisely give the description of $\mathcal{P}$. So α is introspective for $\beta(X)$, i.e. $\mathcal{I}$ is introspective for $\mathcal{P}$. □

[21] in the sense of Definition 3.21 page 22.

[22] This is a slight abuse of language. Saying that $\mathcal{I}$ is introspective for $\mathcal{P}$, means that every element in $\mathcal{I}$ is introspective for polynomials in $\mathcal{P}$.

In the following, we will define two groups $\mathcal{G}$ and $\mathcal{H}$, based on the sets $\mathcal{I}$ and $\mathcal{P}$, described in Claim 5. Let us define $\mathcal{G}$ as

$$\mathcal{G} = \{\text{residues of numbers in } \mathcal{I} \text{ modulo } r\} \subseteq (\mathbb{Z}/r\mathbb{Z})^{\times}. \tag{25}$$

Since $\gcd(n,r) = \gcd(p,r) = 1$ (by Claim 2), this is clearly a subgroup of $(\mathbb{Z}/r\mathbb{Z})^{\times}$. Call $|\mathcal{G}| = t$. Since r was chosen such that $\mathrm{ord}_r(n) > \log^2(n)$, we have that $t > \log^2(n)$.[23]

We will now define the second group. To do so, we first need an important property about cyclotomic polynomials. Let $\Phi_r(X)$ be the rth cyclotomic polynomial over $\mathbb{F}_p[X]$, the finite field with p elements. By Proposition 3.14, $\Phi_r(X)$ splits into irreducible factors of degree $\mathrm{ord}_r(p)$. Let $h(X)$ be such an irreducible factor of degree $\mathrm{ord}_r(p)$. Then it is clear that $\deg h(X) > 1$ because $\mathrm{ord}_r(p) > 1$. We now define the second group by

$$\mathcal{H} = \{\text{residues of polynomials in } \mathcal{P} \text{ modulo } h(X) \text{ and } p\}. \tag{26}$$

This group is generated by the elements $\{X + a\}_{0 \le a \le \ell}$ in the field $K := \mathbb{F}_p[X]/(h(X))$. Moreover, it is a subgroup of the multiplicative group $K^{\times}$.

Now that we have defined the sets $\mathcal{G}$ and $\mathcal{H}$, our goal is to derive a contradiction somewhere, using both sets. In order to do so, we will first find a lower and upper bound for the cardinality of $\mathcal{H}$, that is numbers x, y such that $x \le |\mathcal{H}| \le y$. We start by giving a lower bound in the following claim.

Claim 6.

$$|\mathcal{H}| \ge \binom{t+\ell}{t-1}.$$

Proof. Let $\alpha \in \mathbb{F}_p$ be a root of $h(X)$. Since $h(X)$ is a factor of $\Phi_r(X)$, which itself is a factor of $X^r - 1$ (by Proposition 3.11), we can write $X^r - 1 = A(X)\Phi_r(X) = A(X)B(X)h(X)$ for some polynomials $A(X), B(X) \in \mathbb{F}_p[X]$. Therefore $h(\alpha) = 0$ implies that $\alpha^r - 1 = 0$, i.e. that $\alpha^r = 1$, which shows that the roots of $h(X)$ are primitive rth roots of unity in K. Now we show that to any two distinct polynomials of degree strictly smaller than t in $\mathcal{P}$ correspond two different residues in $\mathcal{H}$. Therefore let $f(X), g(X) \in \mathcal{P}$ be two distinct polynomials of degree stictly smaller than t. Assume by contradiction that $f(X) = g(X)$ in K. Let $m \in \mathcal{I}$. Raising to the power m, we obtain that $f(X)^m = g(X)^m$ in K. Thus, by Claim 5, using introspection of m for both $f(X)$ and $g(X)$, and the fact that $h(X)$ divides $X^r - 1$, we get $f(X^m) = g(X^m)$ in K. This shows that for all $m \in \mathcal{G}$ (since exponents are taken modulo r in K), X^m is a root of the polynomial $Q(Y) = f(Y) - g(Y)$ in $K[Y]$. We are now close to find a contradiction. On the one hand, by construction, $Q(Y)$ has degree strictly smaller than t (since both $f(X)$ and $g(X)$ have). On the other hand, since $\mathcal{G}$ is a subgroup of $(\mathbb{Z}/r\mathbb{Z})^{\times}$, we have that $\gcd(m,r) = 1$. Therefore each α^m is a primitive rth root of unity (by Lemma 3.6), and since $|\mathcal{G}| = t$, there are exactly t distinct roots of $Q(Y)$ in $K[Y]$. Putting both observation together, we see that $Q(Y)$ has more roots than its degree, which is a contradiction. Therefore, $f(X) \ne g(X)$ in K.
We observe that $\ell < r < p$, since $\ell = \lfloor \sqrt{\varphi(r)} \log(n) \rfloor < \sqrt{r}\log(n) < r$,[24] and by Claim 1, $r < p$. This shows that in the field $\mathbb{F}_p$, we have $i \ne j$ for $0 \le i, j \le \ell$, and so the polynomials $\{X + a\}_{0 \le a \le \ell}$ are all distinct in $\mathbb{F}_p$. Additionally, they are all non zero in H, since they all have degree 1 and thus they will not vanish when reduced modulo $h(X)$, since $h(X)$ has degree greater than 1. From this, it follows that there are at least $\ell + 1$ distinct polynomials of degree 1 in $\mathcal{H}$. To find the desired lower bound, we count the polynomials in $\mathcal{H}$ of degree strictly smaller than t. This is the number of polynomials of the form $X^{\lambda_0}(X+1)^{\lambda_1} \cdot \ldots \cdot (X+\ell)^{\lambda_\ell}$ that have degree stricly smaller than t, i.e. such that $\lambda_0 + \lambda_1 + \ldots + \lambda_\ell \le t - 1 < t$. In other words, choosing a polynomial of degree strictly

[23]Indeed, since $n \in \mathcal{G}$, then $\mathcal{G}$ contains an element of order larger than $\log^2(n)$, hence $t = |\mathcal{G}| > \log^2(n)$.

[24]To see that $\sqrt{r}\log(n) < r$, we use the assumption that $\log^2(n) \le \mathrm{ord}_r(n) < r$ and take square roots on both sides. So $\log(n) < \sqrt{r}$ and hence $\sqrt{r}\log(n) < \sqrt{r}\sqrt{r} = r$.

smaller than t in $\mathcal{H}$, is equivalent to choosing $\ell+1$ integers $\lambda_0, \dots \lambda_\ell$ of sum less than $t-1$. Now, by Lemma 3.16, there are at least

$$\binom{t+\ell}{\ell+1} = \binom{t+\ell}{(t+\ell)-(t-1)} \underset{(8)}{=} \binom{t+\ell}{t-1}$$

such integers, and therefore $\mathcal{H}$ contains at least $\binom{t+\ell}{t-1}$ polynomials of degree strictly less than t. $\square$

But under the assumption that n is not a power of p (that is $n \neq p^k$ for any k), we can also find an upper bound for the size of $\mathcal{H}$. This is done in the following claim.

Claim 7. *If n is not a power of p, then*

$$|\mathcal{H}| \leq n^{\sqrt{t}}.$$

Proof. We first consider the subset of $\mathcal{I}$ defined by

$$\widetilde{\mathcal{I}} = \{(n/p)^i p^j : 0 \leq i, j \leq \lfloor\sqrt{t}\rfloor\}.$$

By assumption, since n is not a power of p, the set $\widetilde{\mathcal{I}}$ has $(\lfloor\sqrt{t}\rfloor+1)^2$ distinct elements, and we note that $(\lfloor\sqrt{t}\rfloor+1)^2 > t$, so that $|\widetilde{\mathcal{I}}| > |\mathcal{G}|$. By the pigeon hole principle[25], at least two elements in $\widetilde{\mathcal{I}}$ correspond to one element in $\mathcal{G}$, i.e. they must be equal modulo r. Call these two elements α, β, with $\alpha > \beta$ (without loss of generality). Then $X^\alpha \equiv X^\beta \pmod{X^r-1, p}$.
Let now $f(X) \in \mathcal{P}$. Clearly, since $\widetilde{\mathcal{I}}$ is a subset of $\mathcal{I}$, $\widetilde{\mathcal{I}}$ is also introspective for $\mathcal{P}$ and therefore

$$f(X)^\alpha \equiv f(X^\alpha) \equiv f(X^\beta) \equiv f(X)^\beta \pmod{X^r-1, p}.$$

So $f(X)^\alpha \equiv f(X)^\beta$ in the field K, and $f(X)$ is a root of the polynomial $g(Y) = Y^\alpha - Y^\beta$ in $K[Y]$. Then, since $f(X)$ is a root of $g(Y)$ and since $f(X)$ was chosen arbitrarily in $\mathcal{H}$, we see that $g(Y)$ admits at least $|\mathcal{H}|$ distinct roots in K. But, using the degree of $g(Y)$, we observe that

$$\deg g(Y) = \alpha \leq \max(\widetilde{\mathcal{I}}) = \left(\frac{n}{p}\cdot p\right)^{\lfloor\sqrt{t}\rfloor} \leq n^{\sqrt{t}}.$$

Now we use the fact that a polynomial of degree d can have at most d roots. Applying this to the polynomial $g(Y)$, we obtain by the above observations, that $|\mathcal{H}| \leq n^{\sqrt{t}}$, what was claimed. $\square$

To summarize Claims 6 and 7 in one result, we can write the estimate

$$\binom{t+\ell}{t-1} \leq |\mathcal{H}| \leq n^{\sqrt{t}}.$$

After this sequence of claims, we are now in a position to prove the converse of Lemma 3.35, that is the following.

Lemma 3.39. *If* AKS(n) *returns* `Prime`*, then n is a prime number.*

Proof. Assume that AKS(n) returns `Prime`. Then by Claim 6 and Lemma 3.18,

$$|\mathcal{H}| \geq \binom{t+\ell}{t-1} \geq \binom{\lfloor\sqrt{t}\log(n)\rfloor+1+\ell}{\lfloor\sqrt{t}\log(n)\rfloor},$$

[25]A possible formulation of this principle reads as follows: If n objects are placed among m places, with $n > m$, then one place recieves at least two objects. For instance, among a group of 366 people, at least two people must have the same birthday (if February, 29 is not counted).

since $t > \log^2(n)$ implies that $t > \sqrt{t}\log(n)$.[26] Since $\mathcal{G}$ is a subgroup of $(\mathbb{Z}/r\mathbb{Z})^\times$, as observed in (25), we have by Lagrange theorem, that $t = |\mathcal{G}|$ divides $\varphi(r) = |(\mathbb{Z}/r\mathbb{Z})^\times|$. Thus $\varphi(r) \geq t$, and moreover $\ell = \lfloor\sqrt{\varphi(r)}\log(n)\rfloor \geq \lfloor\sqrt{t}\log(n)\rfloor$. Therefore, again by Lemma 3.18 we have

$$\binom{\lfloor\sqrt{t}\log(n)\rfloor + 1 + \ell}{\lfloor\sqrt{t}\log(n)\rfloor} \geq \binom{2\lfloor\sqrt{t}\log(n)\rfloor + 1}{\lfloor\sqrt{t}\log(n)\rfloor}.$$

Now, since $\lfloor\sqrt{t}\log(n)\rfloor \geq 2$ we can use the estimate of Lemma 3.17, with $k = \lfloor\sqrt{t}\log(n)\rfloor$, to see that

$$\binom{2\lfloor\sqrt{t}\log(n)\rfloor + 1}{\lfloor\sqrt{t}\log(n)\rfloor} > 2^{\lfloor\sqrt{t}\log(n)\rfloor+1},$$

and

$$2^{\lfloor\sqrt{t}\log(n)\rfloor+1} \geq 2^{\lfloor\sqrt{t}\log(n)\rfloor} = 2^{\lfloor\log(n^{\sqrt{t}})\rfloor} = n^{\sqrt{t}}.$$

So we have shown the estimate $|\mathcal{H}| > n^{\sqrt{t}}$. But, if n is not a power of p, we have by Claim 6, that $|\mathcal{H}| \leq n^{\sqrt{t}}$. Thus, n must be a power of p, say $n = p^k$ for some $k \geq 1$. If $k > 1$, then n is a perfect power and step **I** would return `Composite`, which is impossible by assumption. Therefore $k = 1$, and hence $n = p$, which proves that n is prime. $\square$

The proofs of Lemma 3.35 together with Lemma 3.39 show Theorem 3.34.

3.4.2 Running time

After having studied the theoretical part of the AKS primality test, we will embark the study of its complexity. Before getting started, we introduce a variation of the big O notation, namely the soft O notation, denoted by $\tilde{O}$. Let f and g be two functions, then we write that $f(n) = \tilde{O}(g(n))$ if $f(n) = O(g(n)\log^k(g(n)))$ for some $k \in \mathbb{N}$. We will make much use of this notation in the following.

Let us now move to the time complexity result for the AKS test. Before stating and proving the result, we shortly summarize without proof some results, that are already known, in the following remark, for which we follow principally [5].

Remark 3.40. We gather the running times of some standard computations.

- Addition, multiplication and division of two integers a, b with $|a|, |b| \leq n$ require time $\tilde{O}(n)$.
- Addition, multiplication and division of two polynomials $f(X), g(X)$ of degree d with coefficients with absolute value less than or equal to n, can be performed in time $\tilde{O}(dn)$.
- Let $n \geq 2$ be an integer. Then it can be checked if there exist integers $a, b > 1$ such that $n = a^b$ (i.e. n is a perfect power) in time $\tilde{O}(\log^3(n))$.
- Let a, b be positive integers. Then $\gcd(a, n)$ can be computed in time $O(\log(n))$. This can for instance be done by the Euclidean Algorithm.

The result then follows in the coming theorem, from which we see that the algorithm runs in polynomial time.

Theorem 3.41. *Let $n \in \mathbb{N}, n \geq 2$. The running time of* AKS(n) *is* $\tilde{O}(\log^{21/2}(n))$.

[26]Indeed, if $t > \log^2(n)$, then $\sqrt{t} > \log(n)$. Then $t = \sqrt{t}\sqrt{t} > \sqrt{t}\log(n)$.

Proof. In the first step, we need to check whether n is a perfect power, and this can be done in $\tilde{O}(\log^3(n))$ time, by Remark 3.40.

In step **II**, we have to find r such that $\operatorname{ord}_r(n) > \log^2(n)$. In order to do this, we try out successive values for r until we find one such that $n^k \not\equiv 1 \pmod r$ for all $k \leq \log^2(n)$. For a particular r, this step involves at most $O(\log^2(n))$ multiplications modulo r, and each product will have factors less than r after reduction modulo r. Therefore, each such multiplication takes time $\tilde{O}(\log(r))$, and thus the computations on each r involve running time $\tilde{O}(\log(r))\log^2(n) = \tilde{O}(\log(r)\log^2(n))$. But in Lemma 3.38, we saw that $r \leq \max\{3, \lceil \log^5(n) \rceil\}$, and thus only $O(\log^5(n))$ different values of r need to be tested, and therefore the time complexity of step **II** is reduced to $\tilde{O}(\log^7(n))$.

Step **III** computes the greatest common divisor of r numbers, since we compute $\gcd(a, n)$ for $a \leq r$, where r is the integer found in step **II**. By Remark 3.40, each computation takes time $O(\log(n))$, and since we have r such computations, the time complexity of this step is $O(r\log(n)) = O(\log^6(n))$, by our bound on r in Lemma 3.38.

The fourth step of the algorithm consists in a comparison of n and r, which could be done by comparing the number of digits of n and r, therefore the time is proportional to the number of binary digits. So this step runs in time $O(\log(n))$.

Step **V** is the main step, in which we need to check $\ell = \lfloor \sqrt{\varphi(r)}\log(n) \rfloor$ polynomial congruences. Such an equation involves $O(\log(n))$ multiplications of polynomials of degree smaller than or equal to r (since each product is taken modulo $X^r - 1$ and n) and with coefficients less than or equal to n, so that each equation among the ℓ equations can be verified in $\tilde{O}(r\log^2(n))$ steps. Since there are ℓ equations to be verified like this, the running time of step **V** is given by

$$\tilde{O}(r\log^2(n)\sqrt{\varphi(r)}\log(n)) = \tilde{O}(r\sqrt{\varphi(r)}\log^3(n)) = \tilde{O}(r^{3/2}\log^3(n)),$$

and by the bound on r given in Lemma 3.38, this is time $\tilde{O}(\log^{21/2}(n))$.

The latter time clearly dominates all the other times, and is thus the asymptotic time complexity of the algorithm. □

Let us now discuss how one could improve this time complexity, in the following section.

3.4.3 Improvements of the running time

The time complexity obtained in Theorem 3.41 can be improved by modifying the estimate for r given in Lemma 3.38. The best case would be when $r = O(\log^2(n))$, which would reduce the running time of the algorithm to $\tilde{O}(\log^6(n))$. There is among others, one particular conjecture that would justify the existence of such an r, namely *Sophie Germain*[27]'s conjecture, that we write below. Another improved running time is obtained by a result due to Fouvry[28], managing to reduce the time complexity to $\tilde{O}(\log^{15/2}(n))$.

A first improvement. We look more precisely at the improvement that yields Sophie Germain's conjecture. Before giving the conjecture, we write down a preliminary definition that relies on it.

Definition 3.42. Let p be a prime number. Then p is called a *Sophie Germain prime* if $2p + 1$ is a prime number. In this case the number $2p + 1$ is called a *safe prime.*

Example 3.43. Let us see an example to understand the definition. Take $p = 131$, then $2p+1 = 263$, which is also prime, so 131 is a Sophie Germain prime and 263 is its associated safe prime. The Sophie Germain primes less than 100 are $2, 3, 5, 11, 23, 29, 41, 53, 83, 89$. Computing $2p + 1$ for each of these numbers, we obtain the first safe primes.

[27]Marie-Sophie Germain (1776 – 1831) was a French mathematician, physicist and philosopher.
[28]Etienne Fouvry, a French mathematician who is still alive

It is conjectured that there are infinitely many Sophie Germain primes (which is quite clear by intuition, since there are infinitely many prime numbers), but this has not yet been proven. Sophie Germain formulated a conjecture about the density of such primes, that is the number of such primes up to a given bound.

Conjecture 3.44 (Sophie Germain's Prime Density Conjecture)**.** *Let $n \in \mathbb{N}, n \geq 2$. The number of Sophie Germain primes less than n is asymptotically*

$$2C\frac{n}{\ln^2(n)},$$

where C is the twin prime[29] *constant defined, for prime p, by*

$$C = \prod_{p\geq 3} \frac{p(p-2)}{(p-1)^2} \approx 0.66.$$

Example 3.45. For $n = 100$, Sophie Germain's conjecture predicts approximatively 7 Sophie Germain primes less than 100, which has 30% error compared to the exact value of 10, seen in Example 3.43. For larger n, this error decreases, but is also harder to compute, since we need to know how many Sophie Germain primes less than n there are indeed. For instance, if $n = 10^7$, it can be verified that the prediction by Conjecture 3.44 is still 10% off the real number.

Even though this conjecture has not yet been shown, we can say that if it held, then it would be possible to conclude that $r = \tilde{O}(\log^2(n))$, managing to reduce the running time of the AKS algorithm to time $\tilde{O}(\log^6(n))$.

Indeed, assuming that Conjecture 3.44 holds, we could find some constant c such that there are at least $\log^2(n)$ Sophie Germain primes between $8\log^2(n)$ and $c\log^2(n)\log^2(\log(n))$. Then for any such Sophie Germain prime p, denote by q its associated safe prime, that is $q = 2p + 1$. We look at the order of n modulo q. Since $\mathbb{F}_q^\times$ has order $q - 1 = 2p$, either $\text{ord}_q(n) \leq 2$ or $\text{ord}_q(n) \geq (q-1)/2$. If $\text{ord}_q(n) \leq 2$, then q must divide $n^2 - 1$ (since if $\text{ord}_q(n) = 1$, then $q \mid (n-1)$ and if $\text{ord}_q(n) = 2$, then $q \mid (n^2-1)$, so in any case $q \mid (n^2-1)$) and thus the number of such q is bounded by $O(\log^2(n))$, by Conjecture 3.44. Therefore there must exist a prime number $r = \tilde{O}(\log^2(n))$ so that $\text{ord}_r(n) > \log^2(n)$, as required in the second step of the AKS test, and such a value for r will make the test run in time $\tilde{O}(\log^6(n))$.

A second improvement. In this part, we will study the optimized running time due to a result proved by Etienne Fouvry. Let us first fix some notations. Let $n \in \mathbb{N}$. We denote by $P(n)$ the greatest prime divisor of n (i.e. $P(n) = \max\{p \text{ prime} : p \mid n\}$) and define the set of prime numbers

$$A = \left\{p \text{ prime} : p \leq x, P(p-1) > p^{2/3}\right\}, \tag{27}$$

for some positive real valued x. The result that we are to discuss is based on the following lemma, of which we will skip the proof, since we are rather interested in its consequence.

Let us see an example to understand the description of the set that we just defined.

Example 3.46. Take $x = 50$. Then A is the set of all prime numbers p up to 50 such that $p-1$ have greatest prime divisor greater than $p^{2/3}$. For this, we exclude the prime 2, since $P(1)$ is not defined. Table 3.1 gathers these information, allowing to descreibe explicitely the set A.

To write the set A explcitely, we compare the two last columns for each line, and look whether $P(p-1) > p^{2/3}$. If this is the case, then p belongs to A, otherwise not. Doing so, it comes that for $x = 50$, A is the set $A = \{11, 23, 47\}$.

[29]We recall the definition of a twin prime: A prime number p is called a *twin prime* if either $p-2$ or $p+2$ is a prime number. Often the name twin prime is used to refer to a pair of twin primes $(p, p+2)$.

p	$p-1$	$P(p-1)$	$p^{2/3}$	p	$p-1$	$P(p-1)$	$p^{2/3}$
3	2	2	2.08	23	22	11	8.086
5	4	2	2.924	29	28	7	9.439
7	6	3	3.659	31	30	5	9.868
11	10	5	4.642	37	36	3	11.104
13	12	3	5.529	41	40	5	11.89
17	16	2	6.611	43	42	7	12.279
19	18	3	7.12	47	46	23	13.024

Table 3.1: Construction of the set A for $x = 50$

Fouvry proved the following lemma, giving a bound for the cardinality of the set A.

Lemma 3.47. *Let p be a prime number and $x \in \mathbb{R}_+$. There exist constants $c \in \mathbb{R}_+^*$ and $n_0 \in \mathbb{N}$ such that for all $x \geq n_0$,*

$$|A| \geq c\,\frac{x}{\ln(x)},$$

where A is the set defined in (27)*.*

Remark 3.48. The prime number Theorem states that the number of primes p smaller than or equal to x is asymptotically $x/\ln(x)$. Lemma 3.47, with constant $c = 1/N$ for some $N \in \mathbb{N}$, means that in average all Nth prime number lies in A.

With this lemma, it is possible to obtain a better running time than in Theorem 3.41, namely the one described in the following theorem.

Theorem 3.49. *Let $n \in \mathbb{N}, n \geq 2$. The running time of* AKS(n) *is $\tilde{O}(\log^{15/2}(n))$.*

Proof. By Lemma 3.47, a high density of primes p such that $P(p-1) > p^{2/3}$ implies that an r such that $\mathrm{ord}_r(n) > \log^2(n)$ (as required in step **II** of the AKS test) can be found with $r = O(\log^3(n))$. Indeed, if $r \in A$, then $P(r-1) > r^{2/3}$, and we can identify $(\mathbb{Z}/r\mathbb{Z})^\times$ with $C \times R$, where C is a cyclic group of order at least $2/3$ and R is the remaining component. Then $n \in (\mathbb{Z}/r\mathbb{Z})^\times$ (Claim 2) can be viewed as the couple $(n_1, n_2) \in C \times R$. If $n_1 \neq 1$, then $\mathrm{ord}_r(n) \geq r^{2/3}$, and we observe that choosing $r = O(\log^3(n))$ is convenient, since then $\mathrm{ord}_r(n) \geq r^{2/3} > \log^2(n)$. In other words, by testing the primes greater than $\log^3(n)$ one after the other, we find one lying in A and such that $n_1 \neq 1$. By Lemma 3.47, this is instantanous and the condition $n_1 \neq 1$ will be satisfied with probability greater than $1/2$. This then reduces the running time to

$$\tilde{O}(r^{3/2}\log^3(n)) = \tilde{O}((\log^3(n))^{3/2}\log^3(n)) = \tilde{O}(\log^{15/2}(n)),$$

as was claimed. □

Many other attempts. Although we saw in more details essentially two results of improved running times for the AKS test, we should note that there were many other attempts for finding better running times. For instance one of its authors himself, Agrawal, presented a conjecture that would invole the running time of the algorithm go down to $\tilde{O}(\log^3(n))$, but his non-proven conjecture seems to be false, according to Hendrik Lenstra[30] and Carl Pomerance[31], who presented a counter argument to it.

[30]Hendrik Lenstra (born in 1949) is a Dutch mathematician specialized in number theory.
[31]Carl Pomerance (born in 1944) is an American number theorist.

4 Implementations

We chose SAGE for implementing the Miller-Rabin and AKS primality tests. In this section we provide a possible implementation and give some explanations to several steps. We split up this section into two distinct parts, one treating the programmation of the Miller-Rabin primality test, and a second part dealing with the implementation of the AKS test.

4.1 Implementation of the Miller-Rabin primality test

In order to implement the test according to Algorithm 2.1 page 12, we have created two auxiliary functions, namely:

(1) `decompose`(n), that returns integers r and k of the decomposition of n as $2^r k$ with integers r and odd k.

(2) `random_a`(n), that generates a random a in $(\mathbb{Z}/n\mathbb{Z})^\times$.

Clearly, one could directly implement the lines of code of these functions in the main algorithm, but by doing it in this way, the Miller-Rabin algorithm will be easier to read and more compact.

Here are the codes of corresponding to the above enounced functions.

```
def decompose(n):
  r=0, k=n
  while (k % 2==0):
      k=k/2
      r=r+1
  return (r,k)
```

```
def random_a(n):
  a=int(random()*n)
  while gcd(a,n)<>1:
      a=int(random()*n)
  return a
```

Then the implementation of the Miller-Rabin test can look as follows.

```
1  def Miller_Rabin(n,t):
2    if (n%2==0) or (n%3==0) or (n%5==0):
3        return "composite"
4    (r,k)=decompose(n-1)
5    for i in [1..t]:
6         a=random_a(n)
7         b=Mod(a,n)^k
8         if (b!=1):
9             j=0
10            while (j<=r-1) and (b!=n-1):
11                b=b^2%n
12                j=j+1
13            if (b!=n-1):
14               return "composite"
15   return "probably prime"
```

Remark 4.1. We list some general observations and remarks.

- The parameter t is, as we defined it in the theory, the number of times we want the test to run. Remember that the larger we choose it, the more probable it will be that n is prime when MILLER-RABIN(n,t) returns `Probably Prime` (see Theorem 2.17 page 16). The `for` loop in line 5 makes that the main condition of the test is verified t times.

- As we saw in step **II** of algorithm 2.1, we need to decmpose $n-1$ as $2^r k$ with integers r and odd k. Therefore we evaluate the previaosly defined auxiliary function `decompose` at $n-1$. This is done in line 4.
- As wished, we want to choose $a \in (\mathbb{Z}/n\mathbb{Z})^\times$ at random. This is done by the previously defined function `random_a`(n). Since this computation sits in the `for` loop in line 5, t such values for a will be generated to verify the main condition of the test.
- Step **III** of Algorithm 2.1 requires to compute $a^k \mod n$ and we already understood that this is most efficiently done by fast exponentiation. However, SAGE already has the own implemented command `Mod(a,n)^k` to perform this calculation rapidly, i.e. by fast exponentiation. This is why it suffices to define b as we did in line 7.
- In line 8 we test whether b defined above equals 1. Note that if this is the case, then lines 9 up to 14 are ignored by the program, and thus the test returns `Probably Prime` in line 15.
- If b is not equal to 1, then we set up the sequence of squares of the number a^k, and reduce each term modulo n (line 11). This is performed by the `while` loop in line 10, where j is the exponent that increases to come from one term of the sequence to the next and runs through the integers $0, 2, \ldots, r-1$, as seen in Proposition 2.12 page 13. This loop runs as long as a term of the created sequence is not congruent to -1 (note that $n-1 = -1$ modulo n). If none of the terms of the sequence is congruent to -1 modulo n, then n must be composite (lines $13, 14$).

Let us now see some results of this implementation. The following table gathers the results obtained for certain values of n for `decompose`(n). The second column shows the pair (r, k) in the decomposition of n as $2^r k$.

n	`decompose`(n)	Interpretation
198656	$(11, 97)$	$198656 = 2^{11} \cdot 97$
587400	$(3, 73425)$	$587400 = 2^3 \cdot 73425$
1441792	$(17, 11)$	$1441792 = 2^{17} \cdot 11$
240518168576	$(35, 7)$	$240518168576 = 2^{35} \cdot 7$
113715890591105024	$(50, 101)$	$113715890591105024 = 2^{50} \cdot 101$

Table 4.1: Some compilations for the function `decompose`(n)

Table 4.2 shows some results of the Miller-Rabin test, with entrance (n, t), where n is the integer to be checked for primality and t is the bound of number of times we want to go through the test.

n	t	`Miller_Rabin`(n, t)
95	1	Composite
97	5	Probably prime
97	10	Probably prime
101	25	Probably prime
769	50	Probably prime
9787	100	Probably prime
100271	20	Probably prime
17017	20	Composite

Table 4.2: Some compilations for the function `Miller_Rabin`(n, t)

There are some interesting remarks to make on these examples. The test Miller-Rabin$(95, 1)$ did not need to run long, since 95 was immediately detected as multiple of 5 right away in the first step. Therefore, our choice of t was not important, and this result would have been returned for any t. As we also see in Table 4.2, we ran twice the Miller-Rabin test to check the primality of 97, that is firstly Miller-Rabin$(97, 5)$, and secondly Miller-Rabin$(97, 10)$. Since in these conditions, the algorithm returned `Probably Prime`, Theorem 2.17 page 16 gives for $t = 5$, a probability of correctness at least equal to $1-(1/4)^5 \approx 0.999023$, and then for $t = 10$, a probability more than $1-(1/4)^{10} \approx 0.999999$, which makes us quite sure that 97 is a prime number, which indeed is. Analogously, the test Miller-Rabin$(101, 25)$ shows that 101 is a prime number with probability at least $1 - (1/4)^{25} \approx 1$. It results that even more probably, 769 is a prime number, due to the test Miller-Rabin$(769, 50)$, with 50 repetions, giving a correct result with probability at least $1 - (1/4)^{50}$. Finally, note that the test also works for larger numbers, as we see on the tests Miller-Rabin$(9787, 100)$ and Miller-Rabin$(100271, 20)$, both returning `Probably Prime`. It turns out that this is indeed a prime number, and since $t = 100$ for the first, we are get really close to a correct output. Also the algorithm correctly identified 17017 as composite, which eqauls $7 \cdot 11 \cdot 13 \cdot 17$, thus with no multiple of $2, 3$ or 5, meaning that the output does not follow from the first step, but from line 14. It could be the case that the test returned `Probably Prime` for this test, since it is probabilistic.

4.2 Implementation of the AKS primality test

We come right now to the implementation of the AKS test described in Algorithm 3.1 on page 25. Analogously to the previous implementation, we created some auxiliary functions to make the main code easier readable. We defined the following auxiliary functions:

(1) `Perfect_power`(n), that checks whether the given integer n is a perfect power, accordingly to Definition 3.30 given on page 25.

(2) `Find_smallest_r`(n), that returns the smallest integer r such that $\mathrm{ord}_r(n) > \log^2(n)$.

(3) `Check_l_equations`(n), that checks $\ell = \lfloor \sqrt{\varphi(r)} \log(n) \rfloor$ equations, that are precisely the congruences

$$(X + a)^n \not\equiv X^n + a \pmod{X^r - 1, n} \ , \ a = 1, \dots, \ell. \tag{28}$$

We define the function to return `True` when (28) holds, and `False`, otherwise.

Note that the first function could be of general use, whereas two other functions are rather related to the AKS primality test. Let us now come to the line codes of these three functions.

(1) To test whether a known integer n is a perfect power, we test whether n can be written as a^b for some integers $a, b > 1$. If so, then n is a perfect power, namely a's bth power. Otherwise, it is not.
The idea of the algorithm is that if $n = a^b$ for some $a, b > 1$, then $b \leq \lfloor \log(n) \rfloor$. So, we need to test whether $n = a^b$ for some integers $a > 1$ and $1 < b \leq \lfloor \log(n) \rfloor$. We define $y = \log(n)/b$ and $a = 2^y$. Then, indeed $a^b = (2^y)^{\log(n)/y} = n$. Here comes the algorithm that fits this description.

```
def Perfect_power(n):
  for b in range(2,ceil(log(n,2))+1):
      y=(log(n,2)/b).n()
      c=(pow(2,y)).n()
      if (pow(floor(c),b)==n) :
          return True
  return False
```

Note that the extension `n.()` in lines 3 and 4 is used to obtain approximative values. SAGE's function `pow(a,b)` performs a^b and we thus used `pow(2,y)` to compute $a = 2^y$.

Let us see some examples obtained by this implementation.

n	`Perfect_power`(n)	Decomposition
16	True	$16 = 2^4$
243	True	$243 = 3^5$
4096	True	$4096 = 8^4$
2357947691	True	$2357947691 = 11^9$
8835468	False	

Table 4.3: Some compilations for the function `Perfect_power`(n)

Note that it also works for larger numbers (that we did not put in the table for estetic reasons) such as the number

$$n = 3733456322341571760420937867518819428980350494384765625,$$

which is also identified as perfect power by the proposed algorithm, and indeed $n = 105^{27}$.

It is important to remark that the test that we implemented does not yield the decomposition of n as a^b, but only claims whether n is or not a perfect power.

(2) As required in step **II** of Algorithm 3.1, we look for the smallest integer r so that n has order greater than $\log^2(n)$ modulo r. In Lemma 3.38 page 27 we gave a bound to the values of r, by saying that such an existing r satisfies $r \leq \max\{3, \lceil \log^5(n) \rceil\}$. Therefore, it is convenient to test all integers $2 \leq r \leq \lceil \log^5(n) \rceil$ to see if there exists $1 \leq k \leq \log^2(n)$ such that $n^k \equiv 1 \pmod r$, that is $\operatorname{ord}_r(n) = k$. If no such k can be found, then we are done, by taking that particular r, since then $\operatorname{ord}_r(n) > \log^2(n)$, as we required. Note that the idea of this implementation served us in the proof of Theorem 3.41 page 32, to show the standard running time of step **II** of the test. The algorithm could look as follows.

```
1  def Find_smallest_r(n):
2    m=max(3,floor(log(n,2)^5))
3    r=2
4    while(r<=floor(m)):
5        c=0
6        b=floor(log(n,2))^2
7        for k in [1..b]:
8            if ((n^k)%r==1):
9                c=c+1
10       if(c==0):
11           break
12       r=r+1
13   return r
```

In line 5, we define a counter c, that counts each time that $n^k \equiv 1 \pmod r$. If for particular r, $n^k \equiv 1 \pmod r$ then c increases by 1, as we see in lines 8, 9. Otherwise, c does not evolve and the program stops and returns that particular r (see lines 10, 11).

Let us also give here some examples of such values of r found for given n, in Table 4.4. The second column contains approximative values for $\log^2(n)$ for each n. So, the condition to be fulfilled is that the order of n mudulo r is strictly greater than the value in the second column, and the third column gives the smallest possible such r, found by the function `Find_smallest_r`(n).

n	$\log^2(n)$	`Find_smallest_r`(n)
5	5.391	5
15	15.264	15
500	80.385	2
1235	105.479	5
9854531	539.742	197
36952741	631.978	7
700745415221	1548.430	1531

Table 4.4: Some compilations for the function `Find_smallest_r`(n)

From these examples, we observe that there is no order on the values of r found by the function `Find_smallest_r`(n) in the third column. For example for $n = 500$, the computed r is smaller than that for $n = 15$, although 500 is greater than 15.

(3) This is the main and final part of the AKS test. Having found r by the algorithm above, we define $\ell = \lfloor\sqrt{\varphi(r)}\log(n)\rfloor$, and we check the conditions described in (28). This is equivalent to check whether $X^n + a = (X + a)^n$ in the polynomial ring $(\mathbb{Z}/n\mathbb{Z})[X]/(X^r - 1)$. After having defined the polynomial ring, for which SAGE does a lot of work itself, the algorithm goes straight forward.

```
1  def Check_l_equations(n):
2    r=Find_smallest_r(n)
3    l=floor(sqrt(euler_phi(r))*log(n,2))
4    R.<x>=PolynomialRing(Integers(n))
5    S=R.quotient((x^r)-1)
6    for a in [1..l]:
7        f=S((x+a)^n)
8        e=Mod(n,r)
9        g=(x^e)+a
10       if (f!=g):
11           return True
12       else:
13           return False
```

We see that Euler's totient function is already known by SAGE, through the command `euler_phi()`. Note now the simplicity to implement the above described polynomial ring. The instruction `R.<X> = PolynomialRing(Integers(n))` in line 4 defines $R = \mathbb{Z}/n\mathbb{Z}[X]$, which then is reduced modulo $X^r - 1$ by the instruction `R.quotient((X^r)-1)` in line 5. Finally, S contains the ring $(\mathbb{Z}/n\mathbb{Z})[X]/(X^r - 1)$. In line 6, a runs from 1 to ℓ in order to check the ℓ equations $f = g$ in the ring S, with $f = (X+a)^n$ and $g = X^n+a$. To conclude, we say that if $f \neq g$, then we return `True`, and if $f = g$, we return `False`. Remark that the result could be interchanged as well, since the aim is only to see whether the congruence holds or not.

Having implemented these three auxiliary functions as above, the work for implementing AKS test reduces notably and is much easier to read. One could write it as follows.

```
1  def AKS(n):
2    if Perfect_power(n):
3        return "composite"
4    r=Find_smallest_r(n)
5    for a in [1..r]:
6        if (1<gcd(a,n)<n):
7            return "composite"
8    if (n<=r):
9       return "prime"
10   if Check_l_equations(n):
11      return "composite"
12   return "prime"
```

Remark 4.2. Let us also here write down some general observations.

- Step **I** is simply checked by calling the `Perfect_power`(n) function previously defined. If then n turns out to be a perfect power, then it is composite (see lines $2, 3$).
- To find in step **II** the smallest r that verifies $\mathrm{ord}_r(n) > \log^2(n)$, we call the second auxiliary function `Find_smallest_r`(n) that we defined previously (see line 4).
- The next step is to check whether n has a prime factor smaller than r. Since we do not have any direct possibility to go through the prime numbers, other than using SAGE's `is_prime` function, which is of course not really the sense of the job since we would use a primality test to create another primality test. But nevertheless, it does not take much more time to run through all the integers up to r, since values of r are often small compared to n (see Table 4.4). These are the instructions in the `for` loop in line 5, which considers all integers smaller than r. Moreover, if n is itself smaller than r, then n is composite.
- Finally, we apply the last auxiliary function `Check_l_equations`(n) to test whether the ℓ equations described in (28) hold. Line 10 asks whether $f \neq g$, for f and g defined as before. By the choice we made (to decide that in this case `Check_l_equations`(n) returns `True`), the algorithm must then return `Composite`.

The following table shows some examples obtained by this implementation.

n	`AKS`(n)
13	Prime
27	Composite
121	Composite
881	Prime
6917	Prime
28657	Prime
16785407	Prime
50054784687	Composite

Table 4.5: Some compilations for the function `AKS`(n)

We see that our AKS implementation also works on larger numbers, however needs more time to compile. The `timeit(' ',seconds=True)` command returns the time in seconds that the test needs to give an output. For instance,

```
timeit('AKS(16785407)',seconds=True)
```

returned 30.4267636299, meaning that this test needed about 30 seconds to return `Prime` as we see in Table 4.5.

References

[1] Andrew Granville, It is easy to determine whether a given integre is prime
Bulletin of the American mathematical society, Vol. 42, Num. 1, Pages 3 – 38

[2] Gabor Wiese, Théorie des nombres et applications à la cryptographie
Notes de cours, version du 13 juin 2014, Université du Luxembourg

[3] Julien Élie, L'algorithme $\mathcal{AKS}$ ou Les nombres premiers sont de classe $\mathcal{P}$
http://www.trigofacile.com/maths/curiosite/primarite/aks/pdf/algorithme-aks.pdf

[4] Korselt's Criterion for Carmichael numbers, Math 4150, Spring 2011
http://people.math.gatech.edu/~mbaker/pdf/korselt.pdf

[5] Manindra Agrawal, Neeraj Kayal, Nitin Saxena, PRIMES is in P
Annals of Mathematics, Vol. 160, Pages 781 – 793, 2004

[6] SAGE tutorials and documentation
http://sagemath.org/doc/

[7] Underwood Dudley, A guide to Elementary Number Theory
Dolciani Mathematical Expositions, Chapter 29, Pages 103-104

[8] Vijay Menon, Deterministic Primality Testing
http://arxiv.org/pdf/1311.3785.pdf

[9] Wikipedia, the free encyclopedia
https://en.wikipedia.org/wiki/

[10] Yimin Ge, Elementary Properties of Cyclotomic Polynomials
http://www.yimin-ge.com/doc/cyclotomic_polynomials.pdf