Primality and Factoring

Anuj Jakhar

A dissertation submitted for the partial fulfilment of
BS-MS dual degree in Science

Indian Institute of Science Education and Research Mohali
April 2014

Certificate of Examination

This is to certify that the dissertation titled **"Primality and Factoring"** submitted by **Mr. Anuj Jakhar** (Reg. No. MS09020) for the partial fulfilment of BS-MS dual degree programme of the Institute, has been examined by the thesis committee duly appointed by the Institute. The committee finds the work done by the candidate satisfactory and recommends that the report be accepted.

Dr. Alok Maharana Dr. Chanchal Kumar Prof. Kapil H. Paranjape
(Thesis supervisor)

Dated: April 25^{th}, 2014

Declaration

The work presented in this dissertation has been carried out by me under the guidance of Prof. Kapil H. Paranjape at the Indian Institute of Science Education and Research Mohali.

This work has not been submitted in part or in full for a degree, a diploma, or a fellowship to any other university or institute. Whenever contributions of others are involved, every effort is made to indicate this clearly, with due acknowledgement of collaborative research and discussions. This thesis is a bonafide record of original work done by me and all sources listed within have been detailed in the bibliography.

Anuj Jakhar

Dated : April 25^{th}, 2014

In my capacity as the supervisor of the candidate's project work, I certify that the above statements by the candidate are true to the best of my knowledge.

Professor Kapil H. Paranjape
(Thesis supervisor)

Acknowledgment

My first and foremost thanks is to my thesis supervisor Professor Kapil H. Paranjape. It was a great experience to work in his guidance. I very much appreciate his way of handling a student's doubt by providing him/her the correct line of thought and then allowing one to figure out by oneself, as it helped me learning many new things during the process.

I am grateful to Professor K. V. Subrahmanyam (CMI, Chennai) and Dr. Amitava Bhattacharya (TIFR, Mumbai) for teaching me useful algorithms in Primality during my summer internships. I thank my committee members, Dr. Chanchal Kumar and Dr. Alok Maharana, for their insightful comments.

Nothing is possible without financial resources. I would like to sincerely acknowledge DST, Government of India for providing me INSPIRE fellowship. I am much thankful to IISER Mohali for providing me infrastructure and Computer Centre for all the technical support.

I am grateful to all my friends at IISER and otherwise, for giving me all the non-technical support and of course sheer luck because of which I got the opportunity to learn mathematics.

I am deeply thankful to Agastya P. Bhati for all his support throughout my stay at IISER.

No acknowledgement would ever adequately express my gratitude to my family. I would like to give a special mention to Sh. Sahi Ram, Supyar, Atul and Kalpana for always believing in me. Their moral support has always boosted my confidence and motivated me to achieve and their teachings have enabled me for it. It is their love and affection which gives me a reason to succeed.

Anuj Jakhar
MS09020
IISER Mohali

Notation

$\#S$: Cardinality of a set S.

$\mathbb{F}_q$: Finite field with q elements.

$\mathbb{R}$: Field of Real numbers.

If $x \in \mathbb{R}$, then :
$\lceil x \rceil$: smallest integer greater than x.
$\lfloor x \rfloor$: greatest integer smaller than x.

Given n $\in \mathbb{N}$, [n] : the set $\{1, 2, \cdots, n\}$.

Let a, b be elements of some ring R. Then :
b|a : b divides a in R , i.e. $\exists$ c $\in$ R such that a = b·c.

$\mathbb{Z}$/n$\mathbb{Z}$: ring of integers modulo n.

($\mathbb{Z}$/n$\mathbb{Z}$)* : the abelian group consisting of the units in $\mathbb{Z}$/n$\mathbb{Z}$.

Let f(x) be an irreducible polynomial in $\mathbb{F}_p[x]$. Then :
f(x) $\equiv$ g(x) (mod h(x), p) means f(x)–g(x) = 0 in the ring ($\mathbb{Z}$/p$\mathbb{Z}$)[x]/(h(x)).

(a, b) = c means gcd(a, b) = c.

o_r(p) denotes the smallest non negative integer k such that $p^k \equiv 1 (\text{mod } r)$.

If n $\in \mathbb{N}$, Then $\phi(n)$ denotes the number of integers less than or equal to n that are relatively prime to n.

Asymptotic Notation for Runtime Analysis

Throughout this thesis, all logarithms will be to the base 2 and denoted $\log(n)$ rather than $\log_2(n)$ or $\lg(n)$. Additionally, all of our runtime analysis will use "Big O" notation, which is defined as follows :
$\mathrm{O(f)} = \{g \mid \exists c, \exists n_0, for\ all\ n > n_0 : |g(n)| \leq c \cdot |f(n)|\}$
$\mathrm{O^{\sim}(f)} = \{g \mid \exists c, \exists n_0, \exists k, for\ all\ n > n_0 : |g(n)| \leq c \cdot |f(n)| log^k(|f(n)|)\}$
$\Omega\mathrm{(f)} = \{g \mid \exists c, \exists n_0, for\ all\ n > n_0 : |g(n)| \geq c \cdot |f(n)|\}$
$\Theta\mathrm{(f)} = \mathrm{O(f)} \cap \Omega\mathrm{(f)}$

Notation Given a non negative integer n, $||n||$ denote the number of bits in the binary representation of n. Then $||n|| = \lceil log(n+1) \rceil$.

Fact 1 : If $\mathrm{g_1(n)} = \mathrm{O(f_1(n))}$ and $\mathrm{g_2(n)} = \mathrm{O(f_2(n))}$
(1) $\mathrm{g_1}(n) + \mathrm{g_2(n)} = \mathrm{O}(\max\{f_1(n), f_2(n)\})$
(2) $\mathrm{g_1}(n) \cdot g_2(n) = O(f_1(n) \cdot f_2(n))$

As implication of the part (1) of the above fact is that complex algorithms can often have a single step whose complexity dominates the other steps, meaning that the overall complexity of an algorithm is sometimes simply the complexity of a single step of the algorithm.

Definition 0.0.1. *We say that an algorithm has polynomial time complexity (in input size), if for given input n, all computations are performed in $O(||n||^k)$ bit operations for some $k \geq 0$, where a bit operation refers to the number of steps needs to perform an arithmatical operation on a natural number given in binary representation.*

Lemma 0.0.2. *Let $n, m \in \mathbb{N}$. Then :*
(1) Computing m + n takes $O(||n|| + ||m||) = O(log(n)+log(m))$ bit operations.
(2) Computing m·n takes $O(||n|| \cdot ||m||) = O(log(n)\cdot log(m))$ bit operations.
(3) Computing the quotient n div m and the remainder n mod m takes $O((||n|| - ||m|| + 1) \cdot ||m||)$ bit operations.

Addition and subtraction can therefore be computed in linear time, while multiplication and division can be computed in quadratic time, which is still polynomial.

Fact 2 *: Let $m, n \in \mathbb{N}$ with at least k bits each. Then :*
(1) m and n can be multiplied with $O(k(\log(k))(\text{loglog}(k))) = O^{\sim}(k)$ bit operations.
(2) n div m and n mod m can be computed using $O(k(\log(k))(\text{loglog}(k))) = O^{\sim}(k)$ bit operations.
(3) Multiplication of two polynomials of degree d with coefficients at most m bits in size can be done in $O^{\sim}(d \cdot m)$ bit operations.

Proof. : Refer to [39] for proof of (1) and [40] for proof of (2) & (3).

Contents

Abstract

This exposition is the result of a year's study of the Primality and Factoring. It has two parts. In the first part of the thesis, we discuss the most popular methods of primality testing. After providing a brief survey of primality testing algorithms (i.e. the Chinese primality test, Fermat test, Lucas test, the Miller-Rabin primality test etc.), we present a thorough analysis of the unconditional deterministic polynomial time algorithm for determining that a given integer is prime or composite proposed by Agrawal, Kayal and Saxena in their paper "Primes is in P" [6]. In the second part of the report, we discuss the well known algorithms for integer factorization (such as Pollard rho method, Pollard p-1 method, Fermat factor base method, Continued fraction method etc.) along with intermediate steps of their formulation. At the end of this part, we present quadratic Sieve method for factoring integers in exponential time (in the input size) and number field Sieve algorithm briefly. At the end, we discuss Lenstra-Lenstra-Lovasz (LLL) - algorithm for getting reduced basis of a lattice and factoring any arbitrary non-constant polynomial in $\mathbb{Z}[X]$ in polynomial time (as in input size).

Chapter 1

Introduction

Testing whether a number is prime number or not and finding out its prime factors are the fundamental problems in computational number theory. Since ancient times, every mathematician has been fascinated by problems concerning prime numbers. Primality testing and integer factorization have wide applications in computer science, mainly in cryptography.

In the first part, we discuss the well known algorithms and address the attempts at developing an efficient and reliable method for testing primality. A primality test is a function that determines if a given integer greater than 1 is prime or composite. These tests can be subclassified as either probabilistic primality tests or deterministic primality tests. In a deterministic test when we put an integer n, then the output will be yes if the integer is prime, or no if the integer is not prime (or integer is composite). A nondeterministic primality test takes an inputted integer n and returns either no it is not prime or it may be a prime. A probabilistic primality test is a nondeterministic test that returns either that the inputted integer is not a prime or that is probably a prime to some given degree of likelihood. Every time someone uses the RSA public key cryptosystem and they need to generate a private key consisting of two large prime numbers and a public key consisting of their product. To do this, one needs to be able to check rapidly whether a number is prime.
The simplest and well-known algorithm to test whether n is prime is the Sieve of Eratosthenes. Eratosthenes lived in Greece circa 200 B.C. His method for determining primality is as follows :
Suppose we want to determine that n is prime or composite. First we make a list of integers 2, 3, $\cdots$, m where m is the largest integer less than or equal to $\sqrt{n}$. Next we

circle 2 and cross off all multiples of 2 from the list. Then we circle 3 and cross off its multiples. We now continue through the list, each time advancing to the list, integer that is not crossing off, circling that integer, and crossing off all its multiples. We then test to see if any of the circled numbers divide n. If the list of circled numbers is exhausted and no divisor found, then n is prime. The trial division algorithm is based on the simple observation that if n is composite then n has a prime factor less than or equal to $\sqrt{n}$.

As in the application to cryptography, most of the primality testing is concerned with large numbers, usually in excess of 100 digits and often much larger. If we were to use the Sieve of Eratosthenes to determine the primality of a number with just 20 digits, we would need first to find at least all the primes up to 10^{10}. There are around 450 million primes found less than 10^{10} (using "prime number theorem")[1]. At the rate of finding one prime per second, (including crossing off all of its multiples) we would be working for a little over 14 years to find 450 million primes. Certainly their are computers that can run this algorithm faster and more efficiently but the running time of this algorithm is exponential in the input size. We need quicker algorithms.

We know that if an integer is not prime then it is composite and vice versa. In 1980, Michael Rabin discovered a randomized polynomial time algorithm to test whether a number is prime or composite. It is called the Miller-Rabin primality test because it is closely related to a deterministic algorithm studied by Gary Miller in 1976. This is still the most practical known primality testing algorithm, and is widely used in software libraries.

In 2002, Manindra Agrawal, Nitin Saxena and Neeraj Kayal developed a new deterministic primality test. This test runs fairly rapidly as compare to previous deterministic primality tests. Specifically, given x bits of input, the number of steps needed by their test is bounded by some polynomial in x.

The second part is devoted to methods to generate all the prime factors of any given integer greater than 1. Although any schoolchild can multiply two integers and determine their product, determining the prime factorization of a given integer has actually been an active area of mathematical research for over 2300 years. Anybody who is doubtful about the difficulty of factoring is invited to factor the following 212 digit number, for which security company RSA is offering $30,000 :

7403756347956171282804679609742957314259318888923128908493623263897276503402826627689199641962511784399589433050212758537011896809828673317327310893090055250511687706329907239638078671008609696253793465056379635

Suppose we know a certain large odd integer n, which is composite; for example, we found that it fails one of the primality tests in part 1, which does not mean that we have any idea of what a factor of n might be. Of the methods, we have encountered for testing primality, only the very slowest algorithm - Sieve of Eratosthenes - actually gives us a prime factor at the same time as it tells us that n is composite. All the faster primality test algorithms are more indirect. They tell us that n have prime factors but not what they are.

The method of trial division by primes $< \sqrt{n}$ can take more than $O(\sqrt{n})$ bit operations. In 1975, J. M. Pollard discovered a simplest algorithm "The rho method" (also called the "Monte Carlo method") of factorization, which is substantially faster than the trail division method.

In 1980's, Pomerence developed a new method for factoring large integers, called "The quadratic Sieve method" which was more successful than any other method of factoring integers n of general type which have no prime factor of order of magnitude significantly less than $\sqrt{n}$.

Until recently, all of the contenders for the best general purpose of factoring algorithm had running time of the form $O(\exp(\sqrt{lognloglogn}))$. Some people even thought that this function of n might be a natural lower bound on the running time. However, during the last few years a new method - called the number field Sieve - has been developed that has a heuristic running time that is much better, namely $O(\exp((logn)^{1/3}$ $(loglogn)^{2/3}))$.

In the last chapter of this part, we introduce Lenstra-Lenstra-Lovasz (LLL)-algorithm for getting reduced basis of a given basis of lattice. After this, using LLL-reduction method, we discuss an algorithm for getting all irreducible factors of a non-constant polynomial which is in $\mathbb{Z}[X]$ and it has polynomial time (in input size).

Part I

Primality

Chapter 2

Primality Testing Algorithms

2.1 The Sieve of Eratosthenes Primality test and Trial Division

This is the simplest algorithm to test whether a given integer n is prime or not.

2.1.1 Algorithm (The Sieve of Eratosthenes)

Input : $n \in$ N
1: a[1$\cdots$n] integer array;
2: for j = 1 to n do
3: a[j] $\leftarrow$ j;
4: i $\leftarrow$ 2;
5: while $i^2 < n$ do
6: if a[i] $\neq$ 0 then
7: t $\leftarrow$ 2.i;
8: while t $\leq n$ do
9: a[i] $\leftarrow$ 0; t $\leftarrow$ t+i;
10: i $\leftarrow$ i+1;
11. for j = 2 to n do
12. if a[j] $\neq$ 0 then return a[j] is **prime**.

Sieve method has $O(10^{log_{10}(n)})$ operations to prove primality if n has $\log_{10}$n digits. We can say that this algorithm is of exponential time in terms of the input length.

Similarily :

2.1.2 Algorithm (trial division)

---

(1) For k = 2, 3, 4, $\cdots$, $\lfloor\sqrt{n}\rfloor$.
(2) Test if (for any k),
$n \equiv 0$ (mod k), output **composite**;
(3) Else return n is **prime**.

Trial division algorithm runs in time $O(\sqrt{n}log^2(n))$, but this running time is exponential in the input size since the input represents n as a binary number with $\lceil log_2(n)\rceil$ digits. Similarly, for large value of n, the Sieve of Eratosthenes requires a lot of memory and for proving primality, it can require up to $\sqrt{n}$ cycles.

2.2 Wilson's characterization of primes

2.2.1 Algorithm

Input : $n \in$ N
(1) If (n-1)! $\equiv$ -1 (mod n), output **prime**;
(2) otherwise, return **composite**.

This algorithm is based on a theorem by John Wilson in 1770.

2.2.2 Wilson's Theorem

Theorem 2.2.1. *Let $n \in N$. Then n is prime if and only if (n-1)! $\equiv$ -1 (mod n).*

Proof: $\Rightarrow$: Suppose n is prime. Then every integer in the interval [2,3,4, $\cdots$, n-2] is coprime to n and has a unique inverse modulo n. Therefore,

$$\prod_{2\leq j\leq n-2} j \equiv 1(mod(n))$$

and we know that (n-1) $\equiv$ -1 (mod n). Hence,

$$\prod_{2\leq j\leq n-1} j = (n-1)! \equiv -1(mod(n)).$$

$\Leftarrow$: Now, for the converse part, suppose that n is composite. Then {1, 2, 3, $\cdots$, n-1} contains all prime factors of n, which implies that (n-1)! $\neq$ -1 (mod n) (because if $(n-1)! \equiv -1(mod(n))$, then a factor of n, say d, will also satisfy this congruence. One can see that $(n-1)! \equiv 0(mod(d))$). $\square$.

From the Wilson's characterization of primes, we can determine the primality of an integer n by calculating (n-1)! (mod n). But this computation requires (n-1) multiplications, making it very time consuming.

2.3 Euler Test and Lucas test

Euler Test :

This test is based on the following simple lemma :

Lemma 2.3.1. *n is prime if and only if $\phi(n)$ = n-1.*

Proof : If n is prime number, then every integer less than n is relatively prime to it, hence, by definition, $\phi(n)$ = n-1. Conversely, if $n > 1$ is composite, then n has a divisor d such that 1<d<n. It follows that there are at least two integers among 1,2,3,$\cdots$,n which are not relatively prime to n, namely d and n itself. As a result, we get $\phi(n) \leq$ n-2. This proves the lemma. $\square$.

2.3.1 Algorithm (Euler test)

Input $n \in \mathbb{N}$.
Check $\phi(n)$;
If $\phi(n) = n$-1,
then output is **Prime**
Else, return **composite**.

From the Euler characterization of primes, we can determine the primality of an integer n by calculating $\phi(n)$, but for calculating $\phi(n)$, we require the factors of n and factorization is more difficult problem than primality testing.

Lucas Test :

Before going ahead, we will discuss Euler theorem, which is useful in Lucas theorem. Lucas primality test is based on the Lucas theorem.

2.3.2 Euler Theorem

Theorem 2.3.2. *If $n \geq 1$ and $gcd(a, n) = 1$, then $a^{\phi(n)} \equiv 1 \pmod{n}$.*

Before proving Euler theorem, we will do another lemma :

Lemma 2.3.3. *Let $n > 1$ and gcd $(a, n) = 1$. If $a_1, a_2, \cdots, a_{\phi(n)}$ are the positive integers less than n and relatively prime to n, then $aa_1, aa_2, \cdots, aa_{\phi(n)}$ are congruent modulo n to $a_1, a_2, \cdots, a_{\phi(n)}$ in some order.*

Proof : Observe that, none of the integers $aa_1, aa_2, \cdots, aa_{\phi(n)}$ are congruent modulo n, because if $aa_i \equiv aa_j \pmod{n}$, with $1 \leq \mathrm{i} < \mathrm{j} \leq \phi(n)$, then the cancellation laws yields $a_i \equiv a_j \pmod{n}$, and thus $a_i = a_j$, a contradiction. Furthermore, since $\gcd(a_i, n) = 1$ for all i and $\gcd(a, n) = 1$, each of the aa_i is coprime to n. Fixing on a particular aa_i, there exists a unique integer b, where $0 \leq b < n$, for which $aa_i \equiv b(modn)$. Because $gcd(b, n) = gcd(aa_i, n) = 1$. b must be one of the integers $a_1, a_2, \cdots, a_{\phi(n)}$. All told, this proves that the numbers $aa_1, aa_2, \cdots, aa_{\phi(n)}$ and the numbers $a_1, a_2, \cdots, a_{\phi(n)}$ are identical (modulo n) in a certain order. □

Proof of the Euler theorem : There is no harm in taking $n > 1$. Let $a_1, a_2, \cdots, a_{\phi(n)}$ be the positive integers less than n that are relatively prime to n. Because $\gcd(a, n) = 1$, it follows from the above lemma that $aa_1, aa_2, \cdots, aa_{\phi(n)}$ are congruent, not necessarily in order of appearence, to $a_1, a_2, \cdots, a_{\phi(n)}$. Then :
$aa_1 \equiv a_1^{'} \pmod{n}$; $aa_2 \equiv a_2^{'} \pmod{n}$; $\cdots$; $aa_{\phi(n)} \equiv a_{\phi(n)}^{'} \pmod{n}$
where $a_1^{'}, a_2^{'}, \cdots, a_{\phi(n)}^{'}$ are the integers $a_1, a_2, \cdots, a_{\phi(n)}$ in some order. On taking the product of these $\phi(n)$ congruences, we get

$$(aa_1)(aa_2)\cdots(aa_{\phi(n)}) \equiv a_1^{'}a_2^{'}\cdots a_{\phi(n)}^{'}(modn) \equiv a_1a_2\cdots a_{\phi(n)}(modn)$$

and so $\mathrm{a}^{\phi(n)}(a_1a_2 \cdots a_{\phi(n)}) \equiv a_1a_2 \cdots a_{\phi(n)} \pmod{n}$. Since $\gcd(a_i, n) = 1$ for each i, so, $\gcd(a_1a_2 \cdots a_{\phi(n)}, n) = 1$. Therefore, we may divide both sides of the foregoing congruence by the common factor $a_1a_2 \cdots a_{\phi(n)}$, leaving us with the result: $a^{\phi(n)} \equiv 1 \pmod{(n)}$. □

2.3.3 Lucas Theorem

Theorem 2.3.4. *Let $n > 1$. If for every prime factor p of $n - 1$, there exists an integer a such that :*
(1) $a^{n-1} \equiv 1 \pmod{n}$ and
(2) $a^{\frac{n-1}{p}}$ is not congruent to 1 (mod n),
then n is prime.

Proof of Lucas theorem : Suppose n satisfies the conditions of the theorem. To show that n is prime, it is enough to show that $\phi(n) = n - 1$ (using lemma 2.3.1). Since in general $\phi(n) < n - 1$, to show equality we will show that under the above conditions $n - 1$ divides $\phi(n)$ (which means $\phi(n) \geq$ n-1 $\Rightarrow \phi(n) =$ n-1). Suppose not. Then there exists a prime p such that p^r divides $n - 1$ but p^r does not divide $\phi(n)$ for some exponent r ≥ 1. For this prime p, there exists an integer a satisfying the conditions of the theorem. Let m be the order of a modulo n. Then m divides $n - 1$ (since the order of an element divides any power that equals 1). However, by the second condition in the theorem and for the same reason, m does not divide $\frac{n-1}{p}$. Therefore p^r divides m, which divides $\phi(n)$ (using theorem 2.3.2), contradicting our assumption. Hence $n - 1 = \phi(n)$ and therefore n is prime. □

2.3.4 Algorithm (Lucas test)

Input $n \in \mathbb{N}$, $n \geq 2$
Factor n-1 in to prime factors
Choose an a such that $\gcd(a, n) = 1$
If $a^{n-1} \equiv 1 \pmod{n}$ and $a^{\frac{n-1}{p}} \neq 1 \pmod{n}$ for all prime factor p
then return **Prime**
Else return **Probably composite**.

Although this Lucas test is deterministic, but it depends on the factorization of n − 1. In general, factorization is even more difficult than testing for primality.

2.4 Fermat Test and Carmichael Numbers

2.4.1 Algorithm (Fermat Test(n))

Input : $n \in$ N
(1) Choose $x \in \{1, 2, 3, \cdots, \text{n-1}\}$ uniformly at random.
(2) If $x^{n-1} \neq 1$ (mod n), return **composite**;
Else
(3) return **Probably Prime**.

The above algorithm is based on Fermat's little theorem.

2.4.2 Fermat Little theorem

Theorem 2.4.1. *(Fermat's little theorem) : The number n is prime $\Longleftrightarrow$ the congruence*

$$x^{n-1} \equiv 1(mod n)$$

is satisfied for every integer x between 0 and n.

Before proving the theorem, we will discuss a proposition which is needed in the proof.

Proposition 2.4.2. *If A is a subset of the integers which is closed under addition and subtraction, then A is equal to $d\mathbb{Z}$, the set of all multiples of d, for some integer d.*

Proof: If A = $\{0\}$ then we can take d = 0 and we are done. Otherwise, let d be the absolute value of the smallest non-zero element of A. Thus A contains $\{\pm$d$\}$ and A is closed under addition and subtraction. So, it contains all multiple of d. Now it is enough to prove that A does not contain any integer x which is not divisible by d. If A has, then we can subtract x by the nearest multiple of d and we get a nonzero element whose absolute value is less than d, a contradiction. □

Proof of Fermat's little theorem : $\Rightarrow$ Suppose n is prime, let A be the set of integers x which satisfy $x^n \equiv x$ (mod n). This set contains $x = 1$, and it is closed under addition and subtraction [because we know that, if p is prime then for every pair of integers a, b satisfies $(\text{a} + \text{b})^p \equiv \text{a}^p + \text{b}^p$ (mod p)]. Hence every integer x

belongs to A (from proposition 2.4.2).
Now let x be any integer not divisible by n. The fact that $x \in$ A means that n|(x^n-x) implies n|x(x^{n-1}-1). Since n is prime and x is indivisible by n, this implies n|(x^{n-1}-1), i.e. $x^{n-1} \equiv 1$ (mod n).
$\Leftarrow$ Now suppose n is not prime then it has a divisor d>1. Thus if we take $x =$ d then the number d^{n-1} is divisible by d, so, it is not equal to 1 mod n. □

2.4.3 Fermat witness and Fermat liar

Definition 2.4.3. *Let n be composite number. Define a number $x \in \{1,2,\cdots,n-1\}$:*

1. *Fermat witness for n if $x \nmid n$ and $x^{n-1} \neq 1$ (mod n).*
2. *Fermat liar for n if $x \nmid n$ and $x^{n-1} \equiv 1$ (mod n).*

2.4.4 Idea of the Algorithm

The idea of algorithm is simple.
(1) pick a positive integer $x <$ n.
(2) Check Whether x is Fermat witness.
(3) If so, then output "composite". Otherwise output "probably prime".
Now, to determine whether x is Fermat witness for n, we needs to compute x^{n-1} mod n. The obvious way of doing this requires n-2 iterations of mod n multiplication. But using the binary expansion of n-1 and repeated squaring method, we can reduce this to O(logn) multiplication operations.
Example : let's take $n = 23$, then n-1 = 22 = 16+4+2 ; So,

$$x^{22} = x^{16}x^4x^2 = (((x^2)^2)^2)^2.(x^2)^2.x^2$$

and this describes an efficient algorithm for raising any integer to the 22^{nd} power.

Before proving the correctness of the algorithm, we define Carmichael numbers.

2.4.5 Carmichael Numbers and Their Some Properties

Definition 2.4.4. *A composite number n is said to be a Carmichael number if every x satisfying gcd(x,n) = 1 is a Fermat liar for n.*

For proving some properties of Carmichael numbers, we require some lemma's :

Lemma 2.4.5. *Let a, b be any two integers and let $d = gcd(a, b)$. The set $a\mathbb{Z} + b\mathbb{Z} = \{ar + bs : r, s \in \mathbb{Z}\}$ is equal to $d\mathbb{Z}$ where $d = gcd(a, b)$.*

Proof: We can easily observe that the set $a\mathbb{Z} + b\mathbb{Z}$ is closed under addition and subtraction. So, using proposition 2.3.2, $a\mathbb{Z} + b\mathbb{Z} = c\mathbb{Z}$ for some integer c. Given d = gcd(a, b). So, every element of $a\mathbb{Z}$ + b$\mathbb{Z}$ is divisible by d, means d|c. But a and b are both elements of c$\mathbb{Z}$, i.e. both are divisible by c. This means c is common divisor of a and b, so c|d. It follows c = d. □

Lemma 2.4.6. *If gcd(b, n) = 1 then there is an integer b^{-1} such that $b \cdot b^{-1} \equiv 1 \pmod{n}$.*

Proof: By lemma 2.4.5, the set b$\mathbb{Z}$ + $n\mathbb{Z}$ is equal to $\mathbb{Z}$, the set of all integers. This means their are integers r, s such that br + ns = 1. This implies b·r $\equiv 1 \pmod{n}$, as desired. □

2.4.6 Chinese Remainder Theorem

Theorem 2.4.7. *(Chinese Remainder Theorem) : Let n_1,n_2,n_3, $\cdots$,n_k be positive integers such that gcd(n_i,n_j) = 1 for i≠j. Then for any integers a_1,a_2,a_3, $\cdots$,a_k the system of congruences*

$$x \equiv a_1 (mod n_1); x \equiv a_2 (mod n_2); .....; x \equiv a_k (mod n_k)$$

has a solution, which is unique modulo the integer $n_1 n_2 n_3 \cdots n_k$.

Proof: Let m_i denote the product of all elements of the set $\{n_1, n_2, n_3, \cdots, n_k\}$ other than n_i. We can see that gcd(m_i, n_i)=1. So, using lemma 2.4.6 implies that there is a number r_i such that $m_i r_i \equiv 1 \pmod{n_i}$. Now let

$$x = \sum_{i=1}^{k} a_i m_i r_i$$

We can easily see that this x satisfies the given system of congruences.
If x_1, x_2 both satisfy the given system of congruences, then x_1 - x_2 is divisible by each of the $n_1, n_2, n_3, \cdots, n_k$. And we know that gcd$(n_i, n_j) = 1$, So, x_1-x_2 is divisible by $n_1 n_2 n_3 \cdots n_k$. □

Lemma 2.4.8. *If p is prime, then for any k > 0, the number of x ∈ {1,2,3, ⋯ ,p-1} satisfying $x^k \equiv 1 \pmod p$ is at most k.*

Proof: We will prove this lemma more generally, for any non zero polynomial $P(x) = a_0 + a_1x + \cdots + a_kx^k$, the number of $x \in \{1,2,3,\cdots,p\text{-}1\}$ satisfying $P(x) \equiv 0 \pmod p$ is at most k. We will prove it by induction on k.
For k = 0, it is trivial.
Now suppose a satisfies $P(a) \equiv 0 \pmod p$. We may write $P(x) = (x - a)Q(x) + c$, where $Q(x)$ is a polynomial of degree k-1 with integer coefficients. Since $P(a) \equiv 0 \pmod p$, it means c is divisible by p. If b satisfies $P(b) \equiv 0 \pmod p$ but $Q(b) \not\equiv 0 \pmod p$ then p is a divisor of $(b - a)Q(b)$ but not of $Q(b)$, hence $b \equiv a \pmod P$. It follows that every $b \in \{1,2,3,\cdots,p\text{-}1\}$ satisfying either $b = a$ or $Q(b) \equiv 0 \pmod P$. Hence by the induction hypothesis, at most k-1 elements of $\{1,2,3, \cdots,p\text{-}1\}$ satisfy the second congruence. □

Some properties of Carmichael numbers :

Proposition 2.4.9. *Every Carmichael number is odd.*

Proof: If n (≥ 4) is even then $(n - 1)^{n-1} \equiv (-1)^{n-1} \equiv -1 \bmod(n)$, so is not congruent to $1 \bmod(n)$. □

Lemma 2.4.10. *If n is a Carmichael number, then n has at least three distinct prime factor and is not divisible by the square of any prime.*

Proof: Firstly, we will show that Carmichael numbers are square free. Suppose to contrary that p is prime and $p^2 \mid n$. let $n = p^kq$ where $k > 1$ and $p \nmid q$. Using the Chinese remainder theorem, we may find a number x such that $x \equiv p+1 \pmod{p^k}$ and $x \equiv 1 \pmod q$. Now, we can observe that $\gcd(x, n) = 1$, because x has no common factors with p^k and q, and $n = p^kq$. We can also check easily that x is a Fermat witness for n or $x^{n-1} \not\equiv 1 \pmod n$. To prove $x^{n-1} \not\equiv 1 \pmod n$, it is enough to prove that $x^{n-1} \not\equiv 1 \pmod{p^2}$ or equivalently $x^n \not\equiv x \pmod{p^2}$. We know that

$$(p+1)^p = \sum_{k=0}^{p} \binom{p}{k} p^k$$

implies $(p+1)^p \equiv 1 \pmod{p^2}$, which implies that $x^n \equiv 1 \pmod{p^2}$ and which prove that $x^n \not\equiv x \pmod{p^2}$. Hence Carmichael numbers are square free.
Now, it remains to prove that if p, q are distinct odd primes then their product $n = pq$ is not a Carmichael number. Assume without loss of generality that $p < q$. By

using lemma 2.4.8, we can ensure that we can choose a $x \in \{1,2,3,\cdots,q\text{-}1\}$ such that congruence $x^{p-1} \neq 1 \pmod{q}$. Now, it is enough to prove that x is a Fermat witness for n. To prove this, we can observe that

$$x^{n-1} = x^{pq-1} = x^{pq-p}x^{p-1} = (x^p)^{q-1}x^{p-1} = x^{p-1} \neq 1(modq)$$

Since q is a divisor of n. It follows that $x^{n-1} \neq 1 \pmod{n}$. Thus we have shown that x is Fermat witness for n. Therefore $n = pq$ is not a Carmichael number. □

Proposition 2.4.11. *If a prime p divides a Carmichael number n, then n ≡ 1 (mod p-1).*

Proof: Given that n is a Carmichael number. Let p be a prime divisor of n: say $n = p^k r$ for some k, where r is not a multiple of p. Take an element a which has order p−1 in the field $(\mathbb{Z}/p\mathbb{Z})^*$ (such an element exist because we know $(\mathbb{Z}/p\mathbb{Z})^*$ is finite field, which is cyclic). By the Chinese remainder theorem (theorem 2.4.7), there exists b such that $b \equiv a \bmod p$ and $b \equiv 1 \bmod r$. Then b is coprime to both p and r, and therefore to n. By hypothesis, $b^{n-1} \equiv 1 \bmod n$, so $b^{n-1} \equiv 1 \bmod p$. Also, order of b is dividing n−1. So p−1 divides n−1. □

Theorem 2.4.12. *A number n is a Carmichael number if and only if $n = p_1 p_2 \cdots p_k$, a product of (at least three) distinct primes, and $p_j - 1$ divides $n-1$ for each j.*

Proof:

⇒ This part follows using lemma 2.4.10 and proposition 2.4.11.

⇐ Let n be as stated, and let gcd(a, n) = 1. By Fermat's theorem (theorem 2.4.1), for each j, we have $a^{p_j-1} \equiv 1 \bmod p_j$. Since $p_j - 1$ divides n−1, $a^{n-1} \equiv 1 \bmod p_j$. This holds for each j, hence $a^{n-1} \equiv 1 \bmod n$. □

Now, before discussing correctness of the algorithm, we will discuss a lemma :

Lemma 2.4.13. *If a,b,n are positive integers such that gcd(b, n)=1 and the congruence $x^a \equiv b \pmod{n}$ has $k > 0$ solutions, then the $x^a \equiv 1 \pmod{n}$ also has k solutions.*

Proof: Let x_0 be a solution of $x^a \equiv b \pmod{n}$. We can observe that $\gcd(x_0, n) = 1$, because otherwise $\gcd({x_0}^a, n) = \gcd(b, n)$ would be greater than 1, contradicting our hypothesis gcd(b,n)=1. Using lemma 2.4.6, there exist a number ${x_0}^{-1}$ such that $x_0 \cdot {x_0}^{-1} \equiv 1 \pmod{n}$. Now we can define a one to one correspondence between the solution

sets of $x^a \equiv$ b (mod n) and $x^a \equiv 1$ (mod n), by defining the map y $\rightarrow$ y$\cdot x_0$, which proves the lemma. □

2.4.7 Correctness of the Algorithm

Theorem 2.4.14. *The above algorithm outputs probably prime when n is prime and outputs composite with probability at least 1/2 when n is odd composite number but not a Carmichael number.*

Proof: The first part of the theorem (i.e. when n is prime the output is probably prime) is obvious from Fermat little theorem 2.4.1.
For the second part, when n is odd composite number and not a Carmichael number which means that n has at least one Fermat witness x such that gcd(x, n) = 1. Now it is enough to prove that at least half of the elements of 1, 2, 3,$\cdots$, $n-1$ are Fermat witness for n. If gcd(x, n) = 1 and x is a Fermat witness for n, then $x^{n-1} \equiv$ b (mod n) for some b $\neq 1$ satisfying gcd$(b, n) = 1$. Using lemma 2.4.13, we can say that there are at least as many Fermat witnesses as Fermat liar. And we know that 2 is the only even number which is prime, all the other even numbers are composite. Thus from the above arguments, we can see that output will be composite with probability at least 1/2. □

So far we have established the Fermat t est which always outputs "probably prime" when n is "prime", and outputs composite with probability at least 1/2 when n is an odd composite number but not a Carmichael number. But we still don't have a good algorithm for distinguishing Carmichael numbers from prime numbers. The Miller-Rabin test is a more advanced version of the Fermat test which accomplishes this.

2.5 The Miller-Rabin primality Test

This is the advanced version of the Fermat primality test. This is based on Fermat little theorem and the following lemma :

Lemma 2.5.1. (fake square root lemma)
If x, n are positive integers such that $x^2 \equiv 1$ (mod n) but $x \neq \pm 1$ (mod n), then n is composite.

Proof: From the hypothesis of lemma, n divides x^2-1 = (x-1)(x+1), but n divides neither $x + 1$ nor x - 1. This is impossible when n is prime. Hence n is composite. □

2.5.1 Algorithm (Miller-Rabin(n))

Input : $n \in \mathbb{N}$
(1) If $n > 2$ and n is even, return **composite**.
(2) Choose $x \in \{1, 2, 3, \cdots, \text{n-1}\}$ uniformly random.
(3) Test if $x^{n-1} \neq 1 \pmod{\text{n}}$, return **composite**;
Else
(4) Factor n-1 = $2^s \cdot$t, where t is odd.
(5) Compute

$$u_i = x^{2^i \cdot t}(mod(n)); 0 \leq i < s.$$

(6) If there is an i such that $\text{u}_i = 1$ and $\text{u}_{i-1} \neq \pm 1$, then output **composite**;
(7) Else return **probably prime**.

2.5.2 Idea of the algorithm

The idea of the test is to pick a random number x in $\{1, 2, \cdots, n\text{-}1\}$ and use it to try finding either a Fermat witness or a fake square root of 1 mod$n$.

2.5.3 Correctness of the algorithm

Definition 2.5.2. *Let n be an odd composite number . Define a number $x \in \{1,2,3,\cdots,n\text{-}1\}$:*
(1) Miller-Rabin witness (or MR-witness) if the algorithm (2.5.1) outputs **composite**.
(2) Miller-Rabin liar (or MR-liar) if the algorithm (2.5.1) outputs **probably prime**.

Theorem 2.5.3. *If n is prime then Miller-Rabin primality testing algorithm outputs "***probably prime***" with probability 1 and if n is an odd composite number, then algorithm outputs "***composite***" with probability at least 3/4.*

Proof: The first part of the theorem is obvious. Because, when n is prime then we can easily see that **step (1)** will not output **composite**. **step (3)** will also not output composite using Fermat little theorem (theorem 2.4.1). And using lemma

2.5.1, **step (6)** will not output composite. Hence output will be "**probably prime** in **step (7)**.

Now, for proving the second part of the theorem, suppose n is an odd composite number. We will solve this part in two subparts :

(1) When n has at least one square factor.

(2) When n is square free.

(1) : When n has at least one square factor or $n = P_1^{\alpha_1} P_2^{\alpha_2} P_3^{\alpha_3} \cdots P_k^{\alpha_k}$, where at least one $P_j^{\alpha_j}$ has $\alpha_j \geq 2$. As we know that if p is an odd prime, then the number of incongruence solutions of $x^q \equiv 1$ (mod p^α) is at most gcd(q, $p^{\alpha-1}$(p-1)). Let n = 1+2^st, and x be MR-liar for n. We than have :

$$x^t \equiv 1(mod(n) \quad or \quad (x^t)^{2^r} \equiv -1(mod(n))$$

for some r < s. In particular, from the inequality we have $x^{n-1} \equiv 1$ (mod$n$). So, we can now conclude that there are at most gcd(n-1, $P_i^{\alpha_i}$(P_i-1)) = gcd(n-1, P_i-1) incongruent solutions to $x^{n-1} \equiv 1$ (mod $P_i^{\alpha_i}$) for i = 1,2, $\cdots$,k. Hence, by Chinese Remainder theorem (theorem 2.4.7), there are at most $\prod_{i=1}^{k}$gcd(n-1, p_i-1) incongruent solutions to $x^{n-1} \equiv 1$ (mod$n$). Since at least one $P_j^{\alpha_j}$ has $\alpha_j \geq 2$. So, we have :

$$\frac{p_j - 1}{P_j^{\alpha_j}} = \frac{1}{P_j^{\alpha_j - 1}} - \frac{1}{P_j^{\alpha_j}} \leq \frac{2}{9}$$

because p_j must be greater than or equal to 3. Then, for $n \geq 9$, we have $\prod_{i=1}^{k}$gcd(n-1, p_i-1) $\leq \prod_{i=1}^{k}$(p_i-1) $\leq \prod_{i \neq j}^{k}$($p_i$-1)($\frac{2}{9}P_j^{\alpha_j}$) $\leq \prod_{i \neq j}^{k}$($p_i$)($\frac{2}{9}P_j^{\alpha_j}$) $< \frac{n}{4}$.

(2) : Now, suppose n is squre free or $n = P_1 \cdot P_2 \cdots P_k$. Write $P_i = 1 + t_i \cdot 2^{s_i}$ with t_i odd. Hence we have $n = 1 + 2^s \cdot t = P_1 \cdot P_2 \cdots P_k = (1 + t_1 \cdot 2^{s_1})(1 + t_2 \cdot 2^{s_2}) \cdots (1 + t_k \cdot 2^{s_k})$. We can observe from here that s is not less than minimum of the s_i. For proving this subpart, it is enough to show that the set T = $\{x \in \mathbb{Z}/n\mathbb{Z} \mid x^t \equiv 1(mod(n)$ or $(x^t)^{2^r} \equiv -1(mod(n))$ for some r < s $\}$ has cardinality less than$n/4$. We decompose T into the set $T_{-1} = \{x \mid x^t \equiv 1(mod(n)\}$ and the sets $T_r = \{x \mid (x^t)^{2^r} \equiv -1(mod(n))$ for some r < s $\}$. Then, elements of T_{-1} reduce to units in $\mathbb{Z}/p_i\mathbb{Z}$ which have order dividing t. This is a subgroup of order $\gcd(t, p_i - 1) = \gcd(t, t_i)$. Thus, by the Chinese remainder theorem (theorem 2.4.7) :

$$\#T_{-1} = \prod_{i=1}^{k} \gcd(t, t_i)$$

The elements of T_r can be characterized as elements whose t^{th} power has order exactly 2^{r+1} when reduced modulo p_i. In particular, this means that $r < s_i$ for every i ; the other T_r's are empty. There are exactly $\gcd(t \cdot 2^{r+1}, t_i \cdot 2^{r+1}) = \gcd(t, t_i) \cdot 2^{r+1}$ elements in $\mathbb{Z}/p_i\mathbb{Z}$ with order dividing $t \cdot 2^{r+1}$ and among these a subgroup of index 2 has elements of order dividing $t \cdot 2^r$ (Since the group of units in $\mathbb{Z}/p_i\mathbb{Z}$ (finite field) is cyclic group and a subgroup of a cyclic group is cyclic). Thus, by Chinese remainder theorem (theorem 2.4.7), we obtain

$$\#T_r = \prod_{i=1}^{k} \gcd(q, q_i) \cdot 2^{kr}$$

Thus we see that the cardinality of T is :

$$\#T = \#T_{-1} + \sum_{r=0}^{l-1} \#T_r = \prod_{i=1}^{k} \gcd(t, t_i) + \sum_{r=0}^{l-1} \prod_{i=1}^{k} \gcd(t, t_i) \cdot 2^{kr} = \prod_{i=1}^{k} \gcd(t, t_i) \cdot (1 + \sum_{r=0}^{l-1} 2^{kr})$$

which is equal to $\prod_{i=1}^{k} \gcd(t, t_i)(\frac{2^{kl}+2^k-2}{2^k-1})$, where l is the minimum of the s_i's.
Now, using Chinese remainder theorem (theorem 2.4.7), the number of units in $\mathbb{Z}/n\mathbb{Z}$ is precisely $(p_1\text{-}1)\cdot(p_2\text{-}1) \cdots (p_k\text{-}1)$, which is equal to $\prod_{i=1}^{k} t_i \cdot 2^{\sum_{i=1}^{k} s_i}$, which is at least one less than$n$. Thus the proportion of elements in T is strictly smaller than $\frac{\prod_{i=1}^{k} \gcd(t,t_i)(\frac{2^{kl}+2^k-2}{2^k-1})}{\prod_{i=1}^{k} t_i \cdot 2^{\sum_{i=1}^{k} s_i}} = \frac{\prod_{i=1}^{k} \gcd(t,t_i)}{\prod_{i=1}^{k} t_i} \cdot \frac{2^{kl}+2^k-2}{(2^k-1)\cdot 2^{\sum_{i=1}^{k} s_i}}$.
The first term in the above expression is not more than 1, while the second term is not more than $1/2^{k-1}$ (note that $l \geq 1$ and equal to minimum of s_i). Thus we obtain the result unless $k = 2$. Now let $k = 2$ and suppose $s_1 > s_2$ (or vice versa) then we see that the second term is no more than $1/2^k$, so, we have the result in this case. Thus we may assume that $s_1 = s_2 = l$. Now if $\gcd(t, t_1) < t_1$ then (since these are odd numbers and one divides the other) $\gcd(t, t_1) \leq 3t_1$. This implies that the above expression is no more than 1/6. Thus, we may further assume that $\gcd(t, t_1) = t_1$. By expanding the identity $(1 + t \cdot 2^s) = (1 + t_1 \cdot 2^l)(1 + t_2 \cdot 2^l)$, we see that $\gcd(t, t_1) = \gcd(t, t_2)$. Since the primes p_1 and p_2 are distinct, so, $t_1 \neq t_2$; thus $t_1 = \gcd(t, t_1) \leq 3t_1$ as above. Now we again obtain that the above expression is no more than 1/6. This completes the proof of the theorem. □

2.5.4 Run time analysis

The running time of Miller-Rabin primality tesing algorithm is $O(\log^3 n)$. To see this, we have already mentioned in the subsection 2.4.4 that it is possible to compute x^t

(mod$n$) using $O(\log n)$ mod-n multiplication operations (Each mod-n multiplication takes time $O(\log^2 n)$ using the naive algorithms for integer multiplication and division). Once we have computed x^t (mod$n$), the remaining numbers x^{2t}, x^{4t}, $\cdots$,$x^{2^s t}$ (mod$n$) may be obtained by s $\leq \log_2(n)$ iterations of repeated squaring mod$n$, which again provide $O(\log n)$ mod-n multiplication operations. All the remaining operations in the Miller-Rabin algorithm require much less running time.

2.6 Primality Certificate

We now examine the situation where n is probably prime (having passed the Miller-Rabin test with probability about 1 - $(1/4)^n$). In such a situation, we wish to provide a "certificate" that n is a prime. One situation that one can think of is where an "oracle" produces keys for us. While we trust the oracle not to "leak" a key, we are not so sure that oracle (in order to save time and money) may be using some quick and dirty method to generate the modulus, which may be weak. Then we would ask the oracle to "provide proof" that it has given us a prime number. Another situation is that someone "pay" us to factor a number. We would need to certify that factorization is complete. The certificate should be very "easy" to check.

Certificate :

Theorem 2.6.1. *$n \in \mathbb{N}$ is prime iff there exist an element $a \in \mathbb{Z}/n\mathbb{Z}$, such that $a^m = 1$ but $a^{m/q} \neq 1$ and $q \geq \sqrt{n}$ for some integer m and a prime factor q.*

Proof. Suppose we find an element a in $\mathbb{Z}/n\mathbb{Z}$ so that $a^m = 1$ but $a^{\frac{m}{q}} \neq 1$ and $q \geq \sqrt{n}$; for some integer m and a prime factor q. Then n is prime because if p is a factor of n, then gcd($a^{\frac{m}{q}}$-1, p) divides gcd($a^{\frac{m}{q}}$-1,n) and thus $a^{\frac{m}{q}} \neq 1$ in $\mathbb{Z}/p\mathbb{Z}$. But this means q divides p-1. Since q $\geq \sqrt{n}$, so it can not be possible. Hence n is prime. On the other hand , if n is not prime, n must have a prime factor smaller than $\sqrt{n}$.

□

Remark 2.6.2. *The correctness of the above certificate depends on the primality of various q's, so, we would need a certificate for those as well.*

Another Certificate : As we have discussed Lucas primality test (theorem 2.3.4), which is also a primality certificate.

2.7 AKS Algorithm

2.7.1 Algorithm

Input : $n \in \mathbb{N}$
(1) If $\exists\ a, b > 1 \in \mathbb{N}$ such that $n = a^b$, then output **composite**; (Call Algorithm 2.7.4.1)
(2) Find the minimal r $\in \mathbb{N}$ such that $o_r(n) > \log^2(n)$; (by lemma 2.7.4)
(3) For $a = 1$ to r do
if $1 < (a, n) < n$, then output **composite**;
(4) if r $\geq n$, then output **prime**;
(5) For $a = 1$ to $\lfloor \sqrt{\phi(r)} \cdot log(n) \rfloor$ do
if $(X + a)^n \neq X^n + a$ (mod $X^r - 1$,n), output **composite**;
(6) Return **Prime**.

2.7.2 Idea of the algorithm

This algorithm based on the following lemma :

Lemma 2.7.1. *Let $n \in \mathbb{N}$, $n \geq 2$, $a \in \mathbb{Z}$, such that gcd(a,n) = 1. Then*

$$n \ \ is \ \ prime \iff (X + a)^n \equiv X^n + a (mod\ n).$$

Proof: For $0 < i < n$, the coefficient of X^i in $((X + a)^n - (X^n + a))$ is $\binom{n}{i} a^{n-i}$.
$\Rightarrow$: Suppose n is prime then $\binom{n}{i} \equiv 0$ (mod$n$). Hence all the coefficients are zero and we are done.
$\Leftarrow$: Suppose n is composite. Consider a prime q that is a factor of n and q^k |n, where $k \geq 1$. Let $n = q^k \cdot t$. Then, q^k does not divide $\binom{n}{q}$ and coprime to a^q (because $\binom{n}{q} = \frac{n(n-1)(n-2)\cdots(n-q+1)}{q!}$, where numerator is divisible by q^k but not divisible by q^{k+1} and denominator is divisible by q and (a, q)=(a^q, q^k) = 1). Therefore, coefficient of X^{n-q} is not zero (mod n). Thus $(X + a)^n - (X^n + a)$ is not identically zero over $\mathbb{Z}/n\mathbb{Z}$. Thus, our lemma follows. □

The above identity suggests a simple test for primality of an integer n in the following steps :
Given an input $n \in \mathbb{N}$

(1) Choose an integer a such that gcd(a, n) = 1.
(2) Calculate f(X) = $(X + a)^n$ – $(X^n + a)$ (mod$n$)
(3) If f(X) $\equiv$ 0, then output **prime**;
(4) Else return **composite**.
However, this takes time $O(n)$ because we need to evaluate n coefficients in the Step (2) in the worst case. A simple way to reduce the number of coefficients is to evaluate both sides of the identity modulo a polynomial of the form $X^r - 1$ for an appropriately chosen small r. In other words, test if the following equation is satisfied:

$$(X + a)^n = X^n + a \ \ (mod\, X^r - 1,\ n) \tag{2.1}$$

From lemma 2.7.1, it is immediate that all primes n satisfy the equation (2.1) for all values of a and r. The problem now is that some composites n may also satisfy the equation for a few values of a and r (and indeed they do). However, we can almost restore the characterization: we show that for appropriately chosen r if the equation (2.1) is satisfied for several a's then n must be a prime power. The number of a's and the appropriate r are both bounded by a polynomial in log(n) and therefore, we get a deterministic polynomial time algorithm for testing primality.

2.7.3 Correctness of the algorithm

Before proving the correctness of the algorithm, we will prove some lemma's :

Lemma 2.7.2. *Let $n \in \mathbb{N}$ with $n \geq 7$. Then $d_n = lcm([n]) \geq 2^n$.*

Proof: We can easily check that lemma holds for $n = 7$ and $n = 8$. So, we may assume $n \geq 9$. Consider the integral

$$I(m, n) = \int_0^1 x^{m-1}(1 - x)^{n-m} dx \qquad (1 \leq m \leq n)$$

Now using binomial expansion of (1 - x)$^{n-m}$:

$$\int_0^1 x^{m-1}(1-x)^{n-m}dx = \int_0^1 x^{m-1} \sum_{k=0}^{n-m} (-1)^k x^k \binom{n-m}{k} dx = \sum_{k=0}^{n-m} (-1)^k \binom{n-m}{k} \int_0^1 x^{m+k-1} dx$$

Therefore I(m,n) = $\sum_{k=0}^{n-m}$ (-1)$^k \binom{n-m}{k} \frac{1}{m+k}$. And we know that 0 $\leq$ k $\leq$ n-m, so, m+k divides d$_n$. Thus I(m,n)$\cdot$d$_n$ $\in \mathbb{Z}$.

And using iterated integration by parts we get :

$$I(m,n) = \frac{(n-m)!}{n\cdot(n-1)\cdots m} = \frac{(n-m)!(m-1)!}{n!} = \frac{1}{m\binom{n}{m}}$$

.

Hence we get that $m\binom{n}{m}$ $|d_n$ for all $1 \leq m \leq n$. From here, we may conclude that n $\binom{2n}{n}$ $|d_{2n}$ $|d_{2n+1}$ (since $d_n|d_{n+1}$) and $(n+1)\binom{2n+1}{n+1} = (2n+1)\binom{2n}{n}$ $|d_{2n+1}$. As (n, 2n+1)=1, it follows that $n(2n+1)\binom{2n}{n}$ $|d_{2n+1}$. We know that $\binom{2n}{n}$ is the largest of the binomial coefficients in the expansion of $(1+1)^{2n}$. So, $d_{2n+1} \geq n4^n$ (for $n \geq 1$). If $n \geq 2$ then $d_{2n+1} \geq 2\cdot 4^n = 2^{2n+1}$ and if $n \geq 4$ then we have $d_{2n+2} \geq d_{2n+1} \geq 4^{n+1}$. Thus we have shown that $d_n \geq 2^n$ when $n \geq 9$ (since easy to check d_7 and d_8 directly). □

Lemma 2.7.3. *For all $n \in \mathbb{N}$, we have $\binom{2n+1}{n} > 2^{n+1}$.*

Proof: By definition, $\binom{2n+1}{n} = \frac{(2n+1)\cdots(n+2)}{n!} \geq \prod_{i=1}^{n} \frac{2i+1}{i} \geq \prod_{i=1}^{n} 2 = 2^n$. We can check that $\prod_{i=1}^{3} \frac{2i+1}{i} = \frac{35}{2} > 2^4$ from which we conclude that $\binom{2n+1}{n} > 2^{n+1}$. □

Lemma 2.7.4. *There exists an integer $r \in \mathbb{N}$ with the following properties :*
(1) $r \leq max\{3, \lceil log^5(n)\rceil\}$
(2) $o_r(n) > log^2(n)$
(3) $(r, n) = 1$

Proof: If $n = 2$, then r = 3 satisfies the lemma. So, we may assume that $n > 2$. And for $n > 2$, $\lceil \log^5(n)\rceil > 10$. Now, using lemma 2.7.2, we can see that $\mathrm{lcm}([\lceil \log^5(n)\rceil]) > 2^{\lceil log^5(n)\rceil}$. Define

$$N = n \cdot \prod_{i=1}^{\lfloor log^2(n)\rfloor} (n^i - 1)$$

Let r be the smallest integer not dividing N. It means r is not a divisor of (n^i-1) for i $\leq \lfloor \log^2(n)\rfloor$, i.e., $n^i \neq 1 (\mathrm{mod}\ r)$. Hence condition (2) of the lemma is satisfied.
To see that condition (1) is satisfied as well, Observe that $N \leq n^{1+2+\cdots+log^2(n)} = n^{\frac{1}{2}\cdot(log^4(n)+log^2(n))}$ $< n^{log^4(n)} = 2^{log^5(n)} \leq 2^{\lceil log^5(n)\rceil} \leq \mathrm{lcm}([\lceil \log^5(n)\rceil])$ (using lemma 2.7.2). Therefore $\exists r_0 \leq \lceil \log^5(n)\rceil$ such that $r_0 \nmid N$ and as r was chosen to be minimal, so, we have $r < \lceil \log^5(n)\rceil$.
Now, we have to prove condition(3) of the lemma. It is clear that $(r, n) < r$, otherwise r would be divide n and hence N. Thus $\frac{r}{(r,n)}$ is another integer less than $max\{3,$

$\lceil \log^5(n) \rceil\}$ not dividing N. But r was chosen to be minimal, hence r = $\frac{r}{(r,n)}$, i.e., (r, n) = 1. □

Now onwards in the AKS algorithm, we will make the following assumptions :
(1) Natural number n has p as a prime divisor.
(2) r is less than $\lceil \log^5(n) \rceil$ and $o_r(n) > \log^2(n)$.
(3) (r, n) = 1.
(4) Define $l = \lfloor \sqrt{\phi(r)} \cdot \log(n) \rfloor$.

Definition 2.7.5. *Let $f \in \mathbb{Z}[x]$ and $m \in \mathbb{N}$. Then m is said to be introspective for f if :*

$$f(x)^m \equiv f(x^m) (mod\, x^r - 1,\ p)$$

In lemma 2.7.1, we have seen that given a prime p, and an integer $a \in \mathbb{Z}$ such that (a, p) = 1, p is introspective for the function $(x + a)$.

Lemma 2.7.6. *$(f+g)^p = f^p + g^p$ for all f, $g \in \mathbb{F}_p[x]$.*

Proof: By using the binomial theorem, $(f+g)^p = f^p + \sum_{k=1}^{p-1} \binom{p}{k} f^k \cdot g^{p-k} + g^p$. Every term in the summation is divisible by p, and therefore equal to zero with in the ring $\mathbb{F}_p[x]$. Therefore $(f+g)^p = f^p + g^p$. □

Lemma 2.7.7. *Let $f \in \mathbb{F}_p[x]$ for some prime p. Then $f(x)^p = f(x^p)$.*

Proof: Let $f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$. Then by lemma 2.7.6, we have : $f(x)^p = (a_n)^p (x^n)^p + (a_{n-1})^p (x^{n-1})^p + \cdots + (a_1)^p (x)^p + (a_0)^p$.
Now, using Fermat's little theorem (theorem 2.4.1), it follow that $f(x)^p = (a_n)(x^n)^p + (a_{n-1})(x^{n-1})^p + \cdots + (a_1)(x)^p + (a_0) = f(x^p)$. □

Some theory about Cyclotomic polynomials and Finite fields

Definition 2.7.8. *The n^{th} cyclotomic polynomial $\Phi_n(x) \in \mathbb{C}[x]$ is the polynomial whose roots are precisely the $\varphi(n)$ primitive n^{th} roots of unity in $\mathbb{C}$.*

Let μ_n denote the group of n^{th} roots of unity over $\mathbb{Q}$. We know that as a group, $\mu_n \cong \mathbb{Z}/n\mathbb{Z}$ because of the map $a \longmapsto \mu_n^a$ where μ_n is a fixed primitive n^{th} root of unity.

$$\Phi_n(x) = \prod_{\zeta primitive \in \mu_n} (x - \mu) = \prod_{(a,n)=1} (x - \mu_n^a)$$

In particular note that the degree of $\Phi_n(x) = \varphi(n)$.
The roots of (x^n - 1) are precisely the n^{th} roots of unity. So, x^n - 1 = $\prod_{\zeta \in \mu_n} (x - \mu)$. Grouping the factors ($x$ - μ) according to their order in μ_n yields :
x^n - 1 = $\prod_{d|n} \prod_{\zeta primitive \in \mu_n} (x - \mu) = \prod_{d|n} \Phi_d(x)$.

It turns out that for all $n \in \mathbb{N}$, $\Phi_n(x)$ has integer coefficients. To prove this we need an important result regarding polynomial factorizations in $\mathbb{Z}[x]$.

Definition 2.7.9. *A polynomial f(x) $\in \mathbb{Z}[x]$ is called primitive if : f(x) = $a_n x^n$ + $a_{n-1}x^{n-1}$ + $\cdots$ + $a_1 x$ + a_0, then gcd(a_n,a_{n-1},$\cdots$,a_0) = 1.*

Lemma 2.7.10. *(Gauss Lemma) Let p(x) $\in \mathbb{Z}[x]$ be primitive.*
(1) If q|f(x)·g(x), where q is a prime integer and f(x), g(x) in $\mathbb{Z}[x]$, then either q|f(x) or q|g(x) in $\mathbb{Z}[x]$.
(2) If p(x) is reducible in $\mathbb{Q}[x]$ then it is reducible in $\mathbb{Z}[x]$.

Proof: Consider the ring $\mathbb{F}_q[x]$. Then q divides f·g in $\mathbb{F}_q[x]$ means $\overline{f \cdot g} = 0$. But $\mathbb{F}_q[x]$ is an integral domain, so, it must be the case either $\overline{f} = 0$ or $\overline{g} = 0$.
Now to prove (2) part of the lemma, suppose that f|p in $\mathbb{Q}(x)$, where f has integer coefficients. Then $\exists$g $\in \mathbb{Q}(x)$ such that p = f·g. Now, clearing the denominator right hand side gives us d·p = f·g where d$\in\mathbb{Z}$, d·p $\in \mathbb{Z}(x)$. Let p_0 be some prime factor of d. Then by part (1), p_0 divides f or g. But f was assumed to be primitive, so, p_0 divides g. Then by induction, we see that d|g. Therefore p(x) = f(x)·$\frac{h(x)}{d}$. □

Proposition 2.7.11. *$\Phi_n(x) \in \mathbb{Z}[x]$.*

Proof: We already know that $\Phi_n(x)$ is monic of degree $\varphi(n)$. For proving the lemma, we will proceed by induction. In the case that $n = 1$, the result is clear. So, let $n > 1$ and assume that $\Phi_d(x) \in \mathbb{Z}[x]$ for all $1 \leq d < n$. Then x^n - 1 = $\Phi_n(x)$f(x), f(x) = $\prod_{d|n, d<n} \Phi_d(x)$. f($x$) clearly divides x^n - 1 in $\mathbb{Q}(\zeta_n)[x]$, and because both polynomials have coefficients in $\mathbb{Q}$, we have that f(x) divides x^n - 1 in $\mathbb{Q}[x]$. Then by using Gauss lemma, f(x) divides x^n - 1 in $\mathbb{Z}[x]$, hence $\Phi_n(x) \in \mathbb{Z}[x]$. □

Lemma 2.7.12. *Let $n \in \mathbb{N}$ and q be prime with (q, n) = 1. Then $\Phi_n(x)$ factors into the product of $\frac{\varphi(n)}{o_n(q)}$ distinct monic irreducible polynomials in $\mathbb{F}_q[x]$.*

Proof: We begin this proof by showing that o_n(q) is the smallest integer such that $\zeta_n \in \mathbb{F}_{q^{o_n(q)}}$, where ζ_n is a primitive n^{th} root of unity. Indeed $\zeta_n \in \mathbb{F}_{q^k}$ if and only if $\zeta_n^{q^k} = \zeta_n$ which is equivalent to the identity $q^k \equiv 1 (\mathrm{mod}\ n)$. The smallest integer k for which this holds is obviously k = o_n(q). Thus ζ_n lies in $\mathbb{F}_{q^{o_n(q)}}$ but no proper subfield.

Therefore the degree of the minimal polynomial of ζ_n over $\mathbb{F}_q$ is $o_n(q)$. As the degree of $\Phi_n(x)$ is $\varphi(n)$, it follows that Φ_n factors into $\frac{\varphi(n)}{o_n(q)}$ monic irreducible polynomials in $\mathbb{F}_q[x]$.

Now, we have to show that these polynomials are distinct. Let $\overline{\Phi_n(x)}$ be the image of $\Phi_n(x)$ in $\mathbb{F}_q[x]$, and let $\overline{\Phi'_n(x)}$ be its formal derivative. Suppose that $\overline{g(x)}$ is irreducible in $\mathbb{F}_q[x]$ and $\overline{g(x)}^2 \mid \overline{\Phi_n(x)}$. Then we see that $(\overline{\Phi'_n(x)}, \overline{\Phi_n(x)}) \neq 1$. But $\overline{\Phi_n(x)} \mid (x^n - 1)$, So, $((x^n - 1), nx^{n-1}) \neq 1$. We know that $n \neq 0$ in $\mathbb{F}_q$, so, $((x^n - 1), nx^{n-1}) = ((x^n - 1), x^{n-1}) = 1$, which is a contradiction. Therefore, $\overline{g(x)}^2 \nmid \overline{\Phi_n(x)}$, which means that all of the irreducible factors of $\overline{\Phi_n(x)}$ are distinct. □

Lemma 2.7.13. *Let $n \in \mathbb{N}$ have prime divisor p and let $a \in \mathbb{N}$ with $0 \leq a \leq l$. If n, p are introspective for $(x + a)$, then $\frac{n}{p}$ is introspective for $(x + a)$ as well.*

Proof: By lemma 2.7.7, $f(x)^p = f(x^p)$ for all $f \in \mathbb{F}_p[x]$. Also given, as p, n are introspective for $(x + a)$. So, we have : $(x^{\frac{n}{p}} + a)^p \equiv (x^n + a) \equiv (x + a)^n \equiv (x + a)^{p \cdot \frac{n}{p}} \pmod{x^r\text{-}1, p}$. Now define $f = (x^{\frac{n}{p}} + a)$ and $g = (x + a)^{\frac{n}{p}}$. We have just shown that $f^p = g^p$ in the ring $(\mathbb{Z}/p\mathbb{Z})[x]/(x^r\text{-}1)$. Then $f^p + (-g)^p = 0$. Using lemma 2.7.6, we see that $(f\text{-}g)^p = 0$. Let $h = (f\text{-}g)$. Now, it remains to show that $h = 0$ with in the ring $(\mathbb{Z}/p\mathbb{Z})[x]/(x^r\text{-}1)$. Since $(r, p) = 1$. Using lemma 2.7.12, we see that $(x^r\text{-}1)$ factors into distinct irreducibles $h_i(x)$ over $(\mathbb{Z}/p\mathbb{Z})[x]$, i.e., $x^r\text{-}1 = \prod_i h_i(x)$. Therefore, using Chinese remainder theorem, we see that :

$$h^p \in \frac{(\mathbb{Z}/p\mathbb{Z})[x]}{x^r - 1} = \frac{(\mathbb{Z}/p\mathbb{Z})[x]}{(\prod_i h_i(x))} = \prod_i \frac{(\mathbb{Z}/p\mathbb{Z})[x]}{h_i(x)}$$

Because $(x^r\text{-}1) \mid h^p$, we see that each of the irreducible factor $h_i(x)$ of $(x^r - 1)$ divides h. Thus $(x^r - 1) \mid h$, completing the proof. □

We now can see that introspective numbers are closed under multiplication and the set of functions for which a given integer is introspective is closed under multiplication.

Lemma 2.7.14. *Let $f, g \in \mathbb{Z}[x]$, $s, t \in \mathbb{N}$. then the following holds :*
(1) If s and t are introspective for f, then $s \cdot t$ is introspective for f.
(2) If s is introspective for f and g, then s is introspective for $f \cdot g$.

Proof: (1) Given that : t is introspective for f, so, we have : $f(x)^{s \cdot t} \equiv (f(x^t))^s \pmod{x^r\text{-}1, p}$. Also s is introspective for f, so : $f(x)^s \equiv f(x^s) \pmod{x^r\text{-}1, p}$. Now, by substituting y^t for x in this above identity, we see that : $(f(y^t))^s \equiv f(y^{s \cdot t}) \pmod{y^{r \cdot t} - 1, p}$. And we know that $(y^r - 1)$ divides $(y^{r \cdot t} - 1) = (y^r - 1)(y^{r \cdot (t-1)} + y^{r \cdot (t-2)} + \cdots$

+ y^r + 1). So the above equation can be written as : $(f(y^t))^s \equiv f(y^{s \cdot t})$ (mod y^r - 1, p). Now, replacing the value of y by x and comparing with the first equation in the proof, we get the desired result : $f(x)^{s \cdot t} \equiv f(x^{s \cdot t})$ (mod x^r - 1, p).
(2) Since s is introspective for f and g, we have :

$$[f(x) \cdot g(x)]^s = f(x)^s \cdot g(x)^s \equiv f(x^s) \cdot g(x^s) = (f \cdot g)(x^s)(mod\ x^r - 1, p).$$

□

Lemma 2.7.13 and lemma 2.7.14 shows that every element of the set $I = \{(\frac{n}{p})^i \cdot p^j : i, j \geq 0\}$ is introspective for every poynomial in the set $P = \{\prod_{a=0}^{l}(x + a)^{e_a} : e_a \geq 0\}$ (Assumed that n is introspective for (x+a)). We now define two groups based on these sets .

Proposition 2.7.15. *$I_r = i$ (mod r) : $i \in I$ is a multiplicative subgroup of $(\mathbb{Z}/r\mathbb{Z})^*$.*

Sketch of the proof: Firstly, we can notice that I_r is closed under multiplication :

$$((\frac{n}{p})^{i_1} \cdot p^{j_1})(modr) \times ((\frac{n}{p})^{i_2} \cdot p^{j_2})(modr) = ((\frac{n}{p})^{i_1+i_2} \cdot p^{j_1+j_2})(modr)$$

We have already seen that (n, r) = (p, r) = 1, so both n, p are units in ($\mathbb{Z}$/r$\mathbb{Z}$). Because I_r is generated by n (mod r) and p (mod r), so we have $I_r \subset (\mathbb{Z}/r\mathbb{Z})^*$. □

Let $|I_r| = t$, and we have seen that I_r is generated by n (mod r) and p (mod r) which means $n^t \equiv 1$ (mod r). So, $o_r(n) > \log^2(n)$ implies that $t > \log^2(n)$.

Let $\Phi_r(x)$ be the r^{th} cyclotomic polynomial over $\mathbb{F}_p$. Then $\Phi_r(x)|(x^r\text{-}1)$ and by lemma 2.7.14, $\Phi_r(x)$ factors into distinct irreducibles of degree $o_r(p)$. Let F be the splitting field of x^r-1 over $\mathbb{F}_p$ and $\zeta \in F$ be primitive r^{th} root of unity with minimum polynomial $h(x) \in \mathbb{F}_p[x]$. Then $h(x)$ is an irreducible factor of x^r-1 over $\mathbb{F}_p[x]$ of order $o_r(p)$. This implies that $F = \mathbb{F}_p(\zeta) \cong \mathbb{F}_p[x]/(h(x))$ (Note that the map $\Psi : \mathbb{F}_p[x]/(h(x)) \longmapsto \mathbb{F}_p(\zeta)$ which sends $g(x)$ to $g(\zeta)$ is an isomorphism).

Recall that $P = \{\prod_{a=0}^{l}(x + a)^{e_a} : e_a \geq 0\}$. Now, define G = {f (mod h(x), p) : f $\in$ P}. This group is clearly generated by the linear polynomials x, x+1, $\cdots$ + x+l with in the field F.

Let k = o_r(p). We have already seen that k = deg(h(x)), implying that |F| = p^k. Now, we can actually use this fact to prove the lemma 2.7.13 in the case of f(x) ∈ G.

Lemma 2.7.16. *Let f(x) ∈ G.*
If $(f(x))^n = f(x^n)$ *and* $(f(x))^p = f(x^p)$ *then* $(f(x))^{\frac{n}{p}} = f(x^{\frac{n}{p}})$.

Proof: |F| = p^k $\Longrightarrow$ for all $\beta \in$ F, $\beta^u = \beta^v$ whenever u ≡ v (mod p^k-1). Suppose that v is introspective for f. Then $f(\zeta)^u = f(\zeta)^v = f(\zeta^v) = f(\zeta^u)$. Since introspective elements are closed under multiplication, so $f(x)^{n \cdot p^{k-1}} = f(x^{n \cdot p^{k-1}})$. Then the result holds as we know : $n \cdot p^{k-1} \equiv \frac{n}{p}$ (mod p^k-1). □

Before preceeding ahead, we will discuss two lemma's which will be useful in finding out the lower bound for |G|.

Definition 2.7.17. *A k-subset of [n] is a set of distinct elements of [n] having cardinality k. It is well known that the number of k-subsets of [n] is equal to* $\binom{n}{k}$.

Definition 2.7.18. *A k-multiset of [n] is a set of k elements of [n], in which repetition is allowed.*

Let f (k, n) denote the number of k-multisets of [n].

Lemma 2.7.19. *The number of k-multisets of [n] is equal to* $\binom{n+k-1}{k}$.

Proof: Let $1 \leq a_1 \leq a_2 \leq \cdots \leq a_k \leq n$ + k - 1 be a k-subset of [n + k - 1]. Now define $b_i = a_i$ - i + 1. Then {b_1 , $\cdots$, b_k} is a k-multiset on [n]. Conversely, given a k-multiset on [n] : $1 \leq b_1 \leq b_2 \leq \cdots \leq b_k \leq$ n, we can define $a_i = b_i$ + i -1 in order to get a k-subset of [n + k - 1]. Therefore, the number of k-combinations of [n] with repetition is simply the number of k-subsets of [n + k - 1], i.e., $\binom{n+k-1}{k}$. □

Lemma 2.7.20. *Let l, t ∈ ℤ. Then f(t-1,l+2) =* $\sum_{k=0}^{t-1} f(k,l+1)$.

Proof: Let A be the set of all (t − 1) - multisets of [l + 2], and B be the set consisting of all i-multisets of [l + 1] for i = 0, 1, $\cdots$, (t − 1). Then |A| = f(t − 1, l + 2) and |B| = $\sum_{k=0}^{t-1}$f(k, l + 1). For proving the lemma, we now exhibit a bijection between A and B.
Let $A_0 \in$ A. If (l + 2) $\notin A_0$ then A_0 is a (t − 1) - multiset of [l + 1] and thus an element of B. If (l + 2) $\in A_0$, then let a_{l+2} denote its multiplicity in A_0 . Then A_0-$\underbrace{\{(l+2),(l+2),\cdots,(l+2)\}}_{a_{l+2} times}$ is an element of B.

Now, let B_0 be an element of B of cardinality k (i.e., B_0 is a k-multiset of [l+1]). If k = (t-1) then $B_0 \in$ A. If k $\neq$ (t-1), then $B_0 \cup \underbrace{\{(l+2),(l+2),\cdots,(l+2)\}}_{(t-1-k)times}$ is an element of A.

These two operations are clearly inverses of each other, which prove the lemma. □

The next two lemma provide us with lower and upper bound for |G|.

Lemma 2.7.21. $|G| \geq \binom{t+l}{t-1}$.

Proof: Note that x is a primitive r^{th} root of unity in F because h(x) is a factor of $\Phi_r(x)$. We now show that if f, g $\in$ P are distinct polynomials with degree less than t then they map to distinct elements in G.

Suppose that f(x) = g(x) in F. Let m $\in$ I. Then m is introspective for f and g, so, f(x^m) = g(x^m) with in F. Then x^m is a root of j(z) = f(z) - g(z) for every m $\in I_r$. By lemma 2.7.4, (m, r) = 1, so, each such x^m is a primitive r^{th} root of unity. Hence their are $|I_r|$ = t distinct roots of j(z) in F. But the degree of j(z) < t, by choice of f and g. This contradiction implies that f(x) $\neq$ g(x) in F.

We can observe that i$\neq$j in $\mathbb{F}_p$ whenever 1$\leq$i,j$\leq$ l, since $l = \lfloor \sqrt{\phi(r)} \cdot \log(n) \rfloor < \sqrt{r} \cdot \log(n)$ < r < p. Then by above, x, x+1, $\cdots$,x+l are all distinct in F. Since the degree of h(x) is greater than 1, all of these linear polynomials are nonzero in F. Therefore there are at least (l+1) distinct polynomials of degree 1 in G. By lemma 2.7.19, there are at least $\binom{l+s}{s}$ polynomials of degree s in G. Then by lemma 2.7.20, the order of G is at least $\sum_{k=0}^{t-1} \binom{l+s}{s} = \binom{t+l}{t-1}$. □

Lemma 2.7.22. *If n is not a power of p, then $|G| \leq n^{\sqrt{t}}$.*

Proof: Consider the following subset of I : I' = $\{\binom{n}{p}^i \cdot p^j : 0 \leq i,j \leq \lfloor \sqrt{t} \rfloor\}$. If n is not a power of p, then |I'| $\geq (1+\lfloor \sqrt{t} \rfloor)^2 >$ t. Since $|I_r|$ = t, there are at least two elements of I' that are equivalent modulo r. Name these elements m_1, m_2 where m_1 > m_2 (WLOG). Then $x^{m_1} \equiv x^{m_2} \pmod{x^r-1}$. Now, let f($x$) $\in$ P. Then, because m_1, m_2 are introspective,

$$f(x)^{m_1} \equiv f(x^{m_1}) \equiv f(x^{m_2}) \equiv f(x)^{m_2} (mod(x^r-1), p)$$

Thus, $f(x)^{m_1} = f(x)^{m_2}$ in the field F. Therefore the polynomial R(y) = y^{m_1} - y^{m_2} has at least |G| roots in F (Since f(x) $\in$ P was arbitrary). Then deg(R(y)) = $m_1 \leq (\frac{n}{p} \cdot p)^{\lfloor \sqrt{t} \rfloor}$ = $n^{\lfloor \sqrt{t} \rfloor}$, because $(\frac{n}{p})^{\lfloor \sqrt{t} \rfloor} p^{\lfloor \sqrt{t} \rfloor}$ is the largest element of I'. It follows that $|G| \leq n^{\sqrt{t}}$. □

Correctness of the AKS algorithm

Theorem 2.7.23. *The AKS algorithm outputs* **Prime** *if and only if* n *is* **Prime**.

Proof: $\Leftarrow$ Suppose n is prime. Then n is certainly not of the form a^b for any a, b>1, so, **step (1)** of the algorithm will not output **composite**. Since n is prime , we also know that for all $x \in \mathbb{N}$, $(n,\ x) = 1$ or n. Hence **step (3)** will not output **composite**. By lemma 2.7.1, we have already seen that n is prime, then $(x + a)^n \equiv x^n + a \pmod{n}$, so **step (5)** will not output **composite**. Therefore the algorithm will output **prime**.
$\Rightarrow$ Now suppose that the algorithm outputs **prime**. If the algorithm returns **prime** during **step (4)**, then we know that for all m $< n$, (m, n)=1 (this was checked in **step(3)**), which means that n is **prime**. Now assume that algorithm returns **prime** during step (6) which means n is introspective for (x+a). Recall that $|\mathrm{I}_r| = \mathrm{t}$ and is generated by n and p. Therefore $\mathrm{t} = \mathrm{o}_r(n) > \log^2(n) \Rightarrow \mathrm{t}^2 > \mathrm{t}\cdot\log^2(n) \Rightarrow \mathrm{t} > \lfloor\sqrt{t}\log(n)\rfloor$. Then lemma 2.5.21 shows that for $l = \lfloor\sqrt{\phi(r)}\cdot\log(n)\rfloor$: $|\mathrm{G}| \geq \binom{t+l}{t-1} \geq \binom{l+1+\lfloor\sqrt{t}log(n)\rfloor}{\lfloor\sqrt{t}log(n)\rfloor} \geq \binom{1+2\lfloor\sqrt{t}log(n)\rfloor}{\lfloor\sqrt{t}log(n)\rfloor}$ (since $\mathrm{I}_r \subset \mathrm{Z}_r^* \Rightarrow \mathrm{t} \leq \varphi(\mathrm{r}) \Rightarrow l = \lfloor\sqrt{\phi(r)}\cdot\log(n)\rfloor \geq \lfloor\sqrt{t}\cdot\log(n)\rfloor$).
Now using lemma 2.7.3,

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

Then by lemma 2.7.22, we know that $|\mathrm{G}| \leq n^{\sqrt{t}}$ if n is not a power of p. Therefore it must be the case that $n = \mathrm{p}^k$ for some k>0. But **step (1)** did not output **composite**, so, k=1, which proves that n is indeed **prime**. This completes the proof of the theorem. □

2.7.4 Runtime Analysis of the Algorithm

Before finding out the running time of the algorithm, we will find out the running time for the Euclidean Algorithm and Perfect power test Algorithm.

Euclidean Algorithm and its running time

Algorithm 2.7.4.1 :
Input m,n $\in \mathbb{Z}$
(1) a, b integer;
(2) if $|n| \geq |m|$

(3) then $a \leftarrow |n|$; b $\leftarrow |m|$;
(4) else b $\leftarrow |n|$; a $\leftarrow |m|$;
(5) while b > 0 repeat
(6) (a, b) = (b, a mod b);
(7) return a.

Lemma 2.7.24. *The complexity of the above algorithm is O(log(n)·log(m)).*

Proof. : Let a_i, b_i denote the values stored in variables (a, b) after the loop in lines 5,6 has been executed for the i^{th} time. Then (m, n) = (a_i, b_i) (which implies that the algorithm returns (m, n)), and the numbers b_1, b_2, $\cdots$ form a strictly decreasing sequence. Now, we begin by showing that the while loop in lines 5, 6 of the algorithm runs at most 2·min$\{||n||, ||m||\}$ times. In the algorithm, we have seen that this loop ends when the strictly decreasing sequence b_1, b_2, $\cdots$ reaches 0. Consider the following two cases :
Case 1 : $b_{i+1} > \frac{1}{2}b_i = \frac{1}{2}a_{i+1}$. Then $b_{i+2} = a_{i+1}$ - $b_{i+1} < \frac{1}{2}b_i$.
Case 2 : $b_{i+1} \leq \frac{1}{2}b_i = \frac{1}{2}a_{i+1}$. Then $b_{i+2} = a_{i+1} \pmod{b_{i+1}} < b_{i+1} \leq \frac{1}{2}b_i$.
We therfore see that for every two executions of the loop, the bit length of the variable b is reduced by 1. It follows that after 2·min$\{||n||, ||m||\}$ many executions of the while loop, b = 0, stopping the loop.
Using elementary methods for dividing allows us to compute a_i mod b_i in $O((||a_i|| - ||b_i|| + 1) \cdot ||b_i||)$ many bit operations. Since $b_i = a_{i+1}$ for $0 \leq i \leq t$. We have : $((||a_i|| - ||b_i|| + 1) \cdot ||b_i||) = ||a_i|| \cdot ||b_i|| - ||a_{i+1}|| \cdot ||b_i|| + ||b_i|| \leq (||a_i|| - ||a_{i+1}||)||b_0|| + ||b_i||$.
Thus we bound the number of bit operations of the algorithm by

$$\sum_{0 \leq i \leq t} O((||a_i|| - ||b_i|| + 1)||b_i||) = O(\sum_{0 \leq i \leq t} ((||a_i|| - ||a_{i+1}||)||b_0|| + ||b_i||))$$

$$= (||a_0|| \cdot ||b_0|| + t \cdot ||b_0||) = O(||n|| \cdot ||m||).$$

□

Fast Modular Exponentiation

Let $n = 2^{a_1} + 2^{a_2} + \cdots + 2^{a_k}$, where $a_1 > a_1 > \cdots > a_k$. Define $f_0 = (x + a)$, $f_{i+1} = f_i(x)^2 \pmod{x^r\text{-}1, n}$. Then $f_{a_j}(x) = (x + a)^{2^{a_j}}$. If we further define $g_1(x) = f_{a_1}(x)$ and $g_t(x) = g_{t-1}(x)f_{a_t}(x) \pmod{x^r\text{-}1, n}$, then we see that $g_k(x) \equiv (x + a)^{2^{a_1} + \cdots + 2^{a_k}} = (x + a)^n \pmod{x^r\text{-}1, n}$. We have therefore computed $(x+a)^n \pmod{x^r\text{-}1, n}$ in $a_1 + a_k \leq 2\log(n)$ steps, where a step consists of multiplying two polynomials of degree less

than r with coefficients in $\mathbb{Z}/n\mathbb{Z}$. This leads to a total runtime of $O^{\sim}(r \cdot \log^2(n))$.
Idea of the running time in a simple way : We want to discuss a simple algorithm for total running time of Fast Modular Exponentiation is $O^{\sim}(r \cdot \log^2(n))$ in following steps:
(1) In computing the $g_k(x)$: we use the following method :-
$(x+a)^2 = x^2 + 2x \cdot a + a^2$; number of multiplication = 3
$(x^2 + (2xa + a^2))^2 = (x^2)^2 + 2x^2 \cdot (2xa + a^2) + (2xa + a^2)^2$; number of multiplications = 5
We will continue this procedure and the power of the function is bounded by r and we are taking the coefficients mod(n). Hence approximately total running time is $O((2r+1)\log^2(n))$.
(2) for computing $g_k(x)$, we use function $g_t(x)$ which is product of two functions. So we proceed as follows: $4 \cdot g_{t-1}(x) \cdot f_{a_t}(x) = (g_{t-1}(x)+f_{a_t}(x))^2 - (g_{t-1}(x)-f_{a_t}(x))^2$.
Therefore total number of running time (approximately) is $O^{\sim}(r \cdot \log^2(n))$.

Perfect power test Algorithm and its running time

Algorithm 2.7.4.2 :
Input $n \in \mathbb{N}$.
(1) a, b, c, m $\in \mathbb{Z}$
(2) b $\leftarrow$ 2
(3) while b $\leq \log(n)$ do //Loop 1
(4) a =1; c = n;
(5) while c - $a \geq 2$ do //Loop 2
(6) m $\leftarrow$ (a + c) div 2;
(7) p $\leftarrow \min\{m^b, 1\}$;
(8) if p = n then return "n is a perfect power";
(9) if p<n then $a \leftarrow$ m else c $\leftarrow$ m;
(10) b $\leftarrow$ b+1;
(11) return "n is not a perfect power".

Suppose we are given an integer n and want to determine whether or not n is a perfect power, we will be conducting a binary search of the integers $\{1, 2, \cdots, n\}$ for a number m such that $n = m^b$ for some b>1. Let b>1 and suppose that if a solution m to $m^b = n$ exists then it must lie in some interval $[c_i, d_i]$. When i = 0 we may take $[c_0, d_0] = [1, n]$. Now, to define $[c_{i+1}, d_{i+1}]$, consider $\alpha = \lfloor \frac{(c_i+d_i)}{2} \rfloor$. If $\alpha^b = n$, then we

are done. If $\alpha^b > n$, let $[c_{i+1}, d_{i+1}] = [c_i, \alpha]$; otherwise $\alpha^b < n$ and we let $[c_{i+1}, d_{i+1}] = [\alpha, d_i]$. We continue in this manner until $|c_{i+1} - d_{i+1}| \leq 1$. We then increase the value stored in variable b and start the loop again. Performing this loop for all b $\leq \log(n)$ completes our algorithm.

Lemma 2.7.25. *The complexity of the above algorithm is $O^{\sim}(log^3(n))$.*

Proof. : Loop (1) will run at most $\log(n)$ times. Also, it will take atmost $\log(n)$ iterations of loop 2 before $|c-a| \leq 1$. During each iteration of loop 2, we calculate (a + c) div 2 and m^b, which can be done in $O^{\sim}(\log(n))$ [Fact 2 in the Asymptotic Notation] bit operations. The complexity of the entire algorithm is therfore $O^{\sim}(\log^3(n))$. □

Runtime Analysis of AKS Algorithm

Lemma 2.7.26. *The complexity of the above algorithm is $O^{\sim}(log^{10.5}(n))$.*

Proof. : **Step (1)**

We have shown in lemma 2.7.25 that **step (1)** will take at most $O^{\sim}(\log^3(n))$ bit operations.

Step (2)

In this step the algorithm finds the list r such that $o_r(n) > \log^2(n)$. By lemma 2.7.4, there exists an r less than $\lceil log^5(n) \rceil$. The easiest way to find such an r is simply to calculate $n^k \pmod r$ for k = 1,2, $\cdots$,$\log^2(n)$. This involves $O(\log^2(n))$ multiplications modulo r for each r, so **step (2)** takes $O(\log^7(n))$ bit operations.

Step (3)

In this step the algorithm computes (a, n) for $a = 1, \cdots,$ r in order to determine whether $(a, n) > 1$ for some $a \leq$ r. Computing each gcd takes $O^{\sim}(\log^2(n))$ bit operations using Euclidean Algorithm (lemma 2.7.24), resulting in a total of $O^{\sim}(\log^7(n))$.

Step (5)

During this step the algorithm determines whether the congruence $(x + a)^n \equiv (x^n + a) \pmod{x^r\text{-}1, n}$ holds for $a = 1,2,3,\cdots,\lfloor\sqrt{\phi(r)}\cdot\log(n)\rfloor$. Given $a \leq \lfloor\sqrt{\phi(r)}\cdot\log(n)\rfloor$, we may calculate $(x + a)^n$ in the ring $\mathbb{Z}/n\mathbb{Z}$ as reducing modulo x^r-1 is trivial. In order to calculate $(x + a)^n$, we must perform $O(\log(n))$ multiplications of polynomials of degree less than r with coefficients of size $O(\log(n))$ (as the coefficients are in written modulo n; recall that all polynomials are reduced modulo x^r-1 during Fast Modular Exponentiation). Each congruence therfore takes $O^{\sim}(\log^7(n))$ bit operations to verify. **step (5)** therfore takes $O^{\sim}(\sqrt{\phi(r)}\log(n)\log^7(n)) = O^{\sim}(\sqrt{\phi(r)}\log^8(n)) = O^{\sim}(\log^{\frac{21}{2}}(n))$

bit operations. The complexity of the **step (5)** clearly dominates the complexity of the other steps, so the overall complexity of the algorithm is $\tilde{O}(\log^{10.5}(n))$, which is indeed polynomial. □

Part II

Integer Factorization and Polynomial Factorization

Chapter 3

Integer factorization algorithms

Suppose we know that a certain large odd integer n is composite; for example, we found that it fails one of the primality tests in Chapter 2. As mentioned before in introduction part that it does not mean that we have any idea of what a factor of n might be. Of the methods we have encountered for testing primality, only the very slowest - trying to divide by successive primes less than $\sqrt{n}$ - actually gives us a prime factor at the same time as it tells us that n is composite. All of the faster primality test algorithms are more indirect : they tell us that n must have proper factors, but not what they are. The method of trial division by primes $< \sqrt{n}$ can take more than $O(\sqrt{n})$ bit operations. In this chapter, we will discuss faster algorithms of integer factorization.

3.1 Pollard Rho Method

The simplest algorithm which is substantially faster than trial division is J.M. Pollard's "rho method" (also called the "Monte Carlo" method) of factorization.

3.1.1 Algorithm

Algorithm follows in the following steps :
(1) Choose an easily evaluated map from $\mathbb{Z}/n\mathbb{Z}$ to itself, a fairly simple polynomial with integer coefficients, such as x^2+a. [Note : Later we can observe it from proposition 3.1.4 that f(x) must not be linear polynomial and in fact, should not give a 1-to-1 map.]
(2) Choose some particular value $x = x_0$ (perhaps $x_0 = 1$ or 2, or perhaps it is a randomly generated integer).

(3) Compute the successive iterates of f : $x_1 = f(x_0)$, $x_2 = f(f(x_0))$, $x_3 = f(f(f(x_0)))$, etc. That is we define $x_{j+1} = f(x_j)$, j = 0,1,2,$\cdots$
(4) Compare between different x_j's and find two of which are in different residue classes modulo n but in the same residue class modulo some divisor of n (let denote them x_j and x_k) .
(5) Compute gcd(x_j - x_k, n), which is equal to a proper divisor of n. [**Note** : After providing an example, we will describe a way to carry out the algorithm so as to make only one g.c.d. computation for each k.]

Example 3.1.1. *: Let us factor 91 by choosing f(x) = x^2+1, x_0 = 1. Then we have x_1 = 2, x_2 = 5, x_3 = 26, etc. We can find gcd(x_3-x_2, n) = (21, 91) = 7, so 7 is a factor.*

3.1.2 Refinement of the Algorithm :

As we know that rho method works by successively computing $x_k = f(x_{k-1})$ and comparing x_k with the earlier x_j until we find a pair satisfying gcd(x_k - x_j, n) = r > 1. But as k becomes large, it becomes very time consuming to have to compute gcd(x_k-x_j, n) for each $j < k$. We now describe a way to carry out the algorithm so as to make only one gcd computation for each k.
First observe that, once there is a k_0 and j_0 such that $x_{k_0} \equiv x_{j_0}$ mod r for some divisor r|n, we then have the same relation that $x_k \equiv x_j$ mod r for any pair of indices j, k having the same difference k-j = k_0-j_0. To see this, simply set $k = k_0 + m$, $j = j_0 + m$, and apply the polynomial f to both sides of the congruence $x_{k_0} \equiv x_{j_0}$ mod r repeatedly, i.e., m times.

We now describe how the refined rho algorithm works in Step (4). We successively compute the x_k, and for each k we proceed as follows :
(1) Suppose k is an (h+1)-bit integer, i.e., $2^h \leq k < 2^{h+1}$. Let j be the largest h-bit integer : $j=2^h-1$.
(2) Compare x_k with this particular x_j, i.e., compute gcd(x_k-x_j, n).
(3) If this gcd gives a nontrivial factor of n, we stop; otherwise we move on to k+1.

This modified approach has the advantage that we compute only one gcd for each k. It has the disadvantage that we probably will not detect the first time there is a k_0 such that gcd(x_{k_0}-x_{j_0}, n) = r > 1 for some $j_0 < k_0$.

However before long we will detect such a pair x_k, x_j whose difference has a common factor with n. Namely, suppose that k_0 has (h+1) bit integer. Set $j = 2^{h+1}$-1 and $k = j+(k_0-j_0)$, in which case j is the largest (h+1)-bit integer and k is an (h+2) bit integer such that $gcd(x_k-x_j, n) > 1$.
Notice that, we have $k < 2^{h+2} = 4\cdot 2^h \leq 4\cdot k_0$.

Example 3.1.2. *: Suppose we have to factor 4087 using $f(x) = x^2+x+1$ and $x_0 = 2$. Then $x_1 = f(2) = 7$ and $gcd(x_1-x_0, n) = gcd(7-2,4087) = 1$; $x_2 = f(7) = 57$ and $gcd(x_2-x_1, n) = gcd(57-7, 4087) = 1$; $x_3 = f(57) = 3307$ and $gcd(x_3-x_1, n) = gcd(3307-7, 4087) = 1$; $x_4 \equiv f(3307) \equiv 2745 \bmod 4087$ and $gcd(x_4-x_3, n) = gcd(2745-3307, 4087) = 1$; $x_5 \equiv f(2745) \equiv 1343 \bmod 4087$ and $gcd(x_5-x_3, n) = gcd(1343-3307, 4087) = 1$; $x_6 \equiv f(1343) \equiv 2626 \bmod 4087$ and $gcd(x_6-x_3, n) = gcd(2626-3307, 4087) = 1$; $x_7 \equiv f(3734) \equiv 2734 \bmod 4087$ and $gcd(x_7-x_3, n) = gcd(3734-3307, 4087) = 61$.*
Thus we obtain $4087 = 61\cdot 67$.

3.1.3 Running time Analysis

Let us suppose f(x) is a random map from $\mathbb{Z}/n\mathbb{Z}$ to itself and compute how long we expect to have to wait before we have two iterations x_j and x_k such that x_j-x_k has a nontrivial common factor with n. We do this by finding for a fixed divisor r of n (which, in practice, is not yet known to us) the average (taken over all maps from $\mathbb{Z}/n\mathbb{Z}$ to itself and over all values of x_0) of the first index k such that there exists $j<k$ with $x_j \equiv x_k \bmod r$. In other words, we regard f(x) as a map from $\mathbb{Z}/r\mathbb{Z}$ to itself and ask how many iterations are required before we encounter the first repetition of values $x_k = x_j$ in $\mathbb{Z}/r\mathbb{Z}$.

Proposition 3.1.3. *Let S be a set of r elements. Given a map f from S to S and an element $x_0 \in S$, let $x_{j+1} = f(x_j)$ for $j = 0,1,2,\cdots$. Let λ be a positive real number, and let $m = 1+[\sqrt{2\lambda r}]$. Then the proportion of pairs (f, x_0) for which $x_0, x_1, \cdots, x_m$ are distinct, where f runs over all maps from S to S and x_0 runs over all elements of S, is less than $e^{-\lambda}$.*

Proof : *There are r choices of x_0, and for each of the r different $x \in S$ there are r choices of f(x). So, the total number of pairs are r^{r+1}. Now we will check that how many pairs (f, x_0) are there for which $x_0, x_1, \cdots, x_m$ are distinct. There are r choices for x_0, there are r-1 choices for $f(x_0) = x_1$ (since this can not be equal to x_0), there are r-2 choices for $f(x_1) = x_2$, and so on, until f(x) has been defined for $x = x_0, x_1, \cdots, x_{m-1}$. Then the value of f(x) for each of the r-m remaining x is arbitrary, i.e.,*

there are r^{r-m} possibilities for those values. Hence, the total number of possible ways of choosing x_0 and assigning the values f(x) so that x_0, x_1, $\cdots$, x_m are distinct is : (r-m)$\cdot r^{r-m} \cdot \prod_{j=0}^{m-1}$(r-j) = $r^{r-m} \cdot \prod_{j=0}^{m}$(r-j), and the proportion of pairs having the stated property (i.e., the above number divided by r^{r+1}) is : $\frac{r^{r-m} \cdot \prod_{j=0}^{m}(r-j)}{r^{r+1}} = r^{-m-1} \prod_{j=0}^{m}$(r-j) = $\prod_{j=1}^{m}(1-\frac{j}{r})$.

Now we have to show that log of this is less than -λ (where m = 1+$[\sqrt{2r\lambda}]$). To prove this, we take the log of the product and use the fact that log(1-x) < -x for 0 < x < 1. So, we have :

$$log(\prod_{j=1}^{m}(1-\frac{j}{r})) < \sum_{j=1}^{m} -\frac{j}{r} = \frac{-m(m+1)}{2r} < \frac{-m^2}{2r} < \frac{-(\sqrt{2r\lambda})^2}{2r} = -\lambda$$

This completes the proof of the proposition. □

The significance of proposition 3.1.3 is that it gives an estimate for the probable length of time of the rho method, provided that we assume that our polynomial behaves like an average map from $\mathbb{Z}/r\mathbb{Z}$ to itself.

Proposition 3.1.4. *Let n be an odd composite integer, and let r be a nontrivial divisor of n which is less than $\sqrt{n}$ (i.e., r|n, 1 < r < $\sqrt{n}$; we suppose that we are trying to determine what r is). If a pair (f, x_0) consisting of a polynomial f with integer coefficients and an initial value x_0 is chosen which behaves like an average pair (f, x_0) in the sense of proposition 3.1.3 (with f a map from $\mathbb{Z}/r\mathbb{Z}$ to itself and x_0 an integer), then the rho method will reveal factor r in O((n)$^{\frac{1}{4}} \cdot log^2 n$) bit operations with a high probability. More precisely, there exists a constant C such that for any positive real number λ the probability that the rho method fails to find a nontrivial factor of n in C$\sqrt{\lambda}$(n)$^{\frac{1}{4}} log^2 n$ bit operations is less than $e^{-\lambda}$.*

Proof : *Let C_1 be a constant such that gcd(y-z, n) can be computed in $C_1 log^2 n$ bit operations whenever y, z ≤ n (proved in lemma 2.7.24). Let C_2 be a constant such that the least nonnegative residue of f(x) modulo n can be computed in $C_2 log^2 n$ bit operations whenever x < n. If k_0 is the first index for which there exists $j_0 < k_0$ with $x_{k_0} \equiv x_{j_0}$ mod r, then the refined rho algorithm as described above finds r in the k^{th} step, where k < 4k_0.*

Thus the number of bit operations needed to find r is bounded by 4$\cdot k_0$($C_1 log^2 n$ + $C_2 log^2 n$). According to proposition 3.1.3, the probability that k_0 is not greater than 1+$\sqrt{2r\lambda}$ is less than $e^{-\lambda}$. If k_0 is not greater than 1+$\sqrt{2r\lambda}$, then the number of bit

operations need to find r is bounded by (here $r < \sqrt{n}$) : $4(1+\sqrt{2r\lambda})(C_1+C_2)log^2 n < 4(1+\sqrt{2}\sqrt{\lambda}(n)^{\frac{1}{4}})(C_1+C_2)log^2 n$.
If we choose C slightly greater than $4\sqrt{2}(C_1+C_2)$ (so as to take care of the added 1), we conclude, as claimed, that the factor r will be found in $C\sqrt{\lambda}(n)^{\frac{1}{4}}log^2 n$ bit operations, unless we made an unfortunate choice of (f, x_0), of which the likelihood is less than $e^{-\lambda}$. □

Thus according to proposition 3.1.4, if we choose λ large enough to have confidence in success - for example, $e^{-\lambda}$ is only about 0.0001 for $\lambda = 9$ - then we know that for an average pair (f, x_0) we are almost certain to factor n in $3C(n)^{\frac{1}{4}}log^2 n$ bit operations.

3.2 Pollard p-1 method

This is a classical factoring technique. Suppose we want to factor the composite number n, and p is some (as yet unknown) prime factor of n. If p happens to have the property that p-1 has no large prime divisor, then this method is virtually certain to find p.

3.2.1 Algorithm

(1) Choose an integer k that is a multiple of all or most integers less than some bound B. For example, k might be B!, or it might be the least common multiple of all integers $\leq B$.
(2) Choose an integer a between 2 and n-2. For example, a could equal 2, or 3, or a randomly chosen integer.
(3) Compute a^k mod n by the repeated squaring method.
(4) Compute d = gcd(a^k-1, n) using the Euclidean algorithm and the residue of a^k modulo n from step 3.
(5) If d is not a nontrivial divisor of n, start over with a new choice of a and/or a new choice of k.

Now we will explain that when this algorithm will work. Suppose that k is divisible by all positive integers $\leq B$, and further suppose that p is a prime divisor of n such that p-1 is a product of small prime powers, all less than B. Then k is a multiple of p-1 because it is a multiple of all the prime powers in the factorization of p-1. So, by Fermat little theorem 2.4.1, we have $a^k \equiv 1$ mod p, which means p|gcd(a^k-1,n). We

can observe that the only way in which we will get a nontrivial factor of n in step (4) is when $a^k \equiv 1 \bmod n$.

Example 3.2.1. *Suppose we have to factor 540143 by this method, choosing B = 8 and a = 2. Hence, we can get k = 840, which is the least common multiple of 1,2,$\cdots$,8. We find 2^{840} mod 540143, which is equal to 53047. So, gcd(53046, 540143) = 421 provides us a nontrivial factor of n.*

The main weakness of the Pollard p-1 method is clear that if we attempt to use it when all of the prime divisors p of n have p-1 divisible by a large prime (or prime power). In the following example, due to this weakness, it is very difficult to find a nontrivial factor of n by this algorithm.

Example 3.2.2. *Let n = 491389. We would be unlikely to find a nontrivial divisor until we choose $B \geq 191$. This is because it turns out that n = 383·1283. We have 383-1 = 382 = 2·191 and 1283-1 = 2·641 (both 191 and 641 are primes). Except a $\equiv 0, \pm 1 \bmod 383$, all other a's have order modulo 383 either 191 or 382; and except for $a \equiv 1, \pm 1 \bmod 1283$, all other a's have order modulo 1283 either 641 or 1282. So unless k is divisible by 191 (or 641), we are likely to find again and again that gcd(a^k-1, n) = 1 in step (4).*

The basic dilemma with Pollard's p-1 method is that we are pinning our hopes on the group $(\mathbb{Z}/p\mathbb{Z})^*$ (more precisely, the various such groups as p runs through the prime divisors of n). For a fixed n, these groups are fixed. If all of them happen to have order divisible by a large prime, then we are stuck.

3.2.2 Complexity of the algorithm

Theorem 3.2.3. *The complexity of the algorithm is bounded by $O(n^2 log^3(n))$.*

Proof : *In Step (1), If we take k = B! then it will take $O(B^2 log^2(B))$ (by simple naive algorithm), or we can say it is bounded by $O(n^2 log^2(n))$.*
In Step (3), we have described a method to compute a^k mod n in the Miller-Rabin primality test, which takes $O(log^3(n))$.
In Step (4), computing d will take $O(log^2(n))$ by Euclidean algorithm, which is described in the last subsection of AKS algorithm.
Thus the overall complexity of the algorithm is $O(n^2 log^3(n))$. □

3.3 Fermat Factorization and Fermat Factor base Method

3.3.1 Fermat Factorization method

This method is efficient if n is a product of two integers which are close to one another.

3.3.2 Algorithm (Fermat Factorization)

(1) Compute t = [$\sqrt{n}$]+1, [$\sqrt{n}$]+2, $\cdots$ until we obtain a t for which t^2 - n = s^2 is a perfect square (s, t are nonnegative integers).
(2) gcd(t+s, n) is a nontrivial factor of n.

This algorithm is based on the following proposition :

Proposition 3.3.1. *Let n be a positive odd integer. There is a one to one correspondence between factorizations of n in the form n = a·b, where a ≥ b > 0, and representations of n in the form t^2 - s^2, where s and t are nonnegative integers. The correspondence is given by the equations :*

$$t = \frac{a+b}{2}, s = \frac{a-b}{2}; a = t+s, b = t-s.$$

Proof : *Given such a factorization, we can write n = a·b = ((a+b)/2)² - ((a-b)/2)², so we obtain the representation as a difference of two squares.*
Conversely, given n = t^2 - s^2 we can factor the right side as (t+s)(t-s). The equations in the proposition explicitly give the one to one correspondence between the two ways of writing n. □

Example 3.3.2. *Suppose we have to factor 200819. Then firstly, we compute [$\sqrt{200819}$]+1 = 449. But we observe that 449^2-200819 = 782, which is not a perfect square. Next, we try t = 450 and we get 450^2-200819 = 1681 = 41^2. Thus, 200819 = 450^2-41^2 = (450+41)(450-41) = 491·409.*

If n = a·b with a and b close together, then s = (a-b)/2 is small, and t is only slightly larger than $\sqrt{n}$. In this case, we can find a and b by trying all values fot t starting with [$\sqrt{n}$]+1, until we find one for which t^2-n = s^2 is a perfect square.
But we can notice that if a and b are not close together for any factorization n = a·b, then the Fermat factorization method will eventually find a and b, but only after

trying a large number of t = [$\sqrt{n}$]+1, [$\sqrt{n}$]+2, $\cdots$. There is a generalization of the Fermat factorization that often works better in such a situation.

3.3.3 Generalized Fermat factorization Algorithm

(1) Choose a small k, and succesively set t = [$\sqrt{kn}$]+1, [$\sqrt{kn}$]+2, $\cdots$, until we obtain a t for which t^2-kn = s^2 is a perfect square.
(2) gcd(t+s, n) is a nontrivial factor of n.

Since in the generalized Fermat factorization algorithm, kn = t^2-s^2 = (t+s)(t-s). It means that t+s has a nontrivial common factor with n which can be found by computing gcd(t+s, n).

Example 3.3.3. *Let we have to factor 141467. Then if we use Fermat factorization method, setting t = [$\sqrt{141467}$]+1 = 377, 378, $\cdots$, after a while we tire of trying different t's. However if we use generalized Fermat factorization and setting t = [$\sqrt{3n}$]+1 = 652, 653, $\cdots$. We soon find that 655^2-3·141467 = 68^2, at which point we compute gcd(655+68,141467) = 241. We conclude that 141467 = 241·587.*
The reason why generalized Fermat factorization worked with k =3 in the above example is that there is a factorization n = a·b with b close to 3a. With k = 3, we need to try only four t's, whereas with simple Fermat factorization (i.e. k = 1) it would have taken thirty eight t's.

3.3.4 Fermat factor base method

factor bases :

There is a generalization of the idea behind Fermat factorization which leads to a much more efficient factoring method. Namely, we use the fact that any time we are able to obtain a congruence of the form $t^2 \equiv s^2$ mod n with $t \neq \pm s$ mod n, we immediately find a factor of n by computing gcd(t+s, n) (or gcd(t-s, n)). This is because we have gcd(t+s, n) must be a proper factor a of n, and then b = n/a divides gcd(t-s, n).

The main problem in this algorithm is to select a random various b for which the least positive residue of b^2 mod n is a perfect square. That is very unlikely if n is large, so it is necessary to generalize this method in a way that allows much greater flexibility in choosing the b_i's which have the property that b_i^2 mod n is a product of

small prime powers, and such that some subset of them, when multiplied together, give a b whose square is congruent to a perfect square modulo n.

Definition 3.3.4. *A factor base is a set $B = \{p_1, p_2, \cdots, p_h\}$ of distinct primes, except that p_1 may be the integer -1. We say that the square of an integer b is a B-number (for a given n) if the least absolute residue b^2 mod n can be written as a product of numbers from B.*

Note (1) : *By the "least absolute residue" of a number a modulo n we mean the integer in the interval from -n/2 to +n/2 to which a is congruent. We shall denote this a mod n.*

Note (2) : *Let $\mathbb{F}_2^h$ denote the vector space over the field of two elements which consists of h-tuples of zeroes and ones. Given n and a factor base B containing h numbers, we show how to correspond a vector $\overrightarrow{\epsilon} \in \mathbb{F}_2^h$ to every B-number. Namely, we write b^2 mod n in the form $\prod_{j=1}^{h} p_j^{\alpha_j}$ and the j^{th} component ϵ_j equal to α_j mod 2, i.e., $\epsilon_j = 0$ if α_j is even, and $\epsilon_j = 1$ if α_j is odd.*

3.3.5 Idea of the algorithm

Suppose that we have some set of B-number b_i^2 mod n such that the corresponding vectors $\overrightarrow{\epsilon_i} = \{\epsilon_{i1}, \epsilon_{i2}, \cdots, \epsilon_{ih}\}$ add up to the zero vector in $\mathbb{F}_2^h$. Then the product of the least absolute residues of b_i^2 is equal to a product of even powers of all of the p_j in B. That is, if for each i we let a_i denote the least absolute residue of b_i^2 mod n and we write $a_i = \prod_{j=1}^{h} p_j^{\alpha_{ij}}$, we obtain

$$\prod_i a_i = \prod_{j=1}^{h} p_j^{\sum_i \alpha_{ij}} = (\prod_j p_j^{\gamma_j})^2$$

where $\gamma_j = \frac{1}{2}\sum_i \alpha_{ij}$ (we can write this because the exponent of each p_j is an even number). Thus if we set $b = \prod_i b_i$ mod n (least positive residue) and $c = \prod_j p_j^{\gamma_j}$ mod n (least positive residue), we obtain two numbers b and c, constructed in quite different ways (one as a product of b_i's and the other as a product of p_j's) whose squares are congruent to modulo n. Now if $b \neq \pm c$ mod n then gcd(b+c, n) gives us a nontrivial factor of n.

Note (3): *It may happen that $b \equiv \pm c \bmod n$, in which case we are out of luck, and we must start again with another collection of B-numbers whose corresponding vectors sum to zero. This will happen, for example, if we foolishly choose b_i less than $\sqrt{n/2}$, in which case all of the vectors are zero vectors, and we end up with a trivial congruence.*

Selection of B and b_i :

(1) One method is to start with B consisiting of the first h primes (or the first h-1 primes together with p_1 = -1) and choose random b_i's until we find several whose squares are B-numbers.
(2) Another method is to start by choosing some b_i's for which $b_i^2 \bmod n$ (least positive residue) is small in absolute value (for example, take b_i close to $\sqrt{kn}$ for small multiples kn ; one another method for this we will describe in next section). Then choose B to consist of a small set of small primes (and usually p_1 = -1) so that several of the $b_i^2 \bmod n$ can be expressed in terms of the numbers in B.

Now if we choose the first method, then the question arise that, when can we be sure that we have enough b_i to find a sum of $\overrightarrow{\epsilon_i}$ which is the zero vector ? In other words, given a collection of vectors in $\mathbb{F}_2^h$, when can we be sure of being able to find a subset of them which sums to be zero ? For this, it is enpogh to find out the collection of vectors which are linearly dependent over the field $\mathbb{F}_2$. According to the basic linear algebra (which applies as well over the field $\mathbb{F}_2$ as over the real numbers), this is guaranteed to occur as soon as we have h+1 vectors. Thus, at worst we will have to generate h+1 different B-numbers in order to find our first example of $(\prod_i b_i)^2 = (\prod_j p_j^{\gamma_j})^2 \bmod n$. It can also possible that we can get the linearly dependent vectors before h+1 vectors.

Note (4) : *For more randomly chosen b_i, because n is composite, we would expect that b and c would happen to be congruent (up to ± 1) modulo n at most 1/2 of the time. This is because any square modulo n has $2^r \geq 4$ square roots if n has r different prime factors. Therefore a random square root of b^2 has only a $2/2^r \leq \frac{1}{2}$ chance of being either b or -b. Thus, if we go through the above procedure for finding b and c until we find a pair that gives us a nontrivial factor of n, we see that there is at most a 2^{-k} probability that this will take more than k tries.*

We now summarize a systematic method to factor a very large n using a random choice of the b_i.

3.3.6 Factor base Algorithm

(1) Choose an integer y of intermediate size, for example, if n is a 50 decimal digit integer, we might choose y to be a number with 5 or 6 decimal digits.
(2) Set B consists of -1 and all primes $\leq y$.
(3) Choose a large number of random b_i (close to $\sqrt{kn}$ for small integers k, like $[\sqrt{kn}]$, $[\sqrt{kn}]+1, \cdots$), and try to express b_i^2 mod n (least absolute residue) as a product of the primes in B.
(4) From step (3), obtain a large quantity of B-numbers b_i^2 mod n ($\pi(y)+2$ is enough, where $\pi(y)$ denotes the number of primes $\leq y$).
(5) Take the corresponding vectors in $\mathbb{F}_2^h$ (where $h = \pi(y)+1$) and by row reduction determine a subset of the b_i whose corresponding $\overrightarrow{\epsilon_i}$ sum to zero.
(6) Set $b = \prod_i b_i$ mod n and $c = \prod_j p_j^{\gamma_j}$ mod n (as described before Note (3)). Then $b^2 \equiv c^2$ mod n.
(7) If $b \equiv \pm c$ mod n, repeat the process with a new random collection of B-numbers (or, to be more efficient, choose different subset of rows in the matrix of $\overrightarrow{\epsilon_i}$'s which sum is zero, if necessary find a few more B-numbers and there corresponding rows).
(8) If $b \neq c$ mod n, compute gcd(b+c, n), which will be nontrivial factor of n.

Some examples :

Example 3.3.5. *Suppose we have to factor n = 4633. Set B = {-1, 2,3,5}. Choose $b_1 = [\sqrt{n}] = 68$, $b_2 = [\sqrt{n}]+1 = 69$, $b_3 = [\sqrt{2n}] = 96$, etc. Then we see that 68, 69 and 96 are B-numbers because $68^2 = -9$ mod 4633 which has corresponding vector {1,0,0,0}; $69^2 = 128$ mod 4633 which has corresponding vector {0,1,0,0}; and $96^2 = -50$ mod 4633 which has corresponding vector {1,1,0,0}. Since the sum of the these three vectors is zero, we can take $b = 68 \cdot 69 \cdot 96 \equiv 1031$ mod 4633 and $c = 2^4 \cdot 3 \cdot 5 = 240$. So, we see that $b \neq \pm c$ mod 4633. Therefore, we obtain gcd(240+1031, 4633) = 41, a nontrivial factor.*

Example 3.3.6. *Let us factor n = 1829. Take b_i's = $[\sqrt{kn}]$ and $[\sqrt{kn}]+1$ for k = 1,2,$\cdots$, such that b_i^2 mod n is a product of primes less than 20. For each b_i, we write b_i^2 mod $n = \prod_j p_j^{\alpha_{ij}}$ and tabulate the α_{ij}. After taking k =1,2,3,4, we have the following table, in which the number at the top of the j^{th} column is p_j and the entry in the i_{th} row beneath p_j is the power of p_j which occurs in b_i^2 mod n :*

b_i	-1	2	3	5	7	11	13
42	1	-	-	1	-	-	1
43	-	2	-	1	-	-	-
61	-	-	2	-	1	-	-
74	1	-	-	-	-	1	-
85	1	-	-	-	1	-	1
86	-	4	-	1	-	-	-

We now look for a subset of rows whose entries sum to an even number in each column. We see that 2^{nd} and 6^{th} rows sum to the even row i.e. - 6 - 2 - - - , which leads to the congruence $(b_2 \cdot b_6)^2 \equiv (2^{6/2} \cdot 5^{2/2})^2$ mod $n = (43 \cdot 86)^2 \equiv 40^2$ mod 1829. But $43.86 \equiv 40$ mod 1829, so here we find a trivial relationship. Thus we have to look for another subset of rows which sum to a row of even numbers. We notice that the sum of the first three rows and fifth row is 2 2 2 2 2 - 2 , and this gives the congruence $(42 \cdot 43 \cdot 61 \cdot 85)^2 \equiv (2 \cdot 3 \cdot 5 \cdot 7 \cdot 13)^2$ mod 1829, i.e., $1459^2 \equiv 901^2$ mod 1829. Thus we conclude that a factor of 1829 is gcd(1459+901, 1829) = 59.

3.3.7 Running time analysis

We now give a very rough derivation of an estimate for the number of bit operations it takes to find a factor of a very large n using the algorithm described above. We shall use several simplyfying assumptions and approximations, and in any case the result will only be a probabilistic estimate. If we are unlucky in our random choice of b_i, then the algorithm will take longer.

Before estimating the running time of the algorithm, we need following lemma's, we will describe only the idea of the proof of the lemma's :

Lemma 3.3.7. Stirling formula : *For $n \in \mathbb{N}$,*

$$lim_{n\to\infty} \frac{n!}{\sqrt{2n\pi} n^n e^{-n}} = 1 \quad or \quad n! \approx \sqrt{2\pi} n^{n+\frac{1}{2}} e^{-n}$$

or we can say log(n!) is approximately nlogn-n.

Idea of the Proof : *This can be proved by observing that log(n!) is the right-endpoint Riemann sum (with endpoints at 1,2,3,$\cdots$) for the definite integral $\int_1^n logxdx$ = nlogn - n + 1.* □

Lemma 3.3.8. *Given a positive integer N and a positive number u, the total number of nonnegative integer N-tuples α_j such that $\sum_{j=1}^{N} \alpha_j \leq u$ is the binomial coefficient $\binom{[u]+N}{N}$.*

Idea of the Proof : Each N-tuple solution α_j correspond to the following choice of N integers β_j from among 1,2,$\cdots$,[u]+N. Let $\beta_1 = \alpha_1+1$, and for j $\geq$ 1, let $\beta_{j+1} = \beta_j+\alpha_{j+1}+1$, i.e., we choose the β_j's so that there are α_j numbers between β_{j-1} and β_j. This gives a one to one correspondence between the number of solutions and the number of ways of choosing N numbers from a set of [u]+N numbers. □

Proposition 3.3.9. *The probability that a random number less than x is a product of primes less than y (where y is a number much less than x) will be $\frac{1}{u^u}$, where u denote the ratio $\frac{logx}{logy}$.*

Proof : Let x be an r-bit integer and y is an s-bit integer, then u is approximately the ratio of r/s. Now, let π(y) denote the number of prime numbers which are $\leq$ y. We know by "Prime Number Theorem" that π(y) is approximately equal to y/log(y). Now since y is much less than x, so, u is much smaller than y and π(y). In a typical practical application of the algorithm, we might take y, u, x of approximately the following sizes : $x \approx 10^{48}$; y $\approx 10^6$; so that u $\approx$ 8 and π(y) $\approx 7{\cdot}10^4$ and log(y) $\approx$ 14. Let $\psi(x,$ y) denote the number of integers $\leq x$ which are not divisible by any prime greater than y, i.e., the number of integers which can be written as a product $\prod_j p_j^{\alpha_j}$ $\leq x$, where the product is over all primes $\leq$ y and α_j are non negative integers.
We can observe that there is one-to-one correspondence between π(y)-tuples of non-negative integers α_j for which $\prod_j p_j^{\alpha_j} \leq x$ and integers $\leq x$ which are not divisible by any prime greater than y. Thus, $\psi(x$,y) is equal to the number of integer solutions α_j to the inequality $\sum_{j=1}^{\pi(y)} \alpha_j \log p_j \leq \log(x)$ (by taking logarithm).
We can now observe that most of the p_j's have logarithms not too much less than log(y) because most of the primes less than y have almost the same number of digits as y; only relatively few have many fewer digits and hence a much smaller logarithm. Thus we can replace $\log p_j$ by log(y) in the previous inequality. After dividing both sides of the resulting inequality by log(y) and replacing $\log x$/log(y) by u, we can say that $\psi(x$,y) is approximately equal to the number of solutions of the inequality $\sum_{j=1}^{\pi(y)} \alpha_j \leq$ u.
We now make another important simplification, replacing the number of variables π(y) by y. This might appear at first to be a rather reckless modification of our prob-

lem. And in fact, replacing π(y) by y does introduce nontrivial terms; however , it turns out that those terms cancel, and the net result is the same as one would get by a much more careful approximations of $\psi(x, y)$. Thus we can suppose that $\psi(x, y)$ is roughly equal to the number of y-tuple nonnegative integer solutions to the inequality $\sum_{j=1}^{y} \alpha_j \leq u$.

Now using lemma 3.3.8 (with N = y), $\psi(x, y)$ is approximately $\binom{[u]+y}{y}$. We now estimate $\log(\frac{\psi(x,y)}{x})$, which is the logarithm of the probability that a random integer between 1 and x is a product of primes $\leq$ y.

We know that $\log(x) = u\log(y)$, by definition of u. Now using the approximation for $\psi(x, y)$ and lemma 3.3.7, we get :

$\log(\frac{\psi(x,y)}{x}) \approx \log(\frac{([u]+y)!}{[u]!y!})$ - ulogy $\approx$ ([u]+y)log([u]+y)-([u]+y)-([u]log[u]-[u])-(ylogy-y)-ulogy. We now make some further approximations. First, we replace [u] by u. Next, we note that, because u is assumed to be much smaller than y, so, we can replace log(u+y) by logy. After cancellation we obtain $\log(\frac{\psi(x,y)}{x}) \approx$ -ulogu, i.e., $\frac{\psi(x,y)}{x} \approx u^{-u}$. □

The above proposition says that if $x \approx 10^{48}$ and y $\approx 10^6$ as above, then the probability that a random number between 1 and x is a product of primes $\leq$ y is about 1 out of 8^8.

Theorem 3.3.10. *The overall complexity of the Fermat factor base algorithm is $O(e^{c\sqrt{rlogr}})$ for some constant c, where n is an r-bit integer.*

Proof : For analysing the running time of the algorithm, we estimate the number of bit operations required to carry out the following steps :

(1) choose random numbers b_i between 1 and n and express the least positive residue of b_i^2 modulo n as a product of primes $\leq$ y if it can be so expressed, continuing until we have π(y)+1 different b_i's for which b_i^2 mod n is written as such a product.

(2) Find a set of linearly dependent rows in the corresponding $((\pi(y)+1)\times\pi(y))$-matrix of zeroes and ones to obtain a congruence of the form $b^2 \equiv c^2$ mod n.

(3) If $b \equiv \pm c$ mod n, repeat (1) and (2) with new b_i until we obtain $b^2 \equiv c^2$ mod n with $b \neq \pm c$ mod n, at which point find a nontrivial factor of n by computing gcd(b+c, n).

Step (1) (Sieving): Assuming that the b_i^2 mod n (meaning least positive residue of b^2 modulo n) are randomly distributed between 0 and 1, using proposition 3.3.9, we expect that it will take approximately u^u tries before we find a b_i such that b_i^2

mod n is a product of primes $\leq$ y, where u = logn/logy.
Choosing y $\rightarrow$: We will decide later that how to choose y so as to minimize the length of time.
* : If we choose y large enough then it would make u^u small, and so we would frequently encounter b_i such that b_i^2 mod n is a product of primes $\leq$ y. However in this case the factorization of b_i^2 mod n into a product involving all of those primes - which we would have to do π(y)+1 times - and then the row reduction of the matrix would all be very time consuming.
* : Conversely, if we choose y fairly small, then the latter tasks would be easy, but it would take us a very long time to find out any b_i's for which b_i^2 mod n is divisible only by primes $\leq$ y, because in this case u^u would be very large.
$\rightarrow$: So y should be chosen in some intermediate range, as a compromise between these two extremes.
$\rightarrow$: In order to decide how y should be chosen, we first make a very rough estimate in terms of y (and n, of course) of the number of bit operations. We then minimize this with respect to y, and find our estimate with y chosen so that time is minimized.

Suppose n is an r-bit integer and y be an s-bit integer; then u is very close to r/s. Now we can observe that to generate a random integer b_i between 1 and n, takes a fixed amount O(r) bit operations of time to generate a random bit. Next, computing b_i^2 mod n takes $O(r^2)$ bit operations. We must then divide b_i^2 mod n successively by all primes $\leq$ y which divide it evenly (and by any power of the prime that divides it evenly), and we hope that when we are done we will be left with 1. A simple way to do this would be to divide successively by 2 and by all odd integers p from 3 to y, recording as we go along what power of p divides b_i^2 mod n evenly. We can notice that if p is not prime, then it will not divide evenly, since we will have already removed from b_i^2 mod n all the factors of p. Since a division of an integer $\leq$ r bits by an integer of $\leq$ s bits takes time O(rs). Thus we see that each test of a randomly chosen b_i takes O(rsy) bit operations.
To complete step(1) requires testing approximately $u^u(\pi(y)+1)$ values of b_i, in order to find π(y)+1 values for which b_i^2 mod n is a product of primes $\leq$ y. Since we know by prime number theorem that $\pi(y) \approx y/\log y = O(y/s)$. Hence Step (1) takes $O(u^u r y^2)$ bit operations.

Step (2) (finding a linearly dependent set) : The second step to row reduce a matrix, assuming a size of approximately y^2, can be done in at most $O(y^3)$ with

Gaussian elimination, but there are faster methods available. We know that r is at most $\log(n)$, combining the relations to get the desired squares can be done $O(r^h)$, so, this step involves operations which are polynomial in y and r (such as matrix reduction and finding b and c modulo n). Thus, step (2) takes $O(y^j r^h)$ bit operations for some integers j and h. Therefore, overall complexity at the end of the Step (2) is $O(u^u ry^2 + y^j r^h)$.

Step (3) : We know (as discussed above in Note (4)) that each time we perform steps (1)-(2) there is at least a 1/2 probability of success, i.e., of finding that $b \neq c \bmod n$. More precisely, the chance of success is 50% if n is divisible by only two distinct primes, and is greater if n is divisible by more primes. Thus if we say that $1\text{-}2^{-50}$ probability of finding a non trivial factor of n, it suffices to go through the steps 50 times. Therefore taking this as good enough for all practical purposes, we have overall complexity : $O(50(u^u ry^2 + y^j r^h)) = O(r^h u^u y^j) = O(r^h u^u e^{ks}) = O(r^h (r/s)^{r/s} e^{ks})$, for suitable integers h and k.

We now find y and equivalently s, for which this estimate is minimal. Since r, the number of bits in n, is fixed, this means, we have to minimize $(r/s)^{r/s} e^{ks}$ with respect to s, or equivalently, minimize its log, which is $\frac{r}{s}\log\frac{r}{s} + ks$. Thus, we set :

$$0 = \frac{d}{ds}(\frac{r}{s}log\frac{r}{s} + ks) = -\frac{r}{s^2}(log\frac{r}{s} + 1) + k \approx -\frac{r}{s^2}log\frac{r}{s} + k$$

From the above equation we can observe that we have to choose s such a way that $ks \approx \frac{r}{s}\log\frac{r}{s}$ or in other words $(r/s)^{r/s} \approx e^{ks}$. Because k is a constant, it follows from the above approximate inequality that s^2 has the same order of magnitude as $r\log(r/s) = r(\log r - \log s)$, which means that s has order of magnitude between $\sqrt{r}$ and $\sqrt{rlogr}$. But this means that logs is approximately $\frac{1}{2}\log r$, and so, making this substitution $\log s \approx \frac{1}{2}\log r$. We transform this in above approximate inequality and we get :

$$0 \approx -\frac{r}{2s^2}logr + k \;\; or \;\; s \approx \sqrt{\frac{r}{2k}logr}$$

With this value of s, we now estimate the total time. Since $(r/s)^{r/s} \approx e^{ks}$, So, overall complexity of the algorithm simplifies to $O(e^{2ks}) = O(e^{\sqrt{2k}\sqrt{rlogr}})$. Thus replacing the constant $\sqrt{2k}$ by C, we finally obtain $O(e^{c\sqrt{rlogr}})$ bit operations required to factor an r-bit integer n. □

In the above theorem, the argument was very rough. We did not attempt to justify our simplifications or bound the error in our approximate inequalities. In addition, both our algorithm and our estimate of its running time are probabilistic.

3.4 The continued fraction method

In the last section, we saw that the factor base method of finding a nontrivial factor of a large composite integer n works best if one has a good method of finding integers b between 1 and n such that the least absolute residue b^2 mod n is a product of small primes. This is most likely occur if the absolute value of b^2 mod n is small. In this section, we describe a method for finding many b such that $|b^2 \bmod n| < 2\sqrt{n}$. This method uses "continued fractions", so, we shall start with a brief introduction to the continued fraction representation of a real number. We will describe only those features which will be needed here.

3.4.1 Continued fractions

Given a real number x, we construct its continued fraction expansion as follows :
(1) Let $a_0 = [x]$; where $[x]$ denote the greatest integer not greater than x and set $x_0 = x - a_0$
(2) let $a_1 = [\frac{1}{x_0}]$ and set $x_1 = \frac{1}{x_0} - a_1$
(3) for i > 1, let $a_i = [\frac{1}{x_{i-1}}]$, and set $x_i = \frac{1}{x_{i-1}} - a_i$
(4) when $\frac{1}{x_{i-1}}$ is an integer, then we find $x_i = 0$, and the process stops.

It is not hard to see that the process terminates if and only if x is rational (because in that case the x_i are rational numbers with decreasing denominators) or we can say that a_i become a repeating sequence if and only if x is a quadratic irrationality, i.e., of the form $x_1 + x_2\sqrt{n}$ with x_1 and x_2 rational and n not a perfect square. This is known as Lagrange's theorem.

Notation : Because of the construction of the $a_0, a_1, \cdots, a_i$, for each i we can write :

$$x = a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \cdots \frac{1}{a_i + x_i}}}$$

which is usually written in a more compact notation as follows :

$$x = a_0 + \frac{1}{a_1+}\frac{1}{a_2+}\frac{1}{a_3+}\cdots\frac{1}{a_i + x_i}$$

Suppose x is an irrational real number. If we carry out the above expansion to the ith term and then delete x_i, we obtain a rational number b_i/c_i, called the ith convergent of the continued fraction for x :

$$\frac{b_i}{c_i} = a_0 + \frac{1}{a_1+}\frac{1}{a_2+}\frac{1}{a_3+}\cdots\frac{1}{a_{i-1}+}\frac{1}{a_i}$$

Proposition 3.4.1. *In the above notation, we have :*
(a) $\frac{b_0}{c_0} = \frac{a_0}{1}$; $\frac{b_1}{c_1} = \frac{a_0a_1+1}{a_1}$; $\frac{b_i}{c_i} = \frac{a_ib_{i-1}+b_{i-2}}{a_ic_{i-1}+c_{i-2}}$ *for* $i \geq 2$.
(b) The fractions on the right in part (a) are in lowest terms, i.e., if $b_i = a_ib_{i-1} + b_{i-2}$ *and* $c_i = a_ic_{i-1} + c_{i-2}$, *then* $gcd(b_i, c_i) = 1$.
(c) b_ic_{i-1} - $b_{i-1}c_i = (-1)^{i-1}$ *for* $i \geq 1$.

Proof : (a) For proving this part, we define the sequences $\{b_i\}$ and $\{c_i\}$ by the relations in (a) and prove by induction that b_i/c_i is the ith convergent. We will prove this without assuming that the a_i are integers, i.e., we will prove that for any real numbers a_i the ratio b_i/c_i with b_i and c_i defined by the formulas in (a) is equal to $a_0 + \frac{1}{a_1+}\frac{1}{a_3+}\cdots\frac{1}{a_i}$. It is trivial to check the beginning of the induction (i = 0,1,2). We now suppose that the claim is true through the ith convergent, and we prove the claim for the (i+1)th convergent. Note that we obtain the (i+1)th convergent by replacing a_i by a_i+1/a_{i+1} in the formula that expresses the numerator and denominator of the ith convergent in terms of the (i-1)th and (i-2)th. That is, the (i+1)th convergent is :

$$\frac{(a_i + \frac{1}{a_{i+1}})b_{i-1} + b_{i-2}}{(a_i + \frac{1}{a_{i+1}})c_{i-1} + c_{i-2}} = \frac{a_{i+1}(a_ib_{i-1} + b_{i-2}) + b_{i-1}}{a_{i+1}(a_ic_{i-1} + c_{i-2}) + c_{i-1}} = \frac{a_{i+1}b_i + b_{i-1}}{a_{i+1}c_i + c_{i-1}}$$

by the induction assumption. This completes the induction and proves part (a).
(b) This part follows from part (c), because any common divisor of b_i and c_i must divide $(-1)^{i-1}$, which is ± 1.
(c) We will prove this part by induction. Let the inequality of this part hold for i. Now similar to part (a), we will show this for i+1.

$$b_{i+1}c_i - b_ic_{i+1} = (a_{i+1}b_i + b_{i-1})c_i - b_i(a_{i+1}c_i + c_{i-1}) = b_{i-1}c_i - b_ic_{i-1} = -(-1)^{i-1} = (-1)^i$$

This completes the proof of the proposition. □

Note (1) : If we divide the equation in the last part of the above proposition by $c_i c_{i-1}$, we find that $\frac{b_i}{c_i} - \frac{b_{i-1}}{c_{i-1}} = \frac{(-1)^{i-1}}{c_i c_{i-1}}$. Since c_i form a strictly increasing sequence of positive integers, this inequality shows the sequence of convergents behaves like an alternating series, i.e., it oscillates back and forth with shrinking amplitude; thus, the sequence of convergents converges to a limit.

Note (2) : The limit of the convergents is the number x which was expanded in the first place. To see that, we can notice that x can be obtained by forming the $(i+1)^{th}$ convergent with a_{i+1} replaced by i/x_i. Thus using part (a) of the above proposition (with i replaced by i+1 and a_{i+1} replaced by $1/x_i$), we have

$$x = \frac{\frac{b_i}{x_i} + b_{i-1}}{\frac{c_i}{x_i} + c_{i-1}} = \frac{b_i + x_i b_{i-1}}{c_i + x_i c_{i-1}}$$

and this is strictly between b_{i-1}/c_{i-1} and b_i/c_i. To see this, we can consider the two vectors $u = (b_i, c_i)$ and $v = (b_{i-1}, c_{i-1})$ in the plane, both in the same quadrant; and we can observe that the slope of the vector $u+x_i v$ is intermediate between the slopes of u and v. Thus, the sequence b_i/c_i oscillates around x and converges to x.

Example 3.4.2. *Let we have to expand $\sqrt{3}$ as a continued fraction, So, using the procedure which is discussed in starting of the continued fraction, we obtain :*

$$\sqrt{3} = 1 + \frac{1}{1+}\frac{1}{2+}\frac{1}{1+}\frac{1}{2+}\frac{1}{1+}\frac{1}{2+}\cdots .$$

At this point we might conjecture that a_i's alternate between 1 and 2. To prove this, let x equal to the infinite continued fraction on the right with alternating 1's and 2's. Then clearly, $x = 1+\frac{1}{1+(1/(1+x))}$, as we see by replacing x on the right by its definition as a continued fraction. Simplifying the rational expression on the right and multiplying both sides of the equation by (2+x) gives : $2x+x^2 = 3+2x$, i.e., $x = \sqrt{3}$.

Proposition 3.4.3. *Let $x > 1$ be a real number whose continued fraction expansion has convergents b_i/c_i. Then for all i : $|b_i^2 - x^2 c_i^2| < 2x$.*

Proof : *Since x is between b_i/c_i and b_{i+1}/c_{i+1}, and since the absolute value of the difference between these succesive convergents is $\frac{1}{c_i c_{i+1}}$ (by Note (1)), so $|x - \frac{b_i}{c_i}| <$*

$\frac{1}{c_i c_{i+1}}$, *we have*

$$|b_i^2 - x^2 c_i^2| = c_i^2 |x - \frac{b_i}{c_i}||x + \frac{b_i}{c_i}| < c_i^2 \frac{1}{c_i c_{i+1}}(x + (x + \frac{1}{c_i c_{i+1}}))$$

Hence,

$$|b_i^2 - x^2 c_i^2| - 2x < 2x(-1 + \frac{c_i}{c_{i+1}} + \frac{1}{2xc_{i+1}^2}) < 2x(-1 + \frac{c_i}{c_{i+1}} + \frac{1}{c_{i+1}} < 2x(-1 + \frac{c_{i+1}}{c_{i+1}}) = 0$$

This proves the proposition. □

Proposition 3.4.4. *Let n be a positive integer which is not a perfect square. Let b_i/c_i be the convergents in the continued fraction expansion of $\sqrt{n}$. Then the residue of b_i^2 modulo n which is smallest in absolute value (i.e., between -n/2 and n/2) is less than $2\sqrt{n}$.*

Proof : *Write $b_i^2 \equiv b_i^2$ - nc_i^2 mod n. Now taking $x = \sqrt{n}$ in the proposition 3.4.3, we get the result.* □

3.4.2 Idea of the continued fraction factoring method

Proposition 3.4.4 is the key to the continued fraction algorithm. It says that we can find a sequence of b_i's whose squares have small residues by taking the numerators of the convergents in the continued fraction expansion of $\sqrt{n}$. We can observe here that we do not have to find the actual convergent, only numerator b_i is needed, and that is needed only modulo n. Thus the fact that the numerator and denominator of the convergents soon become very large does not worry us. We never need to work with integers larger than n^2 (when we multiply integers modulo n).

3.4.3 Continued fraction factoring algorithm

Let n be the integer to be factored. All computations below in the algorithm will be done modulo n, i.e., products and sums of integers will be reduced modulo n to their least nonnegative residue (or least absolute residue). The algorithm :
(1) Set $b_{-1} = 1$, $b_0 = a_0 = [\sqrt{n}]$, and $x_0 = \sqrt{n}$ - a_0.
(2) Compute b_0^2 mod n.
(3) Set $a_i = [1/x_{i-1}]$ and then $x_i = 1/x_{i-1}$ - a_i for i = 1,2,$\cdots$ succesively.
(4) Set $b_i = a_i b_{i-1} + b_{i-2}$ (reduced modulo n).
(5) Compute b_i^2 mod n.

(6) In step (5), after doing for several i, look at the numbers which factor in to $\pm a$ product of small primes.
(7) Take factor base B which consists -1 and the primes which occur in more than one of the b_i^2 mod n (or which occur to an even power in just one b_i^2 mod n).
(8) List all of the numbers b_i^2 mod n which are B-numbers, along with the corresponding vectors $\overrightarrow{\epsilon_i}$ of zeroes and ones.
(9) If possible, find a subset whose vectors sum to zero.
(10) Set $b = \prod_i b_i$ mod n (working modulo n and taking the product over the subset for which $\sum_i \overrightarrow{\epsilon_i} = 0$) and $c = \prod_j p_j^{\gamma_j}$ mod n (where p_j are the elements of B (except for -1) and $\gamma_j = \frac{1}{2}\sum_i \alpha_{ij}$)
(11) If $b \neq \pm c$ mod n, then gcd(b+c, n) is a nontrivial factor of n.
(12) If $b \equiv \pm c$ mod n, then look for another subset of i such that $\sum_i \overrightarrow{\epsilon_i} = 0$. If it is not possible to find another subset of i such that $\sum_i \overrightarrow{\epsilon_i} = 0$, then continue computing more a_i, b_i, and b_i^2 mod n, enlarging factor base B if necessary.

Remark 3.4.5. *For computing $c = \prod_j p_j^{\gamma_j}$, it is efficient if for each B-number b_i^2 mod n, we record the vector $\overrightarrow{\alpha_i} = \{\cdots, \alpha_{ij}, \cdots\}_j$ rather than $\overrightarrow{\epsilon_i}$, which is simply $\overrightarrow{\alpha_i}$ reduced modulo 2.*

Example 3.4.6. *Suppose we have to factor 17873. We start with the following table :*

i	*0*	*1*	*2*	*3*	*4*	*5*
a_i	*133*	*1*	*2*	*4*	*2*	*3*
b_i	*133*	*134*	*401*	*1738*	*3877*	*13369*
b_i^2 *mod n*	*-184*	*83*	*-56*	*107*	*-64*	*161*

If we set B = {-1,2,7,23}, we have B-numbers when i = 0,2,4,5; the corresponding vectors $\overrightarrow{\alpha_i}$ are respectively {1,3,0,1}, {1,3,1,0}, {1,6,0,0} and {0,0,1,1}. The sum of the first, second and forth of these four vectors is zero modulo 2. However, if we compute $b = 133 \cdot 401 \cdot 13369 \equiv 1288$ mod 17873 and $c = 2^3 \cdot 7 \cdot 23 = 1288$, we find that $b \equiv c$ mod 17873. Thus, we must continue to look for more B-numbers with vectors that sum to zero modulo 2. Continuing the table, we have :

i	*6*	*7*	*8*
a_i	*1*	*2*	*1*
b_i	*17246*	*12115*	*11488*
b_i^2 *mod n*	*-77*	*149*	*-88*

If we enlarge B to include the prime 11, i.e., B = {-1,2,7,11,23}, then for i = 0,2,4,5,6,8 we obtain B-numbers with vectors $\overrightarrow{\alpha_i}$ as follows : {1,3,0,0,1}, {1,3,1,0,0}, {1,6,0,0,0}, {0,0,1,0,1}, {1,0,1,1,0}, {1,3,0,1,0}. We now note that the sum of the second, third, fifth and sixth of these six vectors is zero modulo 2. This leads to b = 7272, c = 4928, and we finally find a nontrivial factor gcd(7272+4928, 17873) = 61. Thus we obtain : 17873 = 61·293.

3.4.4 Running time of the algorithm

In this algorithm, we are doing all the steps similar to Fermat factor base method except the way of choosing random number b_i. Therefore similar to Fermat factor base, we can see that the overall complexity of this algorithm is $O(e^{c\sqrt{rlogr}})$ for some constant c, where n is an r-bit integer.

3.5 The Quadratic Sieve Method

The quadratic Sieve method for factoring large integers, developed by Pomerance in the early 1980's, for a long time was more successful than any other method of factoring integers n of general type which have no prime factor of order of magnitude significantly less than $\sqrt{n}$.

3.5.1 Idea of the algorithm

This algorithm is a variant of the factor base approach which is discussed in section 3.3. In this method, we choose our factor base B, the set of all primes $p \leq P$ (where P is some bound to be chosen in some optimal way) such that n is a quadratic residue mod p, i.e., $(\frac{n}{p}) = 1$ for p odd, and p = 2 is always included in B. The set of integers S in which we look for B-numbers (Recall : a B-number is an integer divisible only by primes in B) will be the same set that we used in Fermat factorization (see section 3.3), namely :

$$S = \{t^2 - n | [\sqrt{n}] + 1 \leq t \leq [\sqrt{n}] + A\}$$

for some suitably chosen bound A.

The main idea of the method is that, instead of taking each $s \in S$ one by one and dividing it by the primes $p \in B$ to see if it is a B-number, we take each $p \in B$ one by one and examine divisibility by p (and powers of p) simultaneously for all

of the $s \in S$. The word "Sieve" refers to this idea. Here we can recall the "Sieve of Eratosthenes", which one can use to make a list of all primes $p \leq A$. For example, to list the primes $\leq$ 1000 one takes the list of all integers $\leq$ 1000 and then for each p = 2,3,5,7,11,13,17,19,23,29,31 = $[\sqrt{1000}]$, one discards all multiples of p greater than p, one "lets them fall through a Sieve which has holes spaced a distance p apart", after which the numbers that remain are the primes.

3.5.2 Algorithm

Suppose we have an odd composite integer n. Then this method for factoring n follows :

(1) Choose bounds P and A, both of order of magnitude roughly $e^{\sqrt{lognloglogn}}$. Generally, A should be larger than P, but not larger than a fairly small power of P, e.g., $P < A < P^2$. (The function $exp(\sqrt{lognloglogn})$, which is denoted by $L(n)$, has order of magnitude intermediate between polynomial in logn and polynomial in n. If $n \approx 10^6$, then $L(n) \approx 400$.)

(2) For $t = [\sqrt{n}]+1, [\sqrt{n}]+2, \cdots, [\sqrt{n}]+A$, make a column listing the integers t^2-n.

(3) For each odd prime $p \leq P$, first check that $(\frac{n}{p}) = 1$; if not, then throw that p out of the factor base.

(4) Assuing that p is an odd prime such that n is a quadratic residue mod p (we will take the case $p = 2$ separately), solve the equation $t^2 \equiv n \pmod{p^\beta}$ for $\beta = 1,2,\cdots$. Take increasing values of β until find that there is no solution t which is congruent modulo p^β to any integer in the range $[\sqrt{n}]+1 \leq t \leq [\sqrt{n}]+A$. Let β be the largest integer such that there is some t in the range for which $t^2 \equiv n \pmod{p^\beta}$. Let t_1 and t_2 be two solutions of $t^2 \equiv n \pmod{p^\beta}$ with $t_2 \equiv -t_1 \pmod{p^\beta}$ (t_1 and t_2 are not necessarily in the range from $[\sqrt{n}]+1$ to $[\sqrt{n}]+A$).

(5) Still with the same value of p, run down the list of $t^2 - n$ from step 2. In a column under p put a 1 next to all values of t^2-n for which t differs from t_1 by a multiple of p, change the 1 to 2 next to all values of t^2-n for which t differs from t_1 by a multiple of p^3, and so on until p^β. Then do the same with t_1 replaced by t_2. The largest integer that appears in this column will be β.

(6) In step (5), as we go through the procedure, each time we put down 1 or change 1 to 2, 2 to 3, etc., divide the corresponding $t^2 - n$ by p and keep a record of what's left.

(7) In the column $p = 2$, if $n \neq 1 \bmod 8$, then simply put a 1 next to the $t^2 - n$ fot t odd and divide the corresponding t^2-n by 2. If $n \equiv 1 \bmod 8$, then solve the equation $t^2 \equiv n \pmod{2^\beta}$ and proceed exactly as in the case of odd p (except that there will be

4 different solutions, t_1, t_2, t_3, t_4 modulo 2^β if $\beta \geq 3$).
(8) When we finish with all primes $\leq P$, throw out all of the t^2-n except for those which have become 1 after division by all the powers of $p \leq P$.
(9) Now we have a table in which the column labeled b_i will have the values of t, $[\sqrt{n}]+1 \leq t \leq [\sqrt{n}]+A$, for which t^2-n is a B-number, and the other columns correspond to all values of $p \leq P$ for which n is a quadratic residue.
(10) The rest of the procedure is the same as in the Fermat factor base algorithm from step (5).

Example 3.5.1. *Suppose we have to factor n = 1042387. Take the bounds P = 50 and A = 500. Here $[\sqrt{n}] = 1020$. Our factor base consists of the 8 primes {2,3,11,17,19,23,43,47} for which 1042387 is a quadratic residue. Since $n \neq 1 \pmod 8$, the column corresponding to p = 2 alternates between 1 and 0, with a 1 beside all odd t, $1021 \leq t \leq 1520$.*

We now describe in detail how to form the column under p = 3. We want a solution $t_1 = t_{1,0} + t_{1,1}\cdot 3 + t_{1,2}\cdot 3^2 + t_{1,3}\cdot 3^3 + \cdots + t_{1,\beta-1}\cdot 3^{\beta-1}$ to $t_1^2 \equiv 1042387 \pmod{3^\beta}$, where $t_{1,j} \in \{0,1,2\}$ (for the other solution we can take $3^\beta - t_1$). We can obviously take $t_{1,0} = 1$ (since solution of $t_{1,0}^2 \equiv 1042387 \pmod 3$ is $t_{1,0} = 1$ (or for all the numbers 1042387 is a quadratic residue) and for each of our 8 primes the first step - solving $t_1^2 \equiv 1042387 \pmod p$ - can be done quickly by trial). Next, we work modulo 9 : $(1+3t_{1,1})^2 \equiv 1042387 \equiv 7 \pmod 9$, i.e., $6t_{1,1} \equiv 6 \pmod 9$, i.e., $2t_{1,1} \equiv 2 \pmod 3$, so $t_{1,1} = 1$. Next modulo 27 : $(1+3+9t_{1,2})^2 \equiv 1042387 \equiv 25 \pmod{27}$, i.e., $16+18t_{1,2} \equiv 25 \pmod{27}$, i.e., $2t_{1,2} \equiv 1 \pmod 3$, $t_{1,2} = 2$. Then modulo 81 : $(1+3+18+27t_{1,3})^2 \equiv 1042387 \equiv 79 \pmod{81}$, which leads to $t_{1,3} = 0$. Continuing until 3^7, we find the solution : $(210211)_3 \pmod{3^7}$, and $t_2 \equiv (2012012)_3 \pmod{3^7}$. However there is no t between 1021 and 1520 which is $\equiv t_1$ or t_2 modulo 3^7. Thus, we have $\beta = 6$, and we can take $t_1 = (210211)_3 = 589 \equiv 1318 \pmod{3^6}$ and $t_2 = 3^6 - t_1 = 140 \equiv 1112 \pmod{3^5}$ (note that there is no number in the range from 1021 to 1520 which is $\equiv t_2 \pmod{3^6}$).

We now construct our "Sieve" for the prime 3 as follows. Starting from 1318, we take jumps of 3 down until we reach 1021 and up until we reach 1519, each time putting a 1 in the column, dividing the corresponding t^2-n by 3, and recording the result of the division. (Actually, for t odd, the number we divide by 3 is half of t^2-n, since we already divided t^2-n by 2 when we formed the column of alternating 0's and 1's under 2.) Now, we do the same with jumps of 9, each time changing the 1 to 2

in the column under 3, dividing the quotient of t^2-n by another 3, and recording the result. We go through the analogous procedure with jumps of 27, 81, 243 and 729 (there is no jump possible for 729 - we merely change the 5 to 6 next to 1318 and divide the quotient of 1318^2-1042387 by another 3). Finally, we go through the same steps with $t_2 = 1112$ instead of $t_1 = 1318$, this time stopping with jumps of 243.

After going through this procedure for the remaining 6 primes in our factor base, we have a 500×8 array of exponents, each row corresponding to a value of t between 1021 and 1520. Now we throw out all rows for which t^2-n is a B-number. After throw out, we are left with the following table (blank spaces denote zero exponents) :

t	t^2-*n*	*2*	*3*	*11*	*17*	*19*	*23*	*43*	*47*
1021	*54*	*1*	*3*	-	-	-	-	-	-
1027	*12342*	*1*	*1*	*2*	*1*	-	-	-	-
1030	*18513*	-	*2*	*2*	*1*	-	-	-	-
1061	*83334*	*1*	*1*	-	*1*	*1*	-	*1*	-
1112	*194157*	-	*5*	-	*1*	-	-	-	*1*
1129	*232254*	*1*	*3*	*1*	*1*	-	*1*	-	-
1148	*275517*	-	*2*	*3*	-	-	*1*	-	-
1175	*338238*	*1*	*2*	-	-	*1*	*1*	*1*	-
1217	*438702*	*1*	*1*	*1*	*2*	-	*1*	-	-
1390	*889713*	-	*2*	*2*	-	*1*	-	*1*	-
1520	*1268013*	-	*1*	-	*1*	-	*2*	-	*1*

Now we look for relations modulo 2 between the rows of the matrix. That is, moving down from the first row, we look for a subset of the rows which sums to an even number in each column. The first such subset we find here is the first three rows, the sum of which is twice the row 1 3 2 1 - - - - . Thus, we obtain the congruence $(1021 \cdot 1027 \cdot 1030)^2 \equiv (2 \cdot 3^3 \cdot 11^2 \cdot 17)^2$ (mod 1042387). But in this case we get only trivial factorization because the two numbers being squared in the above congruence are both 111078 (mod 1042387). So, finally, when we are about to give up - we start over again with a larger A - we notice that the last row (corresponding to our last value of t) is dependent on the earlier rows. More precisely, it is equal modulo 2 to the fifth row. This gives us $(1112 \cdot 1520)^2 \cdot (3^3 \cdot 17 \cdot 23 \cdot 47)^2$ (mod 1042387), i.e., $647853^2 \equiv 496179^2$ (mod 1042387), and we obtain the nontrivial factor gcd(647853-496179, 1042387) = 1487.

3.5.3 Running time Analysis

There are two major steps that the running time is effected by :
(1) Searching for squares with residues that are B-smooth modulo n.
(2) Finding a linear dependent set.
Running time of both steps, we have discussed in Fermat factor base method, takes approximately $O(e^{C\sqrt{rlogr}})$*, where n is an r-bit integer and C is constant.*

3.6 The Number Field Sieve

There are two types of number field Sieve algorithm - special number field Sieve (SNFS) and general number field Sieve (GNFS), which have the smallest time complexity compared to all other factoring algorithms. The SNFS works on a special type of composites, namely integers of the form : r^e*-s, for small integers r,s and integer e and the GNFS works on all types of composites. The difference between SNFS and GNFS is in the polynomial selection part of the algorithm, where the special numbers which SNFS can be applied to make a special class of polynomials especially attractive and the work in the square root step is also more complex for the GNFS. We will describe these algorithms briefly.*

3.6.1 Special Number Field Sieve (SNFS)

Before we describe the special number field Sieve, we will go through some mathematical results that the special number field Sieve rely upon.

Let $f(x) = x^d - t$ *be an irreducible polynomial over* $\mathbb{Q}$*,* $t \in \mathbb{Z}$*. Let* α *be such that* $f(\alpha) = 0$*.* $K = \mathbb{Q}(\alpha)$ *is then an algebraic extension, a number field, and* $\mathbb{Z}[\alpha]$ *is a subring of K.*

Note : *(1) The norm of an ideal* $I \in \mathbb{Z}[\alpha]$*,* $I \neq (0)$ *is the positive integer* $\#(\mathbb{Z}[\alpha]/I)$*. (2) From the above note, it follows that this is finite, and that the norm of an ideal is equal to the norm of the element generating the ideal.*
(3). An ideal I is prime ideal if for all $xy \in I \Leftrightarrow x \in I \cup y \in I$*.*

Definition 3.6.1. *A first degree prime ideal is a prime ideal I with norm p, where p is prime.*

Let I be a first degree prime ideal of norm p, then it is possible to construct a ring homomorphism

$$\theta : \mathbb{Z}[\alpha]/I \longrightarrow \mathbb{Z}/p\mathbb{Z}$$

$$\alpha \longrightarrow c \; modp$$

where c is a root of f(x) in $\mathbb{Z}/p\mathbb{Z}$, equivalently $f(c) \equiv 0 \pmod p$

The pair (p, c mod p) corresponds to the first degree prime ideal I, in fact the first degree prime ideals and the pairs (p, c mod p) are in bijective correspondence with each other. This follows from the two lemmas.

Lemma 3.6.2. *Let p be a prime integer. If $f(c) \equiv 0 \pmod p \Longrightarrow I = \langle p, c\text{-}\alpha \rangle$ is a first degree prime ideal.*

Proof : *Let ϕ be a mapping defined as follows :*

$$\phi : \mathbb{Z}[\alpha]/I \longrightarrow \mathbb{Z}/p\mathbb{Z}$$

$$\alpha \longrightarrow c$$

$$z \longrightarrow z \; (modp)$$

for $\alpha \in \mathbb{Z}[\alpha]$, $z \in \mathbb{Z}$ and $c \in \mathbb{Z}/p\mathbb{Z}$ such that $f(c) \equiv 0 \pmod p$.
For proving lemma, it is enough to prove that ϕ is an isomorphism.
Firstly for ring homomorphism, let $x, y \in \mathbb{Z}[\alpha]/I$. Then we see that $\phi(x + y) = (x + y) + I = (x + I) + (y+I) = \phi(x)+\phi(y)$ and $\phi(xy) = xy+I = (x+I)(y+I) = \phi(x)\phi(y)$. So, we have ϕ is a ring homomorphism.
Now let $y \in \mathbb{Z}/p\mathbb{Z}$, we want to find an $x \in \mathbb{Z}[\alpha]/I$ such that $\phi(x) = y$. We know that $1 \in \mathbb{Z}[\alpha]/I$ and since ϕ is an homomorphism $\phi(1) = 1$. Then $\phi(y) = \phi(1 \cdot y) = y\phi(1) = y$, so, $x = y$ will do the job and hence, ϕ is on to.
Again let $x \in \mathbb{Z}[\alpha]/I$ but assume $\phi(x)=0$. Since $x \in \mathbb{Z}[\alpha]/I$, it is of the form $a_0 + a_1\alpha + \cdots + a_n\alpha^n$ where $a_i \in \mathbb{Z}$. If we then take an element $x' = a_0 + a_1 c + \cdots + a_n c^n \in \mathbb{Z}[\alpha]/I$, add and subtract it from x, we get $x = x - x' + x' = k_1 p + k_2(c\text{-}\alpha)$ for $k_1, k_2 \in \mathbb{Z}[\alpha]$ is an element of I, and we can conclude that ϕ is one-one.

Lemma 3.6.3. *Let p be a prime integer. If $I = \langle p, c\text{-}\alpha \rangle$ and $I' = \langle p, c'\text{-}\alpha \rangle \Rightarrow$ either $I \neq I'$ (and $c \neq c' \pmod p$) or $I \equiv I'$ (and $c \equiv c' \pmod p$).*

Proof : *Consider $(c\text{-}\alpha)\text{-}(c'\text{-}\alpha) = c\text{-}c'$ has two options, either $c\text{-}c'=0$ or $c\text{-}c' \neq 0$. First case $c\text{-}c' = 0$ gives $c = c'$ and then of course $I = I'$ and $c \equiv c' \pmod p$.*

Second case c-c' $\neq$ 0 gives c $\neq$ c'. Then we can evaluate gcd(c-c', p). Since p is prime this has to be equal to 1 or p.
First let gcd(c-c', p) = 1 and assume I = I'. Then p, c-α, c'-α are all in the same ideal which also means 1 is in the ideal. This is a contradiction since I would then be the whole ring. We can then conclude I $\neq$ I' and c $\neq$ c' (mod p) when c $\neq$ c' and gcd(c-c', p) = 1.
Now assume gcd(c-c', p) = p, then c-c' = (c-α)-(c'-α) = kp, k $\in \mathbb{Z}$. This gives c-α = c'-α+kp, and since c'-α+kp $\in$ I', so, c-α is also in I' and I $\subseteq$ I'. Equivalently c'-α = c-α+kp, and since c-α+kp $\in$ I, c'-α $\in$ I and I' $\in$ I. We can then conclude that I = I' and c $\equiv$ c' (mod p) if c-c' $\neq$ 0 and gcd (c-c', p) = p.

From the above two lemma's, it is clear that the pairs (p, c mod p) are in bijective correspondence with the first degree prime ideals I, and I is generated by the elements (p, c-α).

We can also use the map ϕ to check whether a given element in $\mathbb{Z}[\alpha]$ is contained in a first degree prime ideal $I = \langle p, c - \alpha \rangle$. This is because

$$\sum_{i=0}^{d-1} a_i \alpha^i \in I \Longleftrightarrow \sum_{i=0}^{d-1} a_i c^i \equiv 0 (mod p) \tag{3.1}$$

It should now be clear that an element $\Pi_p = \sum_{i=0}^{d-1} a_i c^i$, $a_i \in \mathbb{Z}$ of $\mathbb{Z}[\alpha]$ generates a first degree prime ideal corresponding to a pair (p, c mod p) if and only if N(Π_p) = $\pm$p and $\sum_{i=0}^{d-1} a_i c^i \equiv$ 0 mod p.

- *For describing the factorization of a+bα in $\mathbb{Z}[\alpha]$, we have the following lemma :*

Lemma 3.6.4. *Let a,b $\in \mathbb{Z}$, gcd(a, b) = 1. Then all prime ideals P that occur in a+bα are the first degree prime ideals.*

Proof : *Assume that P occurs in a+bα, and let P be the kernel of a ring homomorphism $\psi : \mathbb{Z}[\alpha] \longrightarrow \mathbb{F}$, where $\mathbb{F}$ is a finite field and suppose that the characteristic of $\mathbb{F}$ is p, such that the field $\mathbb{F}_P$ is a subfield of $\mathbb{F}$.*

Since (a+bα) $\in$ P $\Rightarrow$ ψ(a+bα) = 0 which again gives ψ(a) = -ψ(b)ψ(α).
Now it is easy to see that since both a, b $\in \mathbb{Z} \Rightarrow \psi(a) \in \mathbb{F}_p$ and $\psi(b) \in \mathbb{F}_p$.
Suppose that $\psi(b) = 0$, that means $\psi(a) = 0$ also, which means p|a and p|b, this implies that p|gcd(a, b), which is a contradiction since gcd(a, b) = 1. So, $\psi(b) \neq 0$.

We can conclude that the element $\psi(\alpha) = \frac{-\psi(a)}{\psi(b)} \in \mathbb{F}_p$.
This shows that the ring homomorphism ψ *maps all elements from* $\mathbb{Z}[\alpha]$ *to* $\mathbb{F}_p$, *and* p *is the kernel of* ψ *which is by definition means it is a first degree prime ideal.*

3.6.2 SNFS Algorithm :

Input : Composite number $n = r^e$ - s, *which is not a power of a prime.*
Output : a nontrivial factor p *of* n.

Step (1) *: (Polynomial Selection) : Find an irreducible polynomial f(x) over* $\mathbb{Q}$ *with root m, i.e. f(m)* $\equiv$ *0 (mod n), where f(x)* $\in \mathbb{Z}[x]$.
Method : Decide the degree d of the polynomial, let k be the least positive integer such that $kd \geq e$. *Then let* $t = s \cdot r^{k \cdot d - e}$. *Now f(x) =* x^d - t, *and* $m = r^k$ *satisfies f(m)* $\equiv$ *0 (mod n).*
Step (2) *: Find an element* α *such that f(*α*) = 0 and set up an homomorphism* ϕ *:*

$$\phi : \mathbb{Z}[\alpha] \longrightarrow \mathbb{Z}/n\mathbb{Z}$$

$$\alpha \longrightarrow m \ (modn)$$

Step (3) *: (Factor base) Choose the factor bound* B_1 *such that the factor base* $B^{'}$ *consists of primes less than* B_1. *Now, decide a second bound* B_2 *such that the factor base* $B^{"}$ *consists of the primes less than* B_2 *that together with an integer c correspond to a first degree ideal in* $\mathbb{Z}[\alpha]$ *as proved by lemma 3.6.2 and lemma 3.6.3.*
{ *Explanation : The factor base consists of three parts, the first part is all the prime numbers up to some limit. The second and third part of the factor base is to consist of generators for all the first degree prime ideals that have norm less than some chosen bound, and a set of generators for the group of units of* $\mathbb{Z}[\alpha]$. *Since the subring* $\mathbb{Z}[\alpha]$ *is assumed to be a principal ideal domain we know there exists a generator for all ideals in* $\mathbb{Z}[\alpha]$.
The first part of the factor base is practically the same as the factor base quadratic Sieve uses, set a bound B_1 *and we want to find* B_1 - *smooth integers (a+bm).*

$$(a + bm) = \prod_{p_j \leq B_1} p_j^{\epsilon(p_j)}$$

where $\epsilon(p_j) \in \mathbb{Z}_{\geq 0}$ *is the corresponding exponent of each* $p_j \leq B_1$.
The second and third part works just as the first part in $\mathbb{Z}[\alpha]$, *it factorizes in the*

extension. It does this by means of the first degree prime ideals that occur in the ideal generated by the element. Call the set of the generators for the first degree prime ideals of norm $\leq B_2$, G, and the set of generators for the units U. Then we will look for elements that factors by G and U, that is elements of the form :

$$a + b\alpha = \prod_{u_i \in U} u_i^{\epsilon(u_i)} \prod_{g_i \in G} g_i^{\epsilon(g_i)}$$

For generators of the first degree ideal, firstly choose some bound B_2 and then the generators will be presented with the corresponding pairs (p, c mod p), where f(c) $\equiv$ 0 (mod p). And as we have already shown after the lemma 3.6.3 that an element $\Pi_p = \sum_{i=0}^{d-1} a_i \alpha^i$, $a_i \in \mathbb{Z}$ of $\mathbb{Z}[\alpha]$ generates a first degree prime ideal corresponding to a pair (p, c mod p) if and only if N(Π_p) = $\pm p$ and $\sum_{i=0}^{d-1} a_i c^i \equiv 0$ mod p. Thus we find pair (p, c mod p) and corresponding generator of the first degree prime ideal. }

Step (4) *: (Sieving) In the sieving step, we want to find integers of the form a+bm and a+bα, a,b $\in \mathbb{Z}$, α and m roots of f in $\mathbb{Z}[\alpha]$ and $\mathbb{Z}/n\mathbb{Z}$. The pairs (a, b) which we want in this step, need to satisfy three conditions to be smooth.*
(I) gcd(a, b) = 1
(II) |a+bm| is B_1-smooth.
(III) a+bα is B_2-smooth.

The second condition ensures that (a+bm) is B_1-smooth in the same way as the factor base in quadratic Sieve does.
The first and third condition will give us elements that has only generators of first degree prime ideals with norm less than B_2 as factors (and perhaps a unit). In the same way as we search for smooth integers, we search for smooth elements, that is completely factored by U and G.
Condition one ensures that all ideals containing a+bα are first degree prime ideals, as proved by lemma 3.6.4.
We do not have to use the generators of each first degree prime ideal to check if the elements is B_2-smooth. The idea is as follows, due to equation (3.1) we know exactly when a+bα is contained in the first degree prime ideal corresponding to (p, c (mod p)). Since the extension is assumed to be a principal ideal domain and since the norm is multiplicative this implies that the prime ideal factorization of $\langle a+b\alpha \rangle$ corresponds to the factorization of the norm of $\langle a+b\alpha \rangle$. That is each first degree prime ideal that is a factor of $\langle a+b\alpha \rangle$ corresponds to a factor of the norm. So, one can simply check

the factorization of the norm N(a+bα) = a^d-t(-b)^d to see if the element a+bα factors in the factor base. This means that second and third part of our factor base (U and G) will during the sieving be replaced by all primes less than a limit B_2 that together with an integer c have a corresponding first degree prime ideal in $\mathbb{Z}$. This means in the context of condition 3 that an element a+bα is B_2-smooth if the norm N(a+bα) only has prime factors (norm of generators or p corresponding to (p, c mod p)) less than B_2.

The purpose of the sieving step is to collect as many relations or pairs (a,b) as possible (at least one larger than the elements in all the bases combined).

Step (5) *(Linear Algebra) Once enough relations have been found it is straight forward to put the exponent vectors in to a matrix (modulo 2) and start row reducing to find a zero row, that is a linear dependent set of vectors.*
Step (6) *(Square roots) When all the problems with finding the factor base and, the right combination of relations during the sieving and linear algebra step has been overcome one is left with a set S with the relations that combined are squares both in $\mathbb{Z}$ and $\mathbb{Z}[\alpha]$.*

$$x^2 = \prod_{(a,b)\in S} (a_i + b_i m) \in \mathbb{Z}$$

$$y^2 = \prod_{(a,b)\in S} (a_i + b_i \alpha) \in \mathbb{Z}[\alpha]$$

The square root of x is straight forward to compute. We know all exponents of each relation in the product and it is therefore not even necessary to compute x^2. We just use the prime factorization and compute x straight away.
As for the product of the elements in the number field it is much the same thing. We represent each relation a+bα ∈ S by it factors in G and U. This leaves us with a factorization where each first degree prime ideal is represented an even number of times. The square root can be represented as

$$y = \prod_{u_i\in U} u_i^{\frac{1}{2}e(u_i)} \prod_{\Pi_p\in G} \Pi^{\frac{1}{2}e(\Pi_p)}$$

Since a+bm and a+bα have the same image under φ in $\mathbb{Z}/n\mathbb{Z}$, so, applying φ gives two integers, that squared are congruent modulo n.

$$x^2 \equiv \phi(x^2) \equiv \phi(x)^2 \equiv \prod_{(a,b)\in S} \phi(a_i + b_i m) \equiv \prod_{(a,b)\in S} \phi(a_i + b_i \alpha) \equiv \phi(y)^2 \ (mod\, n)$$

Then find gcd($\phi(x) \pm \phi(y)$, n), hopefully gives a nontrivial factor of n. Otherwise choose a different set of linear dependent vectors or increase bound B_1, B_2 for find out more smooth pairs (a, b).

Note : *(1) In choosing factor base, instead of two factor bases, there is a practical way to let $B^{'}$ and $B^{"}$ be equal (remember to check that the p's have corresponding first degree prime ideals). In this way it is enough to check if $|(a+bm)N(a+b\alpha)|$ is factored by the factor base. This does not change the algorithm, as one can freely choose the primes to put in the factor base.*
(2) The technique described in the Step (4) using the norm $a+b\alpha$ to check for smoothness is not bulletproof. The problem is that for each p there can be several corresponding c's. Still there is one first degree prime ideal for each pair (p, c (mod p)) but the norm does not distinguish between the different ideals as they have the same norm p. As an example of this consider the polynomial $f = x^2+1$ ($\alpha = i$), now the prime $p = 13$ has two corresponding c's, that is the pairs (13, 5 (mod 13)), (13, 8 (mod 13)). Consider the relations (a, b) : (2, 3), (3, 2)

$$N(2 + 3i) = 13$$

$$N(3 + 2i) = 13$$

Multiplying the norm of these together gives a square $13 \cdot 13 = 169$ but the element (2+3i)(3+2i) we get by multiplying the generators for the first degree prime ideals is not a square in $\mathbb{Z}[\alpha]$.
The reason for this is of course that the relation (2, 3) with the factor 13 correspond to the pair (13, 8 (mod 13)), but the relation (3, 2) with the factor 13 correspond to the pair (13, 5 (mod 13)) (Here both elements are generators for two different first degree prime ideals of norm 13). This means no square of an element when they are multiplied together, and we can't use these relations alone together.
The easiest way to get around this is of course to not use the primes that have several corresponding c's in the factor base.
(3) Here we will describe a sieving technique to perform the sieving in practice. This technique uses two separate sieves, and combines their results to use the relations which both sieves find it likely to be smooth.
First decide what range of a's to search from, [a_{min}, a_{max}] and a start value for b, in practice there is no real need for a maximum b value as one can just continue until

the desired number of relations is achieved.
Start the first Sieve by fixing the value of b, and search for a values that are -bm (mod p), for all p's in factor base B'. This gives us a's that have a reasonable chance of being B_1-smooth.
In practice one will have an array with a_{max}-a_{min}+1 columns for each b, and every time a number a is -bm (mod p) one adds log(p) to the a's place in the array. Then after sieving through all p's, the a's that have a value on its place in the array that are close to log(a+bm) will be the most likely candidate to be B_1-smooth.
In the second Sieve, fix the value for b again and start looking for a values that are -bc (mod p), this is to find the pairs (a, b) in which first degree prime ideals occur in the ideal generated by a+bα. And a+bα is contained in an ideal if and only if a $\equiv$ -bc (mod p). In the same way as the first Sieve one would arrange an array and add log(p) to a's location every time it is -bc (mod p), and when we are finished the locations that have a value close to logN(a+bα) are the most likely candidates to be B_2-smooth.
After the two sieving parts are done combine the candidates that both Sieves find it likely to be smooth, check them for gcd and do trial division to find the integers that are completely factored by the factor bases. If after the first Sieve, the number of relations that most likely are B_1-smooth aren't that high it can be preferable to just use trial division to check if they also are B_2-smooth, but in a realistic scenario that number will be considerable so it will be more efficient to apply the second Sieve right away. The two Sieves can then be executed simultaneously.

Example 3.6.5. *Suppose we have to factor n = 260101 = 510^2+1. We will factor this n stepwise :*
Step (1) and (2) : we chose the polynomial to have degree 2, that is d = 2. n has r = 510, e = 2, s = -1 which gives the values k = 1, t = -1, f(x) = x^2 + 1, and m = 510. A root α of f(x) is then the complex number $\sqrt{-1}$ = i, the number field K is $\mathbb{Q}$[i], where the subring $\mathbb{Z}$[i] is the ring of integers of $\mathbb{Q}$[i]. Notice $\mathbb{Z}$[i] is in fact a principal ideal domain. The norm of an element a+bi $\in$ $\mathbb{Z}$[i] is the positive integer a^2 + b^2. We set up the homomorphism ϕ

$$\phi : \mathbb{Z}[i] \longrightarrow \mathbb{Z}/260101\mathbb{Z}$$

$$i \longrightarrow 510$$

Step (3) : Now chose the limit B_1 = 40, so the rational factor base B' = { 2, 3, 5, 7,

11, 13, 17, 19, 23, 29, 31, 37}

For the ideals we chose the first degree prime ideals with norm less than $B_2 = 55$, so the algebraic factor base :

(p, c mod p)	*Generator*	*(p, c mod p)*	*Generator*
(2, 1 mod 2)	*(1+i)*	*(29, 12 mod 29)*	*(5+2i)*
(5, 2 mod 5)	*(1+2i)*	*(37, 6 mod 37)*	*(1+6i)*
(13, 5 mod 13)	*(3+2i)*	*(41, 9 mod 41)*	*(5+4i)*
(17, 4 mod 17)	*(1+4i)*	*(53, 23 mod 53)*	*(7+2i)*

The units of $\mathbb{Z}[i]$ are the set {i, -i, 1, -1}, a generator for this set is i.

Step (4) : Now, we Sieved with the values as follows, $-a_{min} = a_{max} = 200$, and $b = 1$ up to $b = 54$, when we had a total of 23 smooth pairs (a, b), which is just enough to be sure a square as the number of primes in the factor base B' is 12, G has 8 elements and U has 2.

Step (5) : After doing linear algebra step, the smooth pairs of (a, b) we ended up with are : S = {(34, 19), (-70, 1), (-4, 1), (-2, 5), (3, 1), (-5, 7), (3, 2), (-59, 2), (-102, 23)}.

or

After doing Gaussian elimination on the exponent vectors, the following set S of pairs where found to be square when multiplied together :

(a, b)	*a+bm*	*factors*	*N(a+bi)*	*factors*	*ideal factorization*
(34, 19)	*9724*	$2^2 \cdot 11 \cdot 13 \cdot 17$	*1517*	*37·41*	*(-1)(i)(1+6i)(5+4i)*
(-70, 1)	*440*	$2^3 \cdot 5 \cdot 11$	*4901*	$13^2 \cdot 29$	$(i)(3+2i)^2(5+2i)$
(-4, 1)	*506*	*2·11·23*	*17*	*17*	*(i)(1+4i)*
(-2, 5)	*2548*	$2^2 \cdot 7^2 \cdot 13$	*29*	*29*	*(i)(5+2i)*
(3, 1)	*513*	$3^3 \cdot 19$	*10*	*2·5*	*(-1)(i)(1+i)(1+2i)*
(-5, 7)	*3565*	*5·23·31*	*74*	*2· 37*	*(1+i)(1+6i)*
(3, 2)	*1023*	*3·11·31*	*13*	*13*	*(3+2i)*
(-59, 2)	*961*	31^2	*3485*	*5·17·41*	*(1+2i)(1+4i)(5+4i)*
(-102, 23)	*11628*	$2^2 \cdot 3^2 \cdot 17 \cdot 19$	*10933*	$13 \cdot 29^2$	$(i)(3+2i)(5+2i)^2$

Step(6) : This set S gives the two products in $\mathbb{Z}$ and $\mathbb{Z}[\alpha]$ for $(a_i, b_i) \in S$:

$$\prod_{i=1}^{9}(a_i+b_i m) = (2^5 \cdot 3^3 \cdot 5 \cdot 7 \cdot 11^2 \cdot 13 \cdot 17 \cdot 19 \cdot 23 \cdot 31^2)^2 = 1153263403915814430522226944 00$$

$$\prod_{i=1}^{9}(a_i + b_i\alpha) = (-i^3(1+i)(1+2i)(3+2i)^2(1+4i)(5+2i)^2(1+6i)(5+4i)^2)^2$$

$$= (12028960768 - 34623604674i)$$

$$Or$$

$$\prod_{i=1}^{9}(a_i + b_i m) = 1153263403915814430522222694400 = (339597320942880)^2$$

$$\prod_{i=1}^{9}(a_i + b_i\alpha) = (12028960768 - 34623604674i) = (-156017 + 110961i)^2$$

Applying ϕ upon both squares will give us a square relation in $\mathbb{Z}/n\mathbb{Z}$:

$$\phi(339597320942880)^2 = (339597320942880)(mod\, 260101))^2 = 151328^2$$

$$\phi(156017 - 110961i)^2 = ((-156017 + 110961 \cdot 510)(mod\, 260101))^2 = (-7824)^2$$

Thus we can see :

$$(151328)^2 \equiv (-7824)^2(mod\, 260101)$$

This gives us the results as we wanted and by using the Euclidean algorithm to find the greatest common divisor, we achieve two factors of 260101 :

$$gcd(260101, 151328 - 7824) = 8924$$

$$gcd(260101, 151328 + 7824) = 29$$

and we see : 8969·29 = 260101.

Remark 3.6.6. *The special number field Sieve is faster than quadratic Sieve method and is the asymptotically fastest algorithm for integer factorization known today. But the special number field Sieve works only for integers of the special form $n = r^e$ - s. The quadratic Sieve works for all integers. The running time estimates however are not deterministic for neither of the algorithms. This is mostly due to the difficulty of computing the probability of finding smooth integers in a given interval accurately. Currently, the largest integer factored by the special number field Sieve is 2^{1039}-1, a 1039 bit composite integer, that is 313 digits.*

Remark 3.6.7. *The interesting thing is that each algorithm has its own intervals where it is fastest, but since the 'fastest/best' algorithm most often refers to which can*

factor the largest integers in a reasonable amount of time. The special number field Sieve is considered the best at the current moment.

The special number field Sieve is the asymptotically fastest algorithm when the number n to factor is assumed as $n \longrightarrow \infty$. But there are many technicalities and aspects of the algorithm that do not make it practical for the factoring of a random integer. The most obvious is of course that the special number field Sieve only applies to integers of the form $n = r^e$ - s. There has been generalization, the general number field Sieve and it is described in the next subsection.

3.6.3 General number field Sieve (GNFS)

This is the generalization of the SNFS. Before describing the algorithm, we will discuss some mathematical background.

Mathematical Background :

Fields and Roots of Irreducible Polynomials *Suppose a monic, irreducible polynomial f(x) of degree d with rational coeffiecients is known. Then f(x) splits in to distinct linear factors over the complex numbers as*

$$f(x) = (x - \theta_1)(x - \theta_2) \cdots (x - \theta_d)$$

with $\theta \in \mathbb{C}$. One can choose any root $\theta = \theta_i$ and form a ring.

Proposition 3.6.8. *If θ denotes a complex root of a monic, irreducible polynomial f(x) with rational coefficients, then the set of all polynomials in θ with rational coefficients, denoted $\mathbb{Q}(\theta)$, forms a ring.*

Theorem 3.6.9. *Given a monic, irreducible polynomial f(x) with rational coefficients, a root $\theta \in \mathbb{C}$ of f(x), and the associated ring $\mathbb{Q}(\theta)$, the following hold :*
(1). $\mathbb{Q}(\theta) \cong \mathbb{Q}[x]/(f(x))$.
(2). $\mathbb{Q}(\theta)$ is a field.
(3). f(x) divides any polynomial g(x) for which $g(\theta) = 0$.
(4). The set $\{1, \theta, \theta^2, \cdots, \theta^{d-1}\}$ forms a basis for $\mathbb{Q}(\theta)$ as a vector space over $\mathbb{Q}$.

Rings of Algebraic Integers

Definition 3.6.10. *A complex number α is called an algebraic integer if it is the root of a monic polynomial with integer coefficients.*

Thus, if f(x) is an irreducible, monic polynomial of degree d with integer coefficients and $\theta \in \mathbb{C}$ is a root of f(x), it follows that θ is an algebraic integer according to this definition.

Proposition 3.6.11. *Given a monic, irreducible polynomial f(x) of degree d with rational coefficients and a root $\theta \in \mathbb{C}$ of f(x), the set of all algebraic integers in $\mathbb{Q}(\theta)$, denoted $\eth$, forms a subring of the field $\mathbb{Q}(\theta)$.*

The actual ring that will be used in the GNFS is subring of the ring of algebraic integers $\eth$ of $\mathbb{Q}(\theta)$.

Proposition 3.6.12. *Given a monic, irreducible polynomial f(x) of degree d with integer coefficients and a root $\theta \in \mathbb{C}$ of f(x), the set of all $\mathbb{Z}$-linear combinations of the elements $\{1, \theta, \theta^2, \cdots, \theta^{d-1}\}$, denoted $\mathbb{Z}[\theta]$, forms a subring of the ring of algebraic integers $\eth$ of $\mathbb{Q}(\theta)$.*

Before going further, it should be pointed out that the subring $\mathbb{Z}[\theta]$ can be indeed be a proper subring of $\eth$. For instance, the polynomial x^2-5 is easily seen to be irreducible and monic so that $\mathbb{Q}(\sqrt{5})$ forms a field, which is a vector space over $\mathbb{Q}$ with basis S = 1, $\sqrt{5}$. If α = (1+$\sqrt{5}$)/2 then $\alpha \in \mathbb{Q}(\sqrt{5})$, since α is a $\mathbb{Q}$ linear combination of the elements of S. Furthermore α satisfies the polynomial g(x) = x^2-x-1 and hence it is an algebraic integer, but clearly α does not belong to $\mathbb{Z}[\theta]$. Hence, $\mathbb{Q}(\sqrt{5})$ possesses an algebraic integer which is not contained in $\mathbb{Z}[\sqrt{5}]$ and so $\mathbb{Z}[\sqrt{5}] \subsetneqq \eth \subsetneqq \mathbb{Q}(\sqrt{5})$.

Producing a difference of Squares *Having demonstrated that for any monic, irreducible polynomial an associated ring can be constructed that has a natural representation as $\mathbb{Z}$-linear combinations of elements from a finite set. Now we will describe a map that ring onto $\mathbb{Z}/n\mathbb{Z}$ which produce a difference of squares.*

Proposition 3.6.13. *Given a monic, irreducible polynomial f(x) with integer coefficients, a root $\theta \in \mathbb{C}$ of f(x), and an integer $m \in \mathbb{Z}/n\mathbb{Z}$ with $\phi(1) = 1 \pmod n$ and which sends θ to m, is a surjective ring homomorphism.*

Now, to see how this can result in a difference of squares, suppose a set U of pairs of integers (a, b) can be found such that

$$\prod_{(a,b)\in U} (a + b\theta) = \beta^2 \quad and \quad \prod_{(a,b)\in U} (a + bm) = y^2$$

with $\beta \in \mathbb{Z}[\theta]$ and $y \in \mathbb{Z}$. Then applying the natural homomorphism ϕ from proposition 3.6.13 and letting $\phi(\beta) = x \in \mathbb{Z}/n\mathbb{Z}$, it follows that

$$x^2 \equiv \phi(\beta)^2 \equiv \phi(\beta^2) \equiv \phi(\prod_{(a,b)\in U}(a+b\theta)) \equiv \prod_{(a,b)\in U}\phi(a+b\theta) \equiv \prod_{(a,b)\in U}(a+bm) \equiv y^2 (mod\, n)$$

and a difference of square results.

Note : *The condition that the product of the elements $a+b\theta$ corresponding to pairs in U be a perfect square in $\mathbb{Z}[\theta]$ is imposed because the ring homomorphism ϕ is only defined on elements of $\mathbb{Z}[\theta]$. In practice, the condition is relaxed to allow for the product being a perfect square in $\mathbb{Q}(\theta)$, which is less restrictive and hence more likely to be satisfied. Now if*

$$\prod_{(a,b)\in U}(a+b\theta) = \alpha^2$$

for some $\alpha \in \mathbb{Q}(\theta)$, it follows that $\alpha \in \eth$ and in fact f'(θ)·α ∈ $\mathbb{Z}[\theta]$. A difference of squares can still be produced, for if

$$\prod_{(a,b)\in U}(a+b\theta) = \alpha^2 \quad and \quad \prod_{(a,b)\in U}(a+bm) = z^2$$

with $\alpha \in \eth$ and $z \in \mathbb{Z}$, then letting $\beta = f'(\theta)\cdot\alpha \in \mathbb{Z}[\theta]$, $y = f'(m)\cdot z$, and $x = \phi(\beta) \in \mathbb{Z}/n\mathbb{Z}$, it follows that

$$x^2 \equiv \phi(\beta^2) \equiv \phi(f'(\theta)^2 \prod_{(a,b)\in U}(a+b\theta)) \equiv \phi(f'(\theta))^2 \prod_{(a,b)\in U}\phi(a+b\theta)$$

$$\equiv f'(m)^2 \prod_{(a,b)\in U}(a+bm) \equiv y^2 (mod\, n) \tag{3.2}$$

and another difference of squares has been produced.

Now we will describe some definition like norm, first degree prime ideal and their some properties.

Proposition 3.6.14. *Given a monic, irreducible polynomial f(x) of degree d with integers coefficients and a root $\theta \in \mathbb{C}$ of f(x), then the ring of algebraic integers $\eth$ forms a Dedekind domain. In particular, this implies :*

(1) The ring $\eth$ *is noetherian.*
(2) Prime ideals of $\eth$ *are maximal ideals of* $\eth$*, and vice versa.*
(3) Using the canonical notation of ideal multiplication, non-zero ideals of $\eth$ *can be uniquely factored, up to order, into prime ideals of* $\eth$*.*

The idea then is to choose a set I of prime ideals of $\eth$*, which will be called an* ***algebraic factor base****, and to find (a, b) pairs for which the element* $a+b\theta$ *has a principal ideal* $\langle a+b\theta \rangle$ *that factors completely into prime ideals of I. Such an element is said to be* ***smooth*** *over the algebraic factor base I. By collecting more (a, b) pairs than ideals in I, hopefully some of the* $a+b\theta$ *values corresponding these pairs can be multiplied together to produce a perfect square in* $\mathbb{Z}[\theta]$*.*

Concept of Norm :

Theorem 3.6.15. *Given a monic, irreducible polynomial f(x) of degree d with rational coefficients and a root* $\theta \in \mathbb{C}$ *of f(x), there are exactly d ring monomorphisms (embeddings) from the field* $\mathbb{Q}(\theta)$ *into the field* $\mathbb{C}$*. These embeddings are given by* $\sigma_i(\mathbb{Q}) = \mathbb{Q}$ *and* $\sigma_i(\theta) = \theta_i$ *for* $1 \leq i \leq d$*, assuming f(x) splits over* $\mathbb{C}$ *as :*

$$f(x) = (x - \theta_1)(x - \theta_2) \cdots (x - \theta_d).$$

The embeddings of theorem 3.6.15 allow for the definition of the norm function which maps elements of $\mathbb{Q}(\theta)$ *to elements of* $\mathbb{C}$ *:*

Definition 3.6.16. *Given a monic, irreducible polynomial f(x) of degree d with rational coefficients, a root* $\theta \in \mathbb{C}$ *of f(x) and an element* $\alpha \in \mathbb{Q}(\theta)$*, the norm of an element* α*, denoted by N(*α*), is defined as*

$$N(\alpha) = \sigma_1(\alpha)\sigma_2(\alpha) \cdots \sigma_d(\alpha)$$

where the σ_i *are in the distinct embeddings of* $\mathbb{Q}(\theta)$ *into* $\mathbb{C}$ *as detailed in the theorem 3.6.15.*

Example 3.6.17. *Let in the above definition* $\alpha = (a+b\theta)$*.*
Then $N(a+b\theta) = \sigma_1(a+b\theta) \cdot \sigma_2(a+b\theta) \cdots \sigma_d(a+b\theta)$
$N(a+b\theta) = (a+b\theta_1) \cdot (a+b\theta_2) \cdots (a+b\theta_d)$
$N(a+b\theta) = b^d[(\frac{a}{b}+\theta_1) \cdot (\frac{a}{b}+\theta_2) \cdots (\frac{a}{b}+\theta_d)]$
$N(a+b\theta) = (-b)^d[(-\frac{a}{b}-\theta_1) \cdot (-\frac{a}{b}-\theta_2) \cdots (-\frac{a}{b}-\theta_d)]$
$N(a+b\theta) = (-b)^d f(-\frac{a}{b})$

The real power of the norm function, as it is used in the GNFS, stems from the following standard result from algebraic number theory :

Proposition 3.6.18. *Given a monic, irreducible polynomial f(x) of degree d with rational coefficients and a root $\theta \in \mathbb{C}$ of f(x), the norm map of definition 3.6.16 is a multiplicative function that maps elements of $\mathbb{Q}(\theta)$ to $\mathbb{Q} \in \mathbb{C}$. Furthermore, algebraic integers in $\mathbb{Q}(\theta)$ are mapped to elements of $\mathbb{Z}$.*

Corollary 3.6.19. *Given a monic, irreducible polynomial f(x) of degree d with integer coefficients and a root $\theta \in \mathbb{C}$ of f(x), then the norm function of definition 3.6.16 is a multiplicative function that sends elements of $\mathbb{Z}[\theta]$ to elements of $\mathbb{Z}$.*

Though proposition 3.6.18 and its corollary are initially useful because they allow for recasting of questions about factorizations of elements of $\mathbb{Z}[\theta]$ to factorization in $\mathbb{Z}$, the full power of these results comes when the concept of the norm of an element is tied in with the norm of an ideal. Begin with the definition of norm of an ideal :

Definition 3.6.20. *Given a ring R and an ideal I of R, the norm of I is defined to be [R : I], the number of cosets of I in R.*

(Note : The above definition is similar to the definition, which is described in Note (1) before definition 3.6.1 in the SNFS.)

The following results recall elementary properties of the norm function on ideals of $\mathfrak{d}$, and explicitly relates the norm of an element of $\mathfrak{d}$ to the norm of the principal ideal generated by that element :

Proposition 3.6.21. *Let f(x) be a monic, irreducible polynomial of degree d with rational coefficients and $\theta \in \mathbb{C}$ a root of f(x). Then the norm function of definition 3.6.20 is a multiplicative function that maps ideals of $\mathfrak{d}$ to positive integers. Moreover, if $\alpha \in \mathfrak{d}$ then $N(\langle\alpha\rangle) = |N(\alpha)|$.*

Now, we will describe a final result from algebraic number theory, which clarifies that how prime ideals of $\mathfrak{d}$ and prime integers are related :

Proposition 3.6.22. *Let D be a Dedekind domain. If P is an ideal of D with N(P) = p for some prime integer p, then P is a prime ideal of D. Conversely, if P is a prime ideal of D then $N(P) = p^e$ for some prime integer p and positive integer e.*

Given any element $\beta \in \mathfrak{O}$, it follows from proposition 3.6.14 that the principal ideal $\langle\beta\rangle$ of $\mathfrak{O}$ factors uniquely as

$$\langle\beta\rangle = P_1^{e_1} P_2^{e_2} \cdots P_k^{e_k}$$

for distinct prime ideals P_i of $\mathfrak{O}$ and positive integers e_i with $1 \leq i \leq k$. Furthermore, Proposition 3.6.22 indicate that
$|N(\beta)| = N(\langle\beta\rangle) = N(P_1^{e_1} P_2^{e_2} \cdots P_k^{e_k}) = N(P_1)^{e_1} N(P_2)^{e_2} \cdots N(P_k)^{e_k} = (p_1^{f_1})^{e_1} (p_2^{f_2})^{e_2} \cdots (p_k^{f_k})^{e_k} = p_1^{e_1+f_1} p_2^{e_2+f_2} \cdots p_k^{e_k+f_k}$
for (not necessarily distinct) primes p_i and positive integers e_i and f_i with $1 \leq i \leq k$. It becomes the key tool for determining when an ideal $\langle a+b\theta\rangle$ factors completely over an algebraic factor base of prime ideals.

One very practical problem that presents itself is coming up with a representation for prime ideals that can be stored in a computer, and more importantly, that facilitates a sieving procedure for finding smooth $a+b\theta$ values. This is accomplished in GNFS by restricting the algebraic factor base to prime ideals of $\mathbb{Z}[\theta]$ of a special form instead of prime ideals of $\mathfrak{O}$, and then generalizing the above equation to these ideals. With this in mind, begin by defining the special prime ideals of $\mathbb{Z}[\theta]$, which will be used in the algebraic factor base :

Definition 3.6.23. *A first degree prime ideal P of a Dedekind domain D is a prime ideal of D such that $N(P) = p$ for some prime integer p (similar to definition 3.6.1).*

Note : *It should be observed that any ideal P of a ring R with $N(P) = p$ for some prime integer p is necessarily a prime ideal of R. This follows, since $[R : P] = p$ implies that $R/P \cong \mathbb{Z}/n\mathbb{Z}$ is a field and hence P is a maximal ideal of R. But any maximal ideal of R is also a prime ideal of R.*
Before preceeding to determine a good representation for the first degree prime ideals of $\mathbb{Z}[\theta]$, we will describe a lemma :

Lemma 3.6.24. *If R is a commutative ring with identity 1_R, S is a commutative ring with identity 1_S, and $\phi : R \longrightarrow S$ is a ring epimorphism, then $\phi(1_R) = 1_S$.*

Proof : *Let $y \in S$. Since ϕ is a ring epimorphism, so, there exists $x \in R$ such that $\phi(x) = y$. Then $y \cdot \phi(1_R) = \phi(1_R) \cdot y = \phi(1_R) \cdot \phi(x) = \phi(1_R \cdot x) = \phi(x) = y$, hence $\phi(1_R) = 1_S$.*

The following result gives the convenient representation for the first degree prime ideals (This following theorem prove the same thing which we have proved in lemma 3.6.2 and lemma 3.6.3) :

Theorem 3.6.25. *Let f(x) be a monic, irreducible polynomial with integer coefficients and $\theta \in \mathbb{C}$ a root of f(x). The set of pairs (r, p) where p is a prime integer and r $\in \mathbb{Z}/p\mathbb{Z}$ with f(r) $\equiv$ 0 (mod p) is in bijective correspondence with the set of all first degree prime ideals of $\mathbb{Z}[\theta]$.*

Proof. Let P be a first degree prime ideal of $\mathbb{Z}[\theta]$. Then $[\mathbb{Z}[\theta] : P] = p$ for some prime integer p, so that $\mathbb{Z}[\theta]/P \cong \mathbb{Z}/p\mathbb{Z}$ (also proved in lemma 3.6.2). There is a canonical epimorphism of rings $\phi : \mathbb{Z}[\theta] \longrightarrow \mathbb{Z}[\theta]/P$ such that $\ker\phi = P$. Since $\mathbb{Z}[\theta]/P \cong \mathbb{Z}/p\mathbb{Z}$, it follows that ϕ can also be thought of as an epimorphism of rings $\phi : \mathbb{Z}[\theta] \longrightarrow \mathbb{Z}[\theta]/p\mathbb{Z}$ with $\ker\phi = P$, that is the elements in P map to integers that are divisible by p, and any such integer is the image of an element in P. Furthermore $\phi(1) = 1$ by lemma 3.6.24 and hence $\phi(a) \equiv a \pmod p$ for any integer a.
Let $r = \phi(\theta) \in \mathbb{Z}/p\mathbb{Z}$. If $f(x) = x^d + a_{d-1}x^{d-1} + \cdots + a_1x + a_0$ with $a_i \in \mathbb{Z}$ for $0 \leq i < d$, then $\phi(f(\theta)) \equiv 0 \pmod p$ since $f(\theta) = 0$ and hence

$$0 \equiv \phi(f(\theta)) \equiv \phi(\theta^d + a_{d-1}\theta^{d-1} + \cdots + a_\theta + a_0) \equiv \phi(\theta)^d + a_{d-1}\phi(\theta)^{d-1} + \cdots + a_1\phi(\theta) + a_0$$

$$\equiv r^d + a_{d-1}r^{d-1} + \cdots + a_1r + a_0 \equiv f(r) \ (mod\ p)$$

so that r is a root of f(x) (mod p) and the ideal P determines the unique pair (r, p).

Conversely, let p be a prime integer and $r \in \mathbb{Z}/p\mathbb{Z}$ with $f(r) \equiv 0 \pmod p$. Then there is a natural ring epimorphism (analogous to the one discussed in theorem 3.6.9) that maps polynomials in θ to polynomials in r. In particular, $\phi(a) \equiv a \pmod p$ for all $a \in \mathbb{Z}$ and $\phi(\theta) = r \pmod p$. Let $P = \ker\phi$ so that P is an ideal of $\mathbb{Z}[\theta]$. Since ϕ is onto and $\ker\phi = P$, it follows that $\mathbb{Z}[\theta]/P \cong \mathbb{Z}/p\mathbb{Z}$ and hence $[\mathbb{Z}[\theta] : P] = p$ and P is therfore first degree prime ideal of $\mathbb{Z}[\theta]$. Thus the pair (r, p) determines the unique pair (r, p) consistent with the first part of the proof. This gives the result. □

Lemma 3.6.4 "which says Let a, b $\in \mathbb{Z}$, gcd(a, b) = 1. Then all prime ideals P that occur in a+bθ are the first degree prime ideals" holds here.

3.6.4 GNFS Algorithm

Input : Composite integer n.
Output : a nontrivial factor p of n.
Step (1) : *(Polynomial selection)*
Find an irreducible polynomial f(x) with root m, i.e. f(m) ≡ 0 mod n, f(x) ∈ ℤ[x].
Method :
Decide the degree d of the polynomial (experimently, we see that, for factoring an integer with more than 110 digits, the degree d be set to 5. For integers between 50 and 80 digits a value of 3 for d is used. Since every implementations of GNFS restricted d to an odd integer, so for integers between 80 and 110 digits, d = 5 is used).
Having selected a value for d, the choice of f(x) and m is usually made simultaneously. First m is chosen with $m \approx n^{\frac{1}{d}}$ *and such that the quotient of n divided by* m^d *is exactly one.*
A "base - m" expansion of n then gives

$$n = m^d + a_{d-1}m^{d-1} + \cdots + a_1 m + a_0$$

with coefficients $0 \leq a_i < m$ *for* $0 \leq i < d$. *These coefficients may then be used to construct*

$$f(x) = x^d + a_{d-1}x^{d-1} + \cdots + a_1 x + a_0$$

which is monic of degree d. By construction f(m) = n ≡ 0 (mod n) so that m is a root modulo n of f(x). Furthermore if f(x) = g(x)·h(x) for non-constant polynomial g(x) and h(x) and it follows that
n = f(m) = g(m)·h(m) is likely to yield a non-trivial factorization of n. Thus, if f(x) is reducible then n is likely is to be factored and the whole procedure can terminate, or f(x) is irreducible and we can go on further steps for finding out factors of n.
Step (2) : *(Factor bases) Choose the size for the factor bases and set up the rational factor base, algebraic factor base and the quadratic character base.*
The rational factor base ***RFB*** *consists of all primes* p_i *up to some bound which is usually determined by experimenting with the smoothness of a+bm for different (a, b) pairs. The rational factor base is stored as pairs (m (mod p), p).*
The algebraic factor base ***AFB*** *consists of first degree prime ideals of* $\mathbb{Z}[\theta]$, *which are represented as pairs (r, p) where p is a prime integer and r is the root of f(x) considered as a polynomial with coefficients in* $\mathbb{Z}/p\mathbb{Z}$ *(as proved in theorem 3.6.25).*

The size of the algebraic factor base should be 2-3 times the size of the rational factor base.
The quadratic character base ***QCB*** *contains pairs (r, p) with the same properties as the elements of the algebraic factor base, but the p's are larger than the largest in the algebraic factor base. The number of elements in QCB are usually relatively small compared to the number of elements in RFB and AFB.*
Method of finding first degree prime ideals of $\mathbb{Z}[\theta]$ ***or AFB*** *: Finding first degree prime ideals of* $\mathbb{Z}[\theta]$ *for the AFB and the QCB amounts to finding integers pairs (r, p) with p a prime and r satisfying f(r)* $\equiv$ *0 (mod p) according to theorem 3.6.25. In other words, finding first degree prime ideals is equivalent to finding roots of f(x) modulo p for various prime integers p. Fortunately, this happens to be well studied problem in a natural and efficient way.*
A naive approach to finding roots of the polynomial f(x) (mod p) is to simply plug-in all the integers from 0 to p-1 and determine which values are mapped to 0 by f(x). As with most brute-force approaches, this words well for a small number of cases, specially when p is small, but becomes quite impractical for the larger values of p used in GNFS.
A dramatic improvement over this brute-force method can be made using the following result in a clever way :

Theorem 3.6.26. *When considered as a polynomial in* $\mathbb{Z}/p\mathbb{Z}[x]$*, the polynomial* x^p*-x factors as*

$$x^p - x = \prod_{i=0}^{p-1}(x - i)$$

Proof : *It is an elementary result from abstract algebra that the non-zero elements of a field form a group under multiplication. In this case, that means the p-1 non-zero elements of* $\mathbb{Z}/p\mathbb{Z}$ *form a finite group of order p-1 under multiplication. Then for any* $0 < a < p$ *it follows that* $a^{p-1} \equiv 1$ *(mod p) and therefore* $a^p \equiv a$ *(mod p) for all a with* $0 \leq a < p$*. Rearranging the last congruence yields* $a^p - a = 0$ *(mod p) and therefore a is seen to be a root of* x^p*-x (mod p) for* $0 \leq a < p$*. This determines p roots for* x^p*-x (mod p). But* x^p*-x determines a monic, linear factor of* x^p*-x (mod p) and vice versa. So, the result follows.*

Since finding roots of f(x) (mod p) is equivalent to finding monic, linear factors of f(x) (mod p), and $x^p - x$ *(mod p) is the product of all the monic, linear polynomial over* $\mathbb{Z}/p\mathbb{Z}$ *by above theorem, a natural idea is to somehow use* x^p*-x (mod p) in the*

root finding procedure. With this in mind, the first realization is that finding roots of f(x) (mod p) is equivalent to finding roots of g(x) = gcd(f(x), x^p - x). The effect of computing g(x) (mod p) is to isolate the portion of f(x) (mod p) which is the product of monic, linear polynomials over $\mathbb{Z}/p\mathbb{Z}$, since this is the portion where the roots of f(x) (mod p) are to be found. Another way of thinking of this computation is as a way to "strip out" of f(x) (mod p) any quadratic or higher degree polynomials that occur in its canonical factorization into irreducibles, since such polynomials have nothing to do with roots of f(x) (mod p).

Now let b be any integer with $0 \leq b < p$. Since g(x) (mod p) divides x^p-x (mod p). It must be product of distinct, monic, linear polynomials, and therefore so is g(x-b) (mod p). If x is a factor of g(x-b) (mod p) then g(-b) $\equiv$ 0 (mod p) so a root -b of g(x) (mod p) and hence of f(x) (mod p) has been found. On the other hand, if x is not a factor of g(x-b) (mod p) then

$$g(x-b)|x^p - x = x(x^{p-1} - 1) = x(x^{(p-1)/2} + 1)(x^{(p-1)/2} - 1)$$

and the factors of g(x-b) (mod p) fall between $(x^{(p-1)/2} + 1)$ (mod p) and $x^{(p-1)/2} - 1)$ (mod p). If not all of the factors of g(x-b) (mod p) divide into either of these latter polynomials, i.e. if $x^{(p-1)/2} \neq \pm 1$ (mod g(x-b)), then g(x-b) (mod p) can be split non-trivially into the polynomials $g_1(x)$ = gcd(g(x-b), $x^{(p-1)/2}$+1) and $g_2(x)$ = gcd(g(x-b), $x^{(p-1)/2}$-1), with the degree of each polynomial strictly less than the degree of g(x) (mod p).

If $g_i(x)$ (mod p) is a monic polynomial then a root of g(x) (mod p) has been found. Otherwise, the same procedure outlined above is applied to each $g_i(x)$ (mod p) to split them into lesser degree polynomials. The algorithm continues on in this manner until it terminate since polynomials are produced at each stage with degrees strictly less than the degrees of the polynomials of the previous stage.

Note : *In the event that $x^{(p-1)/2} \equiv \pm 1$ (mod g(x-b)), other values for b are substituted until this condition no longer holds. Also note that a root r of g(x-b) (mod p) gives rise to the root r-b of g(x) (mod p), and that r itself is not a root of g(x) (mod p) unless b = 0.*

Step (3) : *(Sieving)*

Find pairs of integers (a, b) with the following properties :

(I) gcd(a, b) = 1

(II) |a+bm| is smooth over the rational factor base.

(III) N(a+bθ) is smooth over the algebraic factor base.

A pair (a, b) with these properties is called a relation. The purpose of the sieving stage is to collect as many relations as possible (at least one larger than the elements in all of the base combined). The sieving step results in a set S of relations.
Here condition one ensures that all ideals containing a+bθ are first degree prime ideals, as proved by lemma 3.6.4.
*(**Note :** The technique, which is useful in th sieving, is the similar to the sieving technique what we have used in SNFS. This is described in the Note (3) after the SNFS Algorithm or before the example 3.6.5.)*
***Step (4) :** (Linear Algebra)*
Once enough relations have been found, it is straight forward to put the exponent vectors ($e_{(a,b)}$) in to a matrix (modulo 2) and start row reducing to find a zero row, that is a linear dependent set of vectors.

Method for Matrices and Dependencies :

If there are k primes in the rational factor base, l first degree prime ideals of $\mathbb{Z}[\theta]$ in the algebraic factor base, and m first degree prime ideals in the quadratic character base, then each $e_{(a,b)}$ will be comprised of 1+k+l+m binary bits, determined by the sign of a+bm and the respective bases. When these binary vectors are grouped together as columns in a matrix B, the binary vector resulting from the addition of the two columns for pairs (a, b) and (c, d) represents (a+bm)·(c+dm) and $\langle a+b\theta\rangle \cdot \langle c+d\theta\rangle$ factored over the rational and algebraic factor bases, respectively, and the quadratic characters for (a+bθ)·(c+dθ). From these relations, we can select a set of those relations which gives a non-trivial dependency among the columns of the matrix B, which yields a product of different a+bm and a+bθ values that gives a square in $\mathbb{Z}$ and $\mathbb{Z}[\theta]$. The first bit of $e_{(a,b)}$ is 0 if a+bm is positive and 1 if it is negative, in which case addition modulo 2 of binary vectors $e_{(a,b)}$ and $e_{(c,d)}$ correctly reflects the sign of (a+bm)·· · (c+dm). The next k-bits of $e_{(a,b)}$ is determined by the exponents modulo 2 of every prime in the rational factor base F, when a+bm is factored over F. Similarly, the next l-bits of $e_{(a,b)}$ are determined by the exponents modulo 2 of the primes p in the first degree prime ideal pairs (r, p) in the algebraic factor base when N(a+bθ) is factored over these primes.
Note that if p divides N(a+bθ) then there is exactly one (r, p) pair in the algebraic factor base for which a ≡ -br (mod p), and that is the (r, p) which is deemed "responsible" for the exponent of the prime p occuring in the factorization of N(a+bθ). It's clear that addition modulo 2 of binary vectors $e_{(a,b)}$ and $e_{(c,d)}$ corresponds to the binary vector represented by (a+bm)·(c+dm) and $\langle a+b\theta\rangle \cdot \langle c+d\theta\rangle$ since the latter two

multiplication essentially involve addition of exponents.
The final l bits of the vector $e_{(a,b)}$ are determined by each (s, q) = (r, p) pair in the quadratic character base. For a fixed (s, q) pair the corresponding bit in $e_{(a,b)}$ is set to zero if the Legendre symbol $(\frac{a+bs}{q})$ has value 1 and is set to 1 otherwise. This last representation preserves the multiplicative nature of the Legendre symbol in the exact same way the sign of a+bm is preserved during multiplication by the first bit of $e_{(a,b)}$. A non-trivial dependence among the column vectors of B represents a set U of (a,b) pairs for which the product of the corresponding a+bm values is a square in $\mathbb{Z}$ and a square has also been produced in $\mathbb{Z}[\theta]$.
Step (5) : *When all the problems with finding the factor base and, the right combination of relations during the sieving and linear algebra step has been overcome, one is left with a set U with the relations that combined are squares both in $\mathbb{Z}$ and $\mathbb{Z}[\theta]$. If*

$$\prod_{(a,b)\in U}(a+b\theta)=\alpha^2 \quad and \quad \prod_{(a,b)\in U}(a+bm)=z^2$$

with $\alpha \in \eth$ and $z \in \mathbb{Z}$, then letting $\beta = f'(\theta)\cdot\alpha \in \mathbb{Z}[\theta]$, $y = f'(m)\cdot z$, and $x = \phi(\beta) \in \mathbb{Z}/n\mathbb{Z}$
Now as, we have seen in equation 3.6.3 that from this, we get :

$$x^2 \equiv \phi(\beta^2) \equiv \phi(f^{'}(\theta)^2 \prod_{(a,b)\in U}(a+b\theta)) \equiv \phi(f^{'}(\theta))^2 \prod_{(a,b)\in U}\phi(a+b\theta)$$

$$\equiv f^{'}(m)^2 \prod_{(a,b)\in U}(a+bm) \equiv y^2(modn)$$

Then gcd(x ± y, n), hopefully gives a nontrivial factor of n. Otherwise choose a different set of linear dependent vectors or increase bound of AFB, RFB and QCB for find out more smooth pairs (a, b).

Example 3.6.27. *Suppose we have to factor n = 45,113 from GNFS. Then we will follow the following steps :*
Step (1) :
Choose degree d = 3. $m \approx n^{1/d} = (45{,}113)^{1/3} \approx 35$. Although m = 35 would yield a monic polynomial following the base-m method of step (1), the value m = 31 serves equally well and is used in this example. The base-m expansion of n:
$45{,}113 = 31^2 + 15\cdot 31^2 + 29\cdot 31 + 8$
yields the polynomial $f(x) = x^3 + 15x^2 + 29x + 8$ for which $f(m) \equiv 0 \pmod n$ since f(x) was constructed with f(31) = 45,113.

Now, the only concern is that f(x) be irreducible over $\mathbb{Q}$, which amounts to verifying that f(x) does not have any roots over $\mathbb{Q}$ since f(x) is cubic. The only possible roots are ± 1, ± 2, ± 4 and ± 8. The positive portions of these possibilities can be immediately dispensed with so only four possibilities for roots need be considered. Now f(-1) = -7, f(-2) = 2, f(-4) = 68 and f(-8) = 224, so that f(x) has no rational roots and hence is irreducible over $\mathbb{Q}$.

Step (2) :

Rational factor base

The rational factor base consists of prime integers 2, 3, 5, 7 and so on up to a particular bound which is usually determined by experimenting with the smoothness of a+bm for different (a, b) pairs. In this example, all the primes up to 29 are used. Rational factor base is stored as pairs (m (mod p), p).

Table (3.6.1)

(m (mod p), p)	*(m (mod p), p)*	*(m (mod p), p)*
(1, 2)	*(9, 11)*	*(8, 23)*
(1, 3)	*(5, 13)*	*(2, 29)*
(1, 5)	*(14, 17)*	
(3, 7)	*(12, 19)*	

Algebraic factor base

The algebraic factor base consists of first degree prime ideals of $\mathbb{Z}[\theta]$, which are represented as pairs (r, p) where p is a prime integer and r is a root of $f(x) = x^3+15x^2+29x+8$ considered as a polynomial with coefficients in $\mathbb{Z}/p\mathbb{Z}$. Computing the algebraic factor base then amounts to finding roots of f(x) modulo 2, 3, 5, 7 and so on.

Using the methods which is discussed in the step (2) of the algorithm, and the prime 67 as an example, start by computing the polynomial $g(x) = gcd(f(x), x^{67}-x)$, where g(x) serves to isolate the linear factors of f(x). In this case, $g(x) \equiv f(x) \pmod{67}$ so that f(x) consists of all linear factors and hence must have three roots over $\mathbb{Z}/67\mathbb{Z}$.

Now $g(0) = 8 \pmod{67}$ so that 0 is not a root of g(x). Since g(x) divides $x^{67}-x = x(x^{33}+1)(x^{33}-1)$, it follows that g(x) must divide $(x^{33}+1)(x^{33}-1)$ and in fact

$g(x) \equiv gcd(x^{33}+1, f(x))\cdot gcd(x^{33}-1, f(x)) \equiv (x^2+21x+21)\cdot(x+61) \pmod{67}$

Hence, $6 \equiv -61 \pmod{67}$ is a root of g(x) in $\mathbb{Z}/67\mathbb{Z}$ and the pair (6, 67) represents a first degree prime ideal of $\mathbb{Z}[\theta]$ that may be added to the algebraic factor base.

The same process can be used to determine the linear factors of $g_1(x) = x^2+21x+21$.

Now $g_{-1} \equiv 1 \pmod{67}$ so that -1 is not a root of $g_1(x)$ and hence $g_1(x\text{-}1)$ must divide $(x^{33}+1)(x^{33}\text{-}1)$ for the same reason g(x) does. However, $gcd(x^{33}\text{-}1,\ g_1(x\text{-}1)) \equiv 1 \pmod{67}$ so that $g_1(x\text{-}1)$ can't be immediately split into non-trivial factors as was done with g(x) in above equation.

Continuing on with $g_{-2} \equiv 50 \pmod{67}$, it is seen that -2 is not a root of $g_1(x)$ and hence $g_1(x\text{-}2)$ must then divide $(x^{33}+1)(x^{33}\text{-}1)$. This time luck prevails and

$g_1(x\text{-}2) \equiv gcd(x^{33}+1,\ g_1(x\text{-}2))\cdot gcd(x^{33}\text{-}1,\ g_1(x\text{-}2)) \equiv (x+21)\cdot(x+63) \pmod{67}$.

The latter yields $46 \equiv \text{-}21 \pmod{67}$ and $4 \equiv \text{-}63 \pmod{67}$ as roots of $g_1(x\text{-}2)$, so that 44 and 2 are roots of $g_1(x)$ over $\mathbb{Z}/67\mathbb{Z}$ and hence the pairs (2, 67) and (44, 67) represent first degree prime ideals of $\mathbb{Z}[\theta]$ which may be used in the algebraic factor base.

Roots finding with primes other than 67 is performed in the exact same manner to determine the rest of the algebraic factor base shown in the table below.

Table (3.6.2)

(r, p)	(r, p)	(r, p)	(r, p)
(0, 2)	(19, 41)	(44, 67)	(62, 89)
(6, 7)	(13, 43)	(50, 73)	(73, 89)
(13, 17)	(1, 53)	(23, 79)	(28, 97)
(11, 23)	(46, 61)	(47, 79)	(87, 101)
(26, 29)	(2, 67)	(73, 79)	(47, 103)
(18, 31)	(6, 67)	(28, 89)	

Quadratic character base

Since the quadratic character base is simply a small set of first degree prime ideals of $\mathbb{Z}[\theta]$ that don't occur in the algebraic factor base, in practice one begins searching for roots of f(x) modulo primes q, with q strictly greater than the largest prime p occurring in a (r, p) pair in the algebraic factor base. The worked example of the AFB serves as sample illustration of how the quadratic character base seen in the table below is computed.

Table (3.6.3)

(r, p)	(r, p)	(r, p)
(4, 107)	(80, 107)	(99, 109)
(8, 107)	(52, 109)	

Step (3) :

For this example, the sieving interval is chosen such that -1000 < a < 1000 for b starting at 1 and proceeding through 2, 3, 4 and so on until more than 39 pairs are found with a+bm and a+bθ smooth. Finding more than 39 pairs will guarantee a linear dependence among the binary vectors associated with those pairs, which leads to perfect squares in $\mathbb{Z}$ and $\mathbb{Z}[\theta]$, as explained in last step of the algorithm.

A straight forward implementation technique is to have two Sieve arrays in memory, one for a+bm and the other for N(a+bθ), each with 2000 entries for all possible a values for a fixed b. Sieving for smooth values of a+bm proceeds exactly as in Note (3) which is before the example 3.6.5 in SNFS algorithm. For instance, the values of a for which a+bm is divisible by the prime 5 for b = 7 are of the form a = -7m + 5k for k ∈ $\mathbb{Z}$ such that -1000 < a < 1000. From table 3.6.1, it is seen that m ≡ 1 (mod 5) and hence a is of the form a = -7 + 5k for k ∈ $\mathbb{Z}$. The positions in the Sieve array for a+bm corresponding to an a value of -997, -992, ⋯, -12, -7, -2, 3, 8, 13, ⋯, 993, 998 then have ln(5) added to their value. This procedure is repeated for all the pairs of table (5.1). A similar procedure is followed with the (r, p) pairs of table (5.2) and the Sieve array for N(a+bθ) are trail divided to test for smoothness. The whole procedure is then repeated for the next value of b.

After enough sieving, 40 (a, b) pairs are found with a+bm and a+bθ, as seen in following table (5.4).

Table (5.4)

(a, b) pair	*(a, b) pair*	*(a, b) pair*	*(a, b) pair*	*(a, b) pair*	*(a, b) pair*	*(a, b) pair*
(-73, 1)	*(-2, 1)*	*(-1, 1)*	*(2, 1)*	*(3, 1)*	*(4, 1)*	*(8, 1)*
(13, 1)	*(14, 1)*	*(15, 1)*	*(32, 1)*	*(56, 1)*	*(61, 1)*	*(104, 1)*
(116, 1)	*(-5, 2)*	*(3, 2)*	*(25, 2)*	*(32, 2)*	*(-8, 3)*	*(2, 3)*
(17, 3)	*(19, 4)*	*(48, 5)*	*(54, 5)*	*(313, 5)*	*(-43, 6)*	*(-8, 7)*
(11, 7)	*(38, 7)*	*(44, 9)*	*(4, 11)*	*(119, 11)*	*(856, 11)*	*(536, 15)*
(5, 17)	*(5, 31)*	*(9, 32)*	*(-202, 43)*	*(24, 55)*		

Step (4) : *(Linear algebra)*

To find the set U in note (before proposition 3.6.14), first make a binary matrix B is constructed as in Step (4) of the algorithm, where a single column of the matrix corresponds to an (a, b) pair found in above Step (Sieving step) with a+bm and a+bθ smooth. Since there are 10 primes in the rational factor base, 23 first degree prime ideals in the algebraic factor base, and 5 first degree prime ideals in the quadratic

character base, each column of the matrix will have 39 entries, or 39 total rows for the matrix (one bit is added for the sign of a+bm).
To show explicitly how the matrix is formed, the column entry pair (-8, 3) found in above step will be calculated. The first entry is set to 0 since a+bm = -8+3·31 = 85 is positive. The next 10 entry in this column vector are determined from the factorization of a+bm = 85 is positive. The next 10 entries in this column vector are determined from the factorization of a+bm = 85 over the rational factor base :

$$85 = 2^0 \cdot 3^0 \cdot 5^1 \cdot 7^0 \cdot 11^0 \cdot 13^0 \cdot 17^1 \cdot 19^0 \cdot 23^0 \cdot 29^0$$

where all the primes in the rational factor base have been shown for clarity. The column vector for (-8, 3) then has 10 entries formed by taking the above 10 exponent vectors modulo 2 :

$$(0, 0, 1, 0, 0, 0, 1, 0, 0, 0)$$

Next, the norm of (-8, 3) is computed and factored over the primes occurring in first degree prime ideal pairs in the algebraic factor base. Recalling from example 3.6.18 that N(a+bθ) = (-b)^{d}f(-a/b), it follows that the norm of an element a+bθ with d = 3 and f(x) = x^3 + 15x^2 + 29x + 8 can be computed as

$$N(a + b\theta) = (-b)^3 \cdot (\frac{-a^3}{b^3} + 15\frac{a^2}{b^2} - 29\frac{a}{b} + 8) = a^3 - 15a^2b + 29ab^2 - 8b^3$$

The norm of -8+3θ is then N(-8+3θ) = -8^3 - 15·(-8)^2·3-29·8·3^2-8·3^3 = -5696 = -1·2^6 89^1 gives the factorization of that norm over the primes p occurring in the (r, p) pairs of the algebraic factor base.
Note that there can be up to d pairs (r, p) in the algebraic factor base that share the same prime p, but only one such pair can have a ≡ -br (mod p). Such an (r, p) pair is the one that will be "responsible" for counting the number of times p divides N(a+bθ). In the case for -8+3θ, there are three first degree prime ideals in table (5.2) that have 89 as the prime in their pair representation, specifically (28, 89), (62, 89) and (73, 89). But -8 ≡ 81 ≡ 3 · 62 (mod 89) so the first degree prime ideal pair (62, 89) is responsible for the exponent of 89. Combining this with the first degree prime ideal having pair (0, 2) yields the next 23 bits in the column vector for (-8, 3) :

$$(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0)$$

Now, to compute a quadratic character for -8+3θ, corresponding to the first degree

prime ideal represented by the pair (s, q) that the Legendre symbol $(\frac{-8+3s}{q})$ *must be calculated. Using (80, 107) from the quadratic character base as an example yields :*

$$(\frac{-8+3\cdot 80}{107}) = -1$$

In this case, the vector coordinate for (-80, 107) is stored as 1 and would have been stored as 0 had the Legendre symbol been 1.
Performing the same operations for the remainder of the quadratic character base yields the final 5 bits in the column for (-8, 3) :

$$(1, 0, 0, 1, 0)$$

The complete 39-bit column vector for (-8, 3) then is seen to be :

$$(0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,1,0)^T$$

This same procedure is used on the rest (a, b) pairs found in above step to produce the 39×40 binary matrix B.
(a, b) pairs occurring in a dependency listed below in the following table :

Table (5.5)

(a, b) pair	*(a, b) pair*	*(a, b) pair*	*(a, b) pair*	*(a, b) pair*
(-1, 1)	*(104, 1)*	*(-8, 3)*	*(-43, 6)*	*(856, 11)*
(3, 1)	*(3, 2)*	*(48, 5)*	*(-8, 7)*	
(13, 1)	*(25, 2)*	*(54, 5)*	*(11, 7)*	

Step (5) : *(Square root and factor of n)*
Computing an explicit square root in $\mathbb{Z}[\theta]$:
Now. we have set U. As we have discussed in step (5) of the algorithm :

$$x^2 \equiv \phi(\beta^2) \equiv \phi(f'(\theta)^2 \prod_{(a,b)\in U}(a+b\theta)) \equiv \phi(f'(\theta))^2 \prod_{(a,b)\in U}\phi(a+b\theta)$$

$$\equiv f'(m)^2 \prod_{(a,b)\in U}(a+bm) \equiv y^2 (mod n)$$

Firstly we compute that f'(θ)² = 138·θ² + 363·θ + 481 ∈ ℤ[θ].
and we see that :

$\prod_{(a,b)\in U}(a + b\theta) = (-1 + \theta)\cdot(3 + \theta)\cdot(13 + \theta)\cdot(104 + \theta)\cdot(3 + 2\theta)\cdot(25 + 2\theta)\cdot(-8 + 3\theta)\cdot(48 + 5\theta)\cdot(54 + 5\theta)\cdot(-43 + 6\theta)\cdot(-8 + 7\theta)\cdot(11 + 7\theta)\cdot(856 + 11\theta)$
$= 20515431297644 85\cdot\theta^2 + 15388377355799440\cdot\theta + 24765692886531904$
where all the computations are treated as multiplication of polynomials modulo $f(x) = x^3+15x^2+29x+8$ *with* θ *substituted for* x.
So, $f'(\theta)^2 \cdot \prod_{(a,b)\in U}(a+b\theta) = 22455983949710645412\cdot\theta^2 + 541001057855125624 27\cdot\theta + 229394026576830712 24$, *which is the square of an element* $\gamma \in \mathbb{Z}[\theta]$.
Indeed without too much more effort it is seen that :
$\gamma = 599923511\cdot\theta^2 + 3686043120\cdot\theta + 3889976768$ *satisfies that* $\gamma^2 \in \mathbb{Z}[\theta]$.
Now applying the ring homomorphism $\phi : \mathbb{Z}[\theta] \longrightarrow \mathbb{Z}/45113\mathbb{Z}$ *(which we have already discussed), to* γ *gives :*

$$\begin{aligned} x = 43922 &\equiv 694683807559 \\ &\equiv 599923511 \cdot 31^2 + 3686043120 \cdot 31 + 3889976768 \\ &\equiv \phi(\gamma)(mod 45331) \end{aligned}$$

Computing the square root in finite field :
$y^2 = f'(m)^2 \prod_{(a,b)\in U}(a + bm) = (138\cdot(31)^2 + 363\cdot(31) + 481)\cdot(-1 + 31)\cdot(3 + 31)\cdot(13 + 31)\cdot(104 + 31)\cdot(3 + 2\cdot31)\cdot(25 + 2\cdot31)\cdot(-8 + 3\cdot31)\cdot(48 + 5\cdot31)\cdot(54 + 5\cdot31)\cdot(-43 + 6\cdot31)\cdot(-8 + 7\cdot31)\cdot(11 + 7\cdot31)\cdot(856 + 11\cdot31) = (3824)^2\cdot(100784047740239128260996 0000)$
$= (3824)^2\cdot(31746503388600)^2$
So, $y = 15160 \equiv 3824\cdot31746503388600 \pmod{45117}$
Thus, we get, $15160^2 \equiv 43922^2 \pmod{45113}$.
Therefore, gcd(15160 - 43922, 45113) = 197 and gcd(15160 + 43922, 45113) = 229 give nontrivial factors of n.

Chapter 4

Polynomial Factorization

4.1 Resultant and some useful properties

Definition 4.1.1. *Let f(x) = $a_n x^n + \cdots + a_0$ and g(x) = $b_m x^m + \cdots + b_0$ be two polynomials of degrees at most n and m respectively, with coefficients in an arbitrary field $\mathbb{F}$. Their resultant R(f, g) = $R_{n,m}$(f, g) is the element of $\mathbb{F}$ given by the determinant of the (m+n) × (m+n) matrix given by (Sylvester matrix) :*

$$M = Syl_{n,m}(f,g) = \begin{bmatrix} a_n & a_{n-1} & a_{n-2} & \cdots & 0 & 0 & 0 \\ 0 & a_n & a_{n-1} & \cdots & 0 & 0 & 0 \\ \cdot & \cdot & \cdot & & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdots & a_1 & a_0 & 0 \\ 0 & 0 & 0 & \cdots & a_2 & a_1 & a_0 \\ b_m & b_{m-1} & b_{m-2} & \cdots & 0 & 0 & 0 \\ 0 & b_m & b_{m-1} & \cdots & 0 & 0 & 0 \\ \cdot & \cdot & \cdot & & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdots & b_1 & b_0 & 0 \\ 0 & 0 & 0 & \cdots & b_2 & b_1 & b_0 \end{bmatrix}$$

where the m first rows contain the coefficients a_n, a_{n-1}, $\cdots$, a_0 of f shifted 0,1, $\cdots$, m-1 steps and padded with zeroes, and the n last rows contain the coefficients b_m, b_{m-1}, $\cdots$, b_0 of g shifted 0, 1, $\cdots$ n-1 steps and padded with zeroes. In other words, the entry at (i, j) equals a_{n+i-j} if $1 \leq i \leq m$ and b_{i-j} if $m+1 \leq i \leq m+n$, with $a_i = 0$ if $i > n$ or $i < 0$ and $b_i = 0$ if $i > m$ or $i < 0$.

Remark 4.1.2. *From the definition, we can observe that*

$$R_{n,m}(g,f) = (-1)^{nm} R_{n,m}(f,g) \tag{4.1}$$

Example 4.1.3. *If n = 3 and m =2,*

$$R(f,g) = det \begin{bmatrix} a_3 & a_2 & a_1 & a_0 & 0 \\ 0 & a_3 & a_2 & a_1 & a_0 \\ b_2 & b_1 & b_0 & 0 & 0 \\ 0 & b_2 & b_1 & b_0 & 0 \\ 0 & 0 & b_2 & b_1 & b_0 \end{bmatrix}$$

If we write $R_{n,m}$(f, g) as a polynomial with integer coefficients for any field with characteristic 0, such as $\mathbb{Q}$ or $\mathbb{C}$, then the formula is valid (with the same coefficients) for every field $\mathbb{F}$ (because the coefficients are given by expanding the determinant of M and thus have a combinatorial interpretation independent of $\mathbb{F}$. For a field of characteristic $p \neq 0$, the coefficients may be reduced modulo p, so they are not unique in that case).

Theorem 4.1.4. *(1) : If deg(g) $\leq k \leq m$ then*

$$R_{n,m}(f,g) = a_n^{m-k} R_{n,k}(f,g) \tag{4.2}$$

(2) : If deg(f) $\leq k \leq n$, then

$$R_{n,m}(f,g) = (-1)^{(n-k)m} b_m^{n-k} R_{k,m}(f,g) \tag{4.3}$$

(3): Let f and g be polynomials with deg(f) $\leq n$ and deg(g) $\leq m$. If $n \geq m$ and h is any polynomial with deg(h) $\leq n$ - m, then

$$R_{n,m}(f + hg, h) = R_{n,m}(f,g) \tag{4.4}$$

Proof. (1) : Suppose deg(g) < m, so, $b_m = 0$. Then the first column of the Sylvester matrix M is 0 except for its first element a_n, and the submatrix of M = $Syl_{n,m}$(f,g) obtained by deleting the first row and column equals $Syl_{n,m-1}$(f,g). Hence, by expanding the determinant $R_{n,m}$(f,g) along the first column, $R_{n,m}(f,g) = a_n R_{n,m-1}(f,g)$.
The equation 4.2 follows for k = m, m-1, m-2, $\cdots$, 0 by backwards induction.

(2) : By equation 4.1 and part (1),
$R_{n,m}(f,g) = (-1)^{nm}R_{m,n}(g,f) = (-1)^{nm}b_m^{n-k}R_{m,k}(g,f) = (-1)^{nm-km}b_m^{n-k}R_{k,m}(f,g)$.

(3) : Given that, $n \geq m$ and $\deg(h) \leq n - m$. Then $Syl_{n,m}(f+hg, g)$ is obtained from $Syl_{n,m}(f,g)$ by row operations that do not change its determinant $R_{n,m}$. (If $h(x) = c_r x^r + \cdots + c_0$, add c_k times row n+i-k to row i, for $i = 1, 2, \cdots, m$ and $k = 0, 1, \cdots, r$.) □

The main importance of the resultant lies in the following formula, which often is taken as the definition :

Theorem 4.1.5. *Let $f(x) = a_n x^n + \cdots + a_0$ and $g(x) = b_m x^m + \cdots + b_0$ be two polynomials of degrees n and m, respectively, with coefficients in an arbitrary field F. Suppose that, in some extension of F, f has n roots (with counting multiplicities) $\alpha_1, \cdots, \alpha_n$ and g has m roots (with counting multiplicities) $\beta_1, \cdots, \beta_m$ (not necessarily distinct). Then*

$$R(f,g) = a_n^m b_m^n \prod_{i=1}^{n}\prod_{j=1}^{m}(\alpha_i - \beta_j) \tag{4.5}$$

Proof. We will prove this theorem by induction on n+m.
Assume that $\deg(f) = n$ and $\deg(g) = m$.
Then at least in some extension field, $f(x) = a_n \prod_{i=1}^{n}(x-\alpha_i)$ and $g(x) = b_n \prod_{j=1}^{m}(x-\beta_j)$.
If n+m = 0 (n = 0, m = 0), then result (equation (4.5)) is obvious.
By induction, let equation 4.5 hold for all smaller values of n+m and all polynomials of these degrees.
Case (1) : Suppose $0 < n = \deg(f) \leq m = \deg(g)$. Divide g by f to obtain polynomials q and r with g = qf + r, where $\deg(r) < \deg(f) = n$. Note that $\deg(q) = \deg(qf) - \deg(f) = \deg(g-r) - n = m - n$.
By previous theorem part (3), we know that

$$R_{n,m}(f,g) = R_{n,m}(f, g-qf) = R_{n,m}(f,r) \tag{4.6}$$

Case 1(a) : Let $r \neq 0$ and let $k = \deg(r) \geq 0$. By previous theorem part (1), we get $R_{n,m}(f, r) = a_n^{m-k}R_{n,k}(f, r)$ and using induction hypothesis, we get the desired result.
Case 1(b) : Suppose that r = 0, so g = qf, but $n > 0$. Then $Syl_{n,m}(f, r) = Syl_{n,m}(f, 0)$ has the last n rows identically zero, so $R_{n,m}(f, r) = 0$, and so by equation (4.6), $R_{n,m}(f, g) = 0$ and we get the result.

Case (2) : Suppose that $n = 0$, Then $R_{0,m}(f, g) = a_0^m$, which agrees with our result. (This includes the case when n=0 and m=0, which starts the induction hypothesis).
Case(3) : Suppose m = deg(g) $\leq$ deg(f). Then we can interchange f and g in case (1) and (2).
This completes the induction hypothesis . □

Important properties of Resultant : *If we use theorem 4.1.5, as the definition of Resultant, then it is clear that : R(f,g) = 0 if and only if f and g have a common root, that is, if and only if f and g have nontrivial g.c.d .*
Hence we get following two useful properties :
(1) R(f,g) = 0 if and only if f and g have a common root.
(2) R(f,g) = 0 if and only if gcd(f,g) $\neq$ 1.

4.2 Discriminant of function :

Let $\mathbb{F}$ be an arbitrary field, and f(x) $\in \mathbb{F}[x]$ a polynomial of degree n $\geq$ 1 and with coefficients a_n. Let $\alpha_1, \alpha_2, \cdots, \alpha_n$ be the roots of f(x) (in some extension of $\mathbb{F}$). The discriminant of f(x) is defined as
$Disc(f(x)) = a_n^{2n-2} \prod_{i=1}^{n} \prod_{j=i+1}^{n} (\alpha_i - \alpha_j)^2 = (-1)^{n(n-1)/2} a_n^{2n-2} \prod_{i=1}^{n} \prod_{j=1, j\neq i}^{n} (\alpha_i - \alpha_j)$.

Lemma 4.2.1. *Let f(x) be as in above definition, f'(x) is the derivative of f, and let deg(f'(x)) = m, Then*
$Disc(f(x)) = (-1)^{n(n-1)/2} a_n^{n-m-2} Res(f(x), f'(x))$.
If m = n-1 (If the characteristic of $\mathbb{F}$ is zero), we have
$Disc(f(x)) = (-1)^{n(n-1)/2} a_n^{-1} Res(f(x), f'(x))$.
Hence, For any field $\mathbb{F}$, Disc(f(x)) $\neq$ 0 if and only if f(x) is square free.

Proof. We can easily see this by using theorem (4.1.5). □

Hensal lifting lemma

Theorem 4.2.2. *Let f be a monic polynomial in $\mathbb{Z}[X]$ and (h mod p^{i-1}) is a irreducible factor of (f mod p^{i-1}) for some integer i $\geq$ 2. Then there exists a polynomial h_1 (uniquely up to mod p^i) such that (h_1 mod p^i) divides (f mod p^i) and (h_1 mod p^{i-1}) = (h mod p^{i-1}).*

Proof. Since (h mod p^{i-1}) divides (f mod p^{i-1}), so, there exists a poynomial g(x) $\in \mathbb{Z}[X]$ such that (f mod p^{i-1}) = (h mod p^{i-1})(g mod p^{i-1}). Moreover, (h_1 mod p^{i-1}) has to be equal to (h mod p^{i-1}), so, h_1 = h + up^{i-1} for some u $\in \mathbb{Z}[X]$ with deg(u) $\leq$ deg(h).
Since (h_1 mod p^i) has to divide (f mod p^i), so, there exists a $g_1 \in \mathbb{Z}[X]$ such that (f mod p^i) = (h_1 mod p^i)(g_1 mod p^i). Set g_1 = g + vp^{i-1} for an appropriate choice of v $\in \mathbb{Z}[X]$ with deg(v) $\leq$ deg(g). Now we have :
(f mod p^i) = (h + up^{i-1})(g+vp^{i-1}) mod p^i
$\Rightarrow$ (f mod p^i) = (hg + (ug + vh)p^{i-1} + uvp^{2i-2}) mod p^i
$\Rightarrow \frac{f}{p^{i-1}}$ mod $p^i = \frac{hg}{p^{i-1}}$ + (ug + vh) mod p^i
$\Rightarrow (\frac{f-hg}{p^{i-1}})$ mod p^i = (ug + vh) mod p^i

Since we had f = hg mod p^{i-1}, so, f = hg + cp^{i-1} where c $\in \mathbb{Z}$, therefore f - hg = cp^{i-1} and $\frac{(f-hg)}{p^{i-1}}$ = c. So, we have :
(c mod p^i) mod p = ((ug + vh) mod p^i) mod p
$\Rightarrow$ c mod p = (ug + vh) mod p.
Hence there exists unique u and v, so h_1 and g_1. □

4.3 Lattice and reduced basis

Definition 4.3.1. *A subset L of the real vector space $\mathbb{R}^n$ is called a lattice if there exist a basis $b_1, b_2, \cdots, b_n$ of $\mathbb{R}^n$ such that :*

$$L = \{\sum_{i=1}^{n} s_i b_i | where\, s_i \in \mathbb{Z}\, for\, i \in \{1, 2, \cdots, n\}\} \tag{4.7}$$

We call $b_1, b_2, \cdots, b_n$: a basis for L and n to be the rank of L.
Moreover we define d(L) = $|det(b_1, b_2, \cdots, b_n)|$ to be the determinant of the lattice.

4.3.1 Gram-Schmidt orthogonalization

Any basis B can be transformed in to an orthogonal basis for the same vector space using the well-known Gram-Schmidt orthogonalization method.

Theorem 4.3.2. *Let $b_1, b_2, \cdots, b_n$ be some linear independent vectors in $\mathbb{R}^n$. Define*
$b_1^* = b_1$
$b_i^* = b_i$ - $\sum_{j=1}^{i-1} \mu_{i,j} b_j^*$ *for* $1 \leq i \leq n$.

where $\mu_{i,j} = \frac{(b_i, b_j^*)}{|b_j^*|^2}$ *for* $1 \leq j < i \leq n$ *and* (,) *is inner product.*
Then b_1^*, b_2^*, $\cdots$, b_n^* *are the orthogonal basis of* $\mathbb{R}^n$.

Proof. For proving our result, we have to show that b_1^*, b_2^*, $\cdots$, b_n^* orthogonal and they span $\mathbb{R}^n$, if b_1, $\cdots$, b_n span $\mathbb{R}^n$.
We will prove this by induction on n.
If $n = 1$, then $b_1^* = b_1$ and so, span(b_1) = span(b_1^*). Hence for n=1, result hold.
Now, assume that for $n \leq$ k, the result hold for the subspace of R^n, which means : all the new basis elements b_1^*, b_2^*, $\cdots$, b_k^* are orthogonal and span (b_1^*, b_2^*, $\cdots$, b_k^*) = span(b_1, b_2, $\cdots$, b_k).
Now for n = k+1. New element $b_{k+1}^* = b_{k+1}$ - $\sum_{j=1}^{k} \frac{(b_{k+1}, b_j^*)}{|b_j^*|^2} b_j^*$. If $1 \leq$ i $\leq$ k, then
(b_{k+1}^*, b_i^*) = (b_{k+1}, b_i^*) - $\sum_{j=1}^{k} \frac{(b_{k+1}, b_j^*)}{|b_j^*|^2}$($b_j^*$, b_i^*)
$\Rightarrow$ (b_{k+1}^*, b_i^*) = (b_{k+1}, b_i^*) - (b_{k+1}, b_i^*)
$\Rightarrow$ (b_{k+1}^*, b_i^*) = 0.
also,
$b_{k+1} = b_{k+1}^* + \sum_{j=1}^{k} \frac{(b_{k+1}, b_j^*)}{|b_j^*|^2} b_j^*$.
Hence span(b_1^*, b_2^*, $\cdots$, b_{k+1}^*) = span(b_1, b_2, $\cdots$, b_{k+1}) and all the elements b_1^*, b_2^*, $\cdots$, b_{k+1}^* are orthogonal. Therefore by induction hypothesis, our result holds. □

4.3.2 Reduced basis of Lattice

Definition 4.3.3. *A basis* b_1, b_2, $\cdots$, b_n *of a lattice L is called reduced if*

$$|\mu_{i,j}| \leq \frac{1}{2} \; for \; 1 \leq j < i \leq n \tag{4.8}$$

and

$$|b_i^* + \mu_{i,i-1} b_{i-1}^*|^2 \geq \frac{3}{4} |b_{i-1}^*|^2 \; for \; 1 \leq i \leq n \tag{4.9}$$

holds.

We can also write second condition as $|b_i^*|^2 \geq (\frac{3}{4}\text{-}\mu_{i,i-1}^2)|b_{i-1}^*|^2$ *(since* b_i^* *are orthogonal), which is known as the Lovasz's condition.*

Remark 4.3.4. *The constant* $\frac{3}{4}$ *in the definition is arbitrary chosen. Indeed, we could take any other constant between* $\frac{1}{4}$ *and 1.*

4.3.3 Procedure to find a reduced basis

Idea of the Process : *In this section, we will discuss an algorithm in which we will give as input any arbitrary basis of a lattice and we will get a reduced basis of the same lattice. From the definition of the reduced basis, we know that for finding reduced basis, first we need an orthogonal basis. So, firstly, we will use Gram-Schmidt orthogonalization process to given basis and after that we will modify these basis elements such that they fulfil the desired condition, which is given in equation (4.8) and (4.9).*

- *Let* $\{b_1, b_2, b_3, \cdots, b_n\}$ *be an arbitrary basis of any Lattice L. Now, we will describe the full procedure for finding reduced basis in the following steps :*

Steps for finding reduced basis :

Step (1) : (Orthogonalization)

In this step, we will compute the orthogonal basis $\{b_1^*, b_2^*, b_3^*, \cdots, b_n^*\}$ *and* $\mu_{i,j}$ $(1 \leq j < i \leq n)$ *for the given any arbitrary basis* $\{b_1, b_2, b_3, \cdots, b_n\}$ *using Gram-Schmidt orthogonalization process which we have discussed in theorem 4.3.2.*

- *Now, we have to modify this orthogonal basis in a reduced basis, means we need to modify this basis such that it satisfies the equation (4.8) and (4.9). For this we will go by an iteration process. Therefore, we will assume that the given basis is already reduced for* $\{b_1, b_2, b_3, \cdots, b_{k-1}\}$ *for some* $k<n$*, which means that orthogonal set of this set satisfies equation (4.8) and (4.9). Now our aim is to modify* b_k *(and if necessary then* $b_1, b_2, b_3, \cdots, b_{k-1}$*) in such a way that we can get a new set* $\{b_1', b_2', b_3', \cdots, b_k'\}$ *which fulfil all the conditions of the reduced basis such that* b_i^**'s and* $\mu_{i,j}$ *satisfies equation (4.8) and (4.9).*

% Step (2) : %

Let k = 2

While k $\leq n$ **% (Loop (1)) %**

{

% Step (3) % :

If $|\mu_{k,k-1}| > \frac{1}{2}$ (Checking condition (4.8) (Condition 1)

→ If $|\mu_{k,k-1}| > \frac{1}{2}$, then, for getting condition (4.8), for i= k and j = k-1, we proceeds in the following way :

Let define an element t which is nearest to $\mu_{k,k-1}$. Now, set $b_k^{'} = b_k$ - tb_{k-1}. We can see that after changing b_k, we now have to modify all the $\mu_{k,j}$, $1 \leq j < k$ as well.

Since we know that $\mu_{k,j} = \frac{(b_k, b_j^*)}{|b_j^*|^2}$.

Hence new $\mu_{k,j}^{'} = \frac{(b_k - tb_{k-1}, b_j^*)}{|b_j^*|^2} = \frac{(b_k, b_j^*)}{|b_j^*|^2}$ - $t\frac{(b_{k-1}, b_j^*)}{|b_j^*|^2}$

$\Rightarrow \mu_{k,j}^{'} = \mu_{k,j}$ - $t\mu_{k-1,j}$ for $1 \leq j <$ k-1 and $\mu_{k,k-1}^{'} = \mu_{k,k-1}$ - t

Thus, we can see if we define t this way then we obtain that $|\mu_{k,k-1}| \leq \frac{1}{2}$.

Now we will see that when we modify b_k in to $b_k^{'}$ for changing $|\mu_{k,k-1}| > \frac{1}{2}$ in to $|\mu_{k,k-1}| \leq \frac{1}{2}$, then there will be no effect on b_i^*'s for $i \in \{1, 2, \cdots, n\}$. For this, we know that

$b_i^* = b_i$ - $\sum_{j=1}^{i-1} \mu_{i,j} b_j^*$

Since, b_k has been modified only, so, we have to check only for b_k^* and b_i^* for $i \in$ [k+1,$\cdots$, n].

$$\begin{aligned}
b_k^{'*} &= b_k^{'} - \sum_{j=1}^{k-1} \mu_{k,j}^{'} b_j^{'*} \\
&= b_k - tb_{k-1} - \sum_{j=1}^{k-1} (\mu_{k,j} - t\mu_{k-1,j}) b_j^* \\
&= b_k - \sum_{j=1}^{k-1} \mu_{k,j} b_j^* - t(b_{k-1} - \sum_{j=1}^{k-1} \mu_{k-1,j} b_j^*) \\
&= b_k^* - t(b_{k-1}^* - \mu_{k-1,k-1} b_{k-1}^*) \\
&= b_k^*.
\end{aligned}$$

Thus we have following result :

$b_i^* = b_i$ for all i.

$\mu_{j,k}^{'} = \frac{(b_j', b_k^{'*})}{|b_k^*|^2} = \frac{(b_j, b_k^*)}{|b_k^*|^2} = \mu_{j,k}$ for j>k.

$|\mu_{k,k-1}| \leq \frac{1}{2}$

Remark : It is still not guaranteed that $|\mu_{k,j}| \leq \frac{1}{2}$ for $j <$ k-1.

% Step (4) %

If $|b_k^* + \mu_{k,k-1} b_{k-1}^*|^2 < \frac{3}{4}|b_{k-1}^*|^2$

{

If above condition holds then for getting condition (4.9) for $i = k$, we follow the below procedure :

Interchange b_k and b_{k-1}, i.e.,

Set b'$_k$ = b_{k-1} and b'$_{k-1}$ = b_k

Then :

$$\begin{aligned} b_{k-1}^{'*} &= b_{k-1}' - \sum_{j=1}^{k-2} \mu_{k-1,j}' b_j^{'*} \text{ (Since } b_j^{'*} = b_j^* \text{ for } 1 \le j \le \text{k-2)} \\ &= b_k - \sum_{j=1}^{k-2} \mu_{k,j} b_j^* \\ &= b_k - \sum_{j=1}^{k-1} \mu_{k,j} b_j^* + \mu_{k,k-1} b_{k-1}^* \\ b_{k-1}'^* &= b_k^* + \mu_{k,k-1} b_{k-1}^* \end{aligned} \tag{4.10}$$

Similarly,

$$\begin{aligned} b_k^{'*} &= b_k' - \sum_{j=1}^{k-1} \mu_{k,j}' b_j^{'*} \\ &= b_{k-1} - \sum_{j=1}^{k-2} \mu_{k-1,j} b_j^{'*} - \mu_{k,k-1}' b_{k-1}^{'*} \text{(Since } b_j'^* = b_j^* \text{ for } 1 \le j \le \text{k-2)} \\ &= b_{k-1} - \sum_{j=1}^{k-2} \mu_{k-1,j} b_j^* - \mu_{k,k-1}' b_{k-1}^{'*} \\ b_k'^* &= b_{k-1}^* - \mu_{k,k-1}' b_{k-1}'^* \\ b_k^{'*} &= b_{k-1}^* - \frac{(b_k', b_{k-1}^{'*})}{|b_{k-1}^{'*}|^2} b_{k-1}^{'*} \\ &= b_{k-1}^* - \frac{(b_{k-1}, b_k^* + \mu_{k,k-1} b_{k-1}^*)}{|b_{k-1}'^*|^2} b_{k-1}^{'*} \end{aligned} \tag{4.11}$$

$$b'^{*}_{k} = b^{*}_{k-1} - \frac{(b^{*}_{k-1} + \sum_{i=1}^{k-2} \mu_{k-1,i} b^{*}_{i}, b^{*}_{k} + \mu_{k,k-1} b^{*}_{k-1})}{|b'^{*}_{k-1}|^2} b'^{*}_{k-1}$$

$$= b^{*}_{k-1} - \frac{(b^{*}_{k-1}, \mu_{k,k-1} b^{*}_{k-1})}{|b'^{*}_{k-1}|^2} b'^{*}_{k-1}$$

$$b'^{*}_{k} = b^{*}_{k-1} - \mu_{k,k-1} \frac{|b^{*}_{k-1}|^2}{|b'^{*}_{k-1}|^2} b'^{*}_{k-1} \tag{4.12}$$

Now, Comparing equation (4.11) and (4.12) :

$$\mu'_{k,k-1} = \mu_{k,k-1} \frac{|b^{*}_{k-1}|^2}{|b'^{*}_{k-1}|^2} \tag{4.13}$$

Thus, after interchanging b_k and b_{k-1}, we have following result :

$b'^{*}_{j} = b^{*}_{j}$ *for* $j \neq k$, *k-1*

$b'^{*}_{k-1} = b^{*}_{k} + \mu_{k,k-1} b^{*}_{k-1}$

$b'^{*}_{k} = b^{*}_{k-1}$ - $\mu'_{k,k-1} b'^{*}_{k-1}$

$\mu'_{k,k-1} = \mu_{k,k-1} \frac{|b^{*}_{k-1}|^2}{|b'^{*}_{k-1}|^2}$

So, now we have to check the effect of interchanging the b_k and b_{k-1} on the following two things :

(1) $\mu_{i,k-1}$ and $\mu_{i,k}$ for $i > k$.

(2) $\mu_{k-1,i}$ and $\mu_{k,i}$ for $1 \leq i <$ k-1.

Case (1) : Since we know that $b_i = b'_i$ for $i > k$, Hence :

$b_i = b^{*}_{i} + \sum_{j=1}^{i-1} \mu_{i,j} b^{*}_{j} = b'^{*}_{j} + \sum_{j=1}^{i-1} \mu'_{i,j} b'^{*}_{j} = b'_i$

Since we know that $b'^{*}_{j} = b^{*}_{j}$ for $j \neq k$, k-1, So, we can remove the common terms on both sides of the equation by subtraction to get :

$$\mu_{i,k-1} b^{*}_{k-1} + \mu_{i,k} b^{*}_{k} = \mu'_{i,k-1} b'^{*}_{k-1} + \mu'_{i,k} b'^{*}_{k} \tag{4.14}$$

and from equation (4.11), we have :

$$b^{*}_{k-1} = b'^{*}_{k} + \mu'_{k,k-1} b'^{*}_{k-1}$$

and from equation (4.10)*, we have :*

$$b^{*}_{k} = b'^{*}_{k-1} - \mu_{k-1} b^{*}_{k-1}$$

$$= b'^{*}_{k-1} - \mu_{k-1}(b'^{*}_{k} + \mu'_{k,k-1} b'^{*}_{k-1})$$

$$= (1 - \mu_{k,k-1} \mu'_{k,k-1}) b'^{*}_{k-1} - \mu_{k,k-1} b'^{*}_{k}$$

Now, using equation (4.13), we get :

$$\begin{aligned} b_k^* &= (1 - \mu_{k,k-1}^2 \frac{|b_{k-1}^*|^2}{|b_{k-1}'^*|^2}) b_{k-1}'^* - \mu_{k,k-1} b_k'^* \\ &= (1 - \frac{(b_k, b_{k-1}^*)^2}{|b_{k-1}^*|^2 |b_{k-1}'^*|^2}) b_{k-1}'^* - \mu_{k,k-1} b_k'^* \\ &= (1 - \frac{(b_k^* + \sum_{i=1}^{k-1} \mu_{k,i} b_i^*, b_{k-1}^*)^2}{|b_{k-1}^*|^2 |b_{k-1}'^*|^2}) b_{k-1}'^* - \mu_{k,k-1} b_k'^* \end{aligned}$$

Using orthogonality of b_i^*, we can write :

$$\begin{aligned} &= (1 - \frac{(\mu_{k,k-1} b_{k-1}^*, b_{k-1}^*)^2}{|b_{k-1}^*|^2 |b_{k-1}'^*|^2}) b_{k-1}'^* - \mu_{k,k-1} b_k'^* \\ &= (1 - \frac{|b_{k-1}^*|^4 |\mu_{k,k-1}^2|}{|b_{k-1}^*|^2 |b_{k-1}'^*|^2}) b_{k-1}'^* - \mu_{k,k-1} b_k'^* \\ &= (\frac{|b_{k-1}'^*|^2 - \mu_{k,k-1}^2 |b_{k-1}^*|^2}{|b_{k-1}'^*|^2}) b_{k-1}'^* - \mu_{k,k-1} b_k'^* \end{aligned}$$

Now using equation (4.10)*, we get :*

$$\begin{aligned} b_k^* &= (\frac{|b_k^* + \mu_{k,k-1} b_{k-1}^*|^2 - \mu_{k,k-1}^2 |b_{k-1}^*|^2}{|b_{k-1}'^*|^2}) b_{k-1}'^* - \mu_{k,k-1} b_k'^* \\ &= (\frac{|b_k^*|^2 + \mu_{k,k-1}^2 |b_{k-1}^*|^2 - \mu_{k,k-1}^2 |b_{k-1}^*|^2}{|b_{k-1}'^*|^2}) b_{k-1}'^* - \mu_{k,k-1} b_k'^*. \end{aligned}$$

Hence, we have :

$$b_k^* = \frac{|b_k^*|^2}{b_{k-1}'^*} b_{k-1}'^* - \mu_{k,k-1} b_k'^* \tag{4.15}$$

By putting the value of b_k^* (from the above equation) and b_{k-1}^* (from equation (4.11)), we get :

$\mu_{i,k-1}(b_k'^* + \mu'_{k,k-1} b_{k-1}'^*) + \mu_{i,k}(\frac{|b_k^*|^2}{b_{k-1}'^*} b_{k-1}'^* - \mu_{k,k-1} b_k'^*) = \mu'_{i,k-1} b_{k-1}'^* + \mu'_{i,k} b_k'^*$

$\Rightarrow (\mu_{i,k-1} - \mu_{i,k}\mu_{k,k-1}) b_k'^* + (\mu_{i,k-1}\mu'_{k,k-1} + \mu_{i,k}\frac{|b_k^*|^2}{|b_{k-1}^*|^2}) b_{k-1}'^* = \mu'_{i,k-1} b_{k-1}'^* + \mu'_{i,k} b_k'^*$

Comparing the terms of $b_k'^*$ and $b_{k-1}'^*$ both sides, we get the following result :

$$\mu'_{i,k-1} = \mu_{i,k-1}\mu'_{k,k-1} + \mu_{i,k}\frac{|b_k^*|^2}{|b_{k-1}^*|^2} \;\; and \;\; \mu'_{i,k} = \mu_{i,k-1} - \mu_{i,k}\mu_{k,k-1} \;\; (for \; i > k) \tag{4.16}$$

Case (2) : $\mu'_{k-1,j} = \frac{(b'_{k-1}, b_j'^*)}{|b_j'^*|^2}$

Since we know that $b'_{k-1} = b_k$ *and* $b_j'^* = b_j^*$ *for* $j \neq k$, *k-1. Hence :*

$\mu'_{k-1,j} = \frac{(b_k, b_j^*)}{|b_j^*|^2} = \mu_{k,j}$.
Similarly, $\mu'_{k,j} = \frac{(b'_k, b_j^{'*})}{|b_j^{'*}|^2} = \frac{(b_{k-1}, b_j^*)}{|b_j^*|^2} = \mu_{k-1,j}$.
Therefore, In this condition, we have the following results :

$$
\begin{aligned}
b_j'^* &= b_j^* \text{ for } j \neq k, k-1 \\
b_{k-1}^{'*} &= b_k^* + \mu_{k,k-1} b_{k-1}^* \\
b_k^{'*} &= b_{k-1}^* - \mu'_{k,k-1} b_{k-1}^{'*} \\
\mu'_{k,k-1} &= \mu_{k,k-1} \frac{|b_{k-1}^*|^2}{|b_{k-1}'^*|^2} \\
\mu'_{i,k-1} &= \mu_{i,k-1}\mu'_{k,k-1} + \mu_{i,k} \frac{|b_k^*|^2}{|b_{k-1}^*|^2} \text{ for } i > k \\
\mu'_{i,k} &= \mu_{i,k-1} - \mu_{i,k}\mu_{k,k-1} \text{ for } i > k \\
\mu'_{k-1,j} &= \mu_{k,j} \text{ for } 1 \leq j < k-1 \\
\mu'_{k,j} &= \mu_{k-1,j} \text{ for } 1 \leq j < k-1
\end{aligned}
$$

And now after changing b_{k-1} *and* b_k*, we have also* $|b_k^* + \mu_{k,k-1} b_{k-1}^*|^2 \geq \frac{3}{4}|b_{k-1}^*|^2$.

If k $\geq$ 3, Set k = k-1

}

% Step (5) %

Else

{

For j = k-2, k-3, $\cdots$, 1

{

If ($|\mu_{k,j}| > \frac{1}{2}$) :

If this condition holds then for satisfying the condition (4.8), without changing any b_j^**, we follow below procedure :*
Let l be the largest index such that $|\mu_{k,l}| > \frac{1}{2}$ *for 1≤l<k-1.*
Then, let r be the nearest integer to $\mu_{k,l}$*. Now, set b'*$_k$ = b_k - rb_l.

Claim : (1) $\mu'_{k,j}$ = $\mu_{k,j}$ - r $\mu_{l,j}$ *for* $1 \leq j < l$
(2) $\mu'_{k,l} = \mu_{k,l}$ - r
(3) $\mu'_{k,j} = \mu_{k,j}$ *for* $l < j \leq k$
(4) $\mu'_{j,k} = \mu_{j,k}$ *for* $j > k$
(5) $b_i^{'*} = b_i^*$ *for all i.*
Proof : *(1) and (2) : Since* $b_j^{'*} = b_j^*$ *for* $j \leq l < k$ *and by definition,*

$$\begin{aligned}
\mu'_{k,j} &= \frac{(b'_k, b_j^{'*})}{|b_j^{'*}|^2} \\
&= \frac{(b_k - r b_l, b_j^*)}{|b_j^*|^2} \\
&= \frac{(b_k, b_j^*)}{|b_j^*|^2} - r\frac{(b_l, b_j^*)}{|b_j^*|^2} \text{ for } 1 \leq j \leq l < k \\
\Rightarrow \mu'_{k,j} &= \mu_{k,j} - r\mu_{l,j} \text{ for } 1 \leq j < l \\
\text{and for } j = l,\ \mu'_{k,l} &= \mu_{k,l} - r.
\end{aligned}$$

(3) and (4) : Since when j>k, we did not changed b'$_j = b_j$*, So,* $b_j^{'*} = b_j^*$.
For, j>k ; $\mu_{j,k} = \frac{(b'_j, b_k^{'*})}{|b_k^{'*}|^2} = \frac{(b_j, b_k^*)}{|b_k^*|^2} = \mu_{j,k}$
Similarly for l<j≤k,

$$\begin{aligned}
\mu'_{k,j} &= \frac{(b'_k, b_j^{'*})}{|b_j^{'*}|^2} \\
&= \frac{(b_k - r b_l, b_j^*)}{|b_j^*|^2}
\end{aligned}$$

Since from (5), $b_k^{'*} = b_k^*$.

$$\begin{aligned}
&= \frac{(b_k, b_j^*)}{|b_j^*|^2} - r\frac{(b_l, b_j^*)}{|b_j^*|^2} \\
&= \mu_{k,j} - r\frac{(b_l^* - \sum_{i=1}^{l-1} \mu_{l,i} b_i^*, b_j^*)}{|b_j^*|^2} \\
&= \mu_{k,j} - r(\frac{(b_l, b_j^*)}{|b_j^*|^2} + \sum_{i=1}^{l-1} \mu_{l,i} \frac{(b_i^*, b_j^*)}{|b_j^*|^2})
\end{aligned}$$

Using the orthogonality of all b_j^* *for j>l, we get :*

$$\mu'_{k,j} = \mu_{k,j}.$$

(5) : We can see that we need to check only for $b_k'^* = b_k^*$.

$$\begin{aligned} b_k'^* &= b_k' - \sum_{j=1}^{k-1} \mu_{k,j}' b_j'^* \\ &= b_k - rb_l - \sum_{j=1}^{k-1} (\mu_{k,j} - r\mu_{l,j}) b_j^* \text{ for } 1 \le j < l \le k-1. \\ &= b_k - \sum_{j=1}^{k-1} \mu_{k,j} b_j^* - r(b_l - \sum_{j=1}^{l} \mu_{l,j} b_j^*) \\ &= b_k^* - r(b_l - \sum_{j=1}^{l-1} \mu_{l,j} b_j^* - \mu_{l,l} b_l^*) \\ &= b_k^* - r(b_l^* - b_l^*) = b_k^*. \end{aligned}$$

We can also see from claim (2) that $|\mu_{k,l}| \le \frac{1}{2}$ *for 1≤l<k-1.*

}

Set k = k+1

}
}
Thus, we get reduced basis $b_1, b_2, \cdots, b_n$.

4.3.4 LLL Algorithm

(for getting reduced basis of a given basis $b_1, b_2, \cdots, b_n$ *of a Lattice L)*

Algorithm (1) : (for getting $\mathbf{b_1^*, b_2^*, \cdots, b_n^*}$**) :**

Input : $b_1, b_2, \cdots, b_n$
for j = 1,2,3,$\cdots$*,n*
Compute : $b_j^* = b_j - \sum_{i=1}^{j-1} \mu_{j,i} b_i^*$, *where* $\mu_{j,i} = (b_j, b_i^*)/(b_i^*, b_i^*)$.
Output : $b_1^*, b_2^*, \cdots, b_n^*$

Algorithm (2) : (for handling $|\mu_{k,l}| > \frac{1}{2}$, where 1≤l<k≤n)

Let r be the nearest integer to $\mu_{k,l}$.
Set b'$_k$ = b_k - rb_l
Update $\mu'_{k,j} = \mu_{k,j}$ - $r\mu_{l,j}$ *for j = 1,2,3,*$\cdots$, *l-1*
Update $\mu'_{k,l} = \mu_{k,l}$ - *r.*
Output : for $b_1, b_2, \cdots, b'_k, b_{k+1}, \cdots, b_n$, *we have* $|\mu'_{k,l}| \leq \frac{1}{2}$

Algorithm (3) : (for handling $|b^*_k + \mu_{k,k-1}b^*_{k-1}|^2 < \frac{3}{4}|b^*_{k-1}|^2$)

Set b'$_k$ = b_{k-1}
Set b'$_{k-1}$ = b_k
Update $b'^*_k = b^*_{k-1}$ - $\mu'_{k,k-1}b'^*_{k-1}$
Update $b'^*_{k-1} = b^*_k + \mu_{k,k-1}b^*_{k-1}$
Update $\mu'_{k,k-1} = \mu_{k,k-1}\frac{|b^*_{k-1}|^2}{|b'^*_{k-1}|^2}$
Update $\mu'_{i,k-1} = \mu_{i,k-1}\mu'_{k,k-1} + \mu_{i,k}\frac{|b^*_k|^2}{|b^*_{k-1}|^2}$ *for i>k.*
Update $\mu'_{i,k} = \mu_{i,k-1}$ - $\mu_{i,k}\mu_{k,k-1}$ *for i>k*
Output : for $b_1, b_2, \cdots, b'_{k-1}, b'_k, b_{k+1}, \cdots, b_n$, *we have :* $|b'^*_k + \mu'_{k,k-1}b'^*_{k-1}|^2 \geq \frac{3}{4}|b'^*_{k-1}|^2$

(LLL) Algorithm (4) : (for getting reduced basis)

Input : $b_1, b_2, \cdots, b_n$
Compute : $b^*_1, b^*_2, \cdots, b^*_k$ *(using algorithm (1)*
Set k = 2,
While (k ≤ 2) {
    If ($\mu_{k,k-1} > \frac{1}{2}$*), Call algorithm (2) with l = k-1.*
    If ($|b^*_k + \mu_{k,k-1}b^*_{k-1}|^2 < \frac{3}{4}|b^*_{k-1}|^2$*)* {
        Call algorithm (3).
        If k ≥ 3, Set k = k-1.
    }
    Else {
        For j = k-2,k-3, $\cdots$, *1* {

If ($|\mu_{k,j}| > \frac{1}{2}$), Call algorithm (2) with $l = j$.

}

Set k = k+1.

}

}

Output : Reduced basis b_1, b_2, $\cdots$, b_n.

Theorem 4.3.5. *The LLL algorithm which is described above terminates and it returns a reduced basis b'$_1$, b'$_2$, $\cdots$, b'$_n$.*

Proof. Firstly, we can easily see that it returns a reduced basis. The main loop (while loop) of algorithm (4) maintains the invariance on k, that condition (4.8) satisfied for $1 \leq$ j $<$ i $\leq$ k-1, and condition (4.9) is satisfied for $1 \leq$ i $\leq$ k-1. Each iteration of the loop attempts to enforce the conditions (4.8) and (4.9) at the current value of k.

First the condition on $\mu_{k,k-1}$ is checked, and if this condition doesn't hold then it is repaired by using algorithm (2) without any side effects.

Next, Condition (4.9) is checked for i = k. If this condition holds, then other $\mu_{k,j}$ values are handled, and we increased the current index of k by one, and the next iteration of the while loop is started. However, if Condition (4.9) does not hold for i = k, the vector b_{k-1} and consequently all $\mu_{k-1,j}$ values change in the process. Therefore k is decremented, and the next iteration of the while loop is started.

However, when k = 2, there are no valid values of the $\mu_{k-1,j}$, so, k is not decremented. When the while loop terminates, we have k = n+1, by the loop invariance, both the conditions (4.8) and (4.9) for a reduced basis are satisfied for all relevant values of i and j. Thus, we get a reduced basis of the given lattice.

Claim (1) : The algorithm terminates or there are finitely only finitely many times where we need to interchange b_{k-1} and b_k (because every time we interchanged b_{k-1} and b_k, we lowered the current index by one and in every other case we increased it by one.)

Proof of Claim (1) : Define : $d_i = \det(D_i)$ with $D_i = (b_j, b_k)_{1 \leq j,k \leq i}$.

Define $D = \prod_{i=1}^{n-1} d_i$

Now, we will compute the effect of the changes of the b_i's in the algorithm on the two quantities d_i and D.

Claim (2) : $d_i = \prod_{j=1}^{i} |b_j^*|^2$, for all i $\in \{1, 2, \cdots, n\}$.

Proof of claim (2) : We will show this by using Gaussian elimination procedure to D_i, because we know that determinant will not change under these operations.

Since

$$d_i = det(D_i) = det \begin{bmatrix} (b_1, b_1) & (b_1, b_2) & \cdots & (b_1, b_i) \\ (b_2, b_1) & (b_2, b_2) & \cdots & (b_2, b_i) \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ (b_i, b_1) & (b_i, b_2) & \cdots & (b_i, b_i) \end{bmatrix}$$

Now, the first step of Gaussian elimination method, we subtract $\frac{(b_j,b_1)}{(b_1,b_1)}$ times the first row from the j^{th} row, for all j $\in \{2, 3, \cdots, i\}$. After subtracting, we get zeroes on each entry of the first column except in the first row. The first row of the matrix will remain unchanged and the k^{th} column entry (for k = {2,3,$\cdots$,i}) of the j^{th} row will be :

(b_j,b_k) - $\frac{(b_j,b_1)}{(b_1,b_1)}(b_1, b_k) = (b_j - \mu_{j1}b_1^*, b_k)$ (Since $b_1^* = b_1$)

Since $b_2 - \mu_{2,1}b_1^* = b_2^*$, hence for k=2 and j=2, we get $(b_j^*,b_j) = (b_j^*, b_j^* + \sum_{i=1}^{j-1} \mu_{j,i}b_i^*)$ $= |b_j^*|^2$. Hence we get :

$$d_i = det(D_i) = det \begin{bmatrix} |b_1^*|^2 & (b_1^*, b_2) & (b_1^*, b_3) & \cdots & (b_1^*, b_i) \\ 0 & |b_2^*|^2 & (b_2^*, b_3) & \cdots & (b_2^*, b_i) \\ 0 & (b_3 - \mu_{3,1}b_1^*, b_2) & (b_3 - \mu_{3,1}b_1^*, b_3) & \cdots & b_3 - \mu_{3,1}b_1^*, b_i) \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ 0 & (b_i - \mu_{i,1}b_1^*, b_2) & (b_i - \mu_{i,1}b_1^*, b_3) & \cdots & (b_i - \mu_{i,1}b_1^*, b_i) \end{bmatrix}$$

Now, the second step of Gaussian elimination method, we subtract $\frac{(b_j - \mu_{j,1}b_1^*, b_2)}{|b_2^*|^2}$ times the second row from the j^{th} row, for all j $\in \{3, 4, \cdots, i\}$). After subtracting, we get zeroes on each entry of the second column except in the first and second row.

The k^{th} entry (k $\in \{3, 4, \cdots, i\}$) of the j^{th} row will be :

$$(b_j - \mu_{j,1}b_1^*, b_k) - \frac{(b_j - \mu_{j,1}b_1^*, b_2)}{|b_2^*|^2}(b_2^*, b_k) = (b_j - \mu_{j,1}b_1^* - \frac{(b_j - \mu_{j,1}b_1^*, b_2)}{|b_2^*|^2}b_2^*, b_k)$$

$$= (b_j - \mu_{j,1}b_1^*) - \frac{(b_j - \mu_{j,1}b_1^*, b_2^* + \mu_{2,1}b_1^*)}{|b_2^*|^2}(b_2^*, b_k)$$

$$= (b_j - \mu_{j,1}b_1^* - (\mu_{j,2} + \frac{(b_j, \mu_{2,1}b_1^*)}{|b_2^*|^2} - \frac{(\frac{(b_j,b_1^*)}{|b_1^*|^2}, \mu_{2,1}b_1^*)}{|b_2^*|^2})b_2^*, b_k)$$
$$= (b_j - \mu_{j,1}b_1^* - (\mu_{j,2} + \frac{(b_j, \mu_{2,1}b_1^*)}{|b_2^*|^2} - \frac{(b_j,b_1^*)\mu_{2,1}}{|b_2^*|^2})b_2^*, b_k)$$
$$= (b_j - \mu_{j,1}b_1^* - \mu_{j,2}b_2^*, b_k)$$

Hence we get :

$$d_i = det(D_i) = det \begin{bmatrix} |b_1^*|^2 & (b_1^*, b_2) & (b_1^*, b_3) & \cdots & (b_1^*, b_i) \\ 0 & |b_2^*|^2 & (b_2^*, b_3) & \cdots & (b_2^*, b_i) \\ 0 & 0 & |b_3^*|^2 & \cdots & b_3 - \mu_{3,1}b_1^*, b_i) \\ 0 & 0 & (b_4 - \mu_{4,1}b_1^* - \mu_{4,2}b_2^*, b_3) & \cdots & (b_4 - \mu_{4,1}b_1^* - \mu_{4,2}b_2^*, b_i) \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ 0 & 0 & (b_i - \mu_{i,1}b_1^* - \mu_{i,2}b_2^*, b_3) & \cdots & (b_i - (b_i - \mu_{i,1}b_1^* - \mu_{i,2}b_2^*, b_i) \end{bmatrix}$$

Thus, when we follow this procedure i times, then we get :

$$d_i = det(D_i) = det \begin{bmatrix} |b_1^*|^2 & (b_1^*, b_2) & (b_1^*, b_3) & \cdots & (b_1^*, b_i) \\ 0 & |b_2^*|^2 & (b_2^*, b_3) & \cdots & (b_2^*, b_i) \\ 0 & 0 & |b_3^*|^2 & \cdots & b_3 - \mu_{3,1}b_1^*, b_i) \\ 0 & 0 & 0 & \cdots & (b_4 - \mu_{4,1}b_1^* - \mu_{4,2}b_2^*, b_i) \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ 0 & 0 & 0 & \cdots & |b_i^*|^2 \end{bmatrix} = \prod_{j=1}^{i} |b_j^*|^2$$

Thus, we proved claim (2).

Now, we had to modify some of the b_i^*'s (namely b_k^* and b_{k-1}^*) when $|b_k^* + \mu_{k,k-1}b_{k-1}^*|^2 < \frac{3}{4}|b_{k-1}^*|^2$. In this case, we had $|b_{k-1}'^*|^2 = |b_k^* + \mu_{k,k-1}b_{k-1}^*|^2 < \frac{3}{4}|b_{k-1}^*|^2$. Thus, we can see that d_{k-1} is decreased by a factor of $\frac{3}{4}$ with every such change being made. We can also observe that all the other d_j for j $\in \{1, 2, \cdots, i\}$-{k-1} remains unchanged.
Since we decrement the value of the current index of the algorithm if and only if such a change is made, there is a one - one correspondence between the decremental of the current index and D. Thus, if after one modification of the b_i's, we define the product

of the determinants defined above by D', then we have D' $< \frac{3}{4}$D.
Now, we will show a positive lower bound for d_i depending only on L.
Claim (3) : There is a positive lower bound for d_i.
Proof of claim (3) : Firstly, we define m(L) = min$\{|x|^2;\ x \neq 0,\ x \in L\ \}$, which is a positive real number (Here we are not claiming that this algorithm finds such a shortest vector). Since we know by the definition of the determinant of the lattice that :
$d(L) = |\det(b_1, b_2, \cdots, b_n)| = |\det(b_1^*, b_2^*, \cdots, b_n^*)| = \prod_{i=1}^{n} |b_i^*|$
We know that b_i^*'s are pairwise orthogonal, so, $d_n = (d(L))^2$ and d_i (for $1 \leq i < n$) is equal to the determinant of the lattice spanned by the vectors $b_1, b_2, \cdots, b_i$. Since lattice have the property that there exists a vector $x \neq 0$ such that $|x| \leq \frac{4}{3}^{\frac{(i-1)}{2}} d_i^{\frac{1}{i}}$. Therefore
$d_i \geq \frac{3}{4}^{\frac{i(i-1)}{2}} |x|^{2i} \geq \frac{3}{4}^{\frac{i(i-1)}{2}} m(L)^i$, which is the lower bound for d_i, as we wanted. Thus we proved claim (3).

By the above discussion, we know that when we interchange b_{k-1} and b_k, then we lower D by a factor of $\frac{3}{4}$ every time, and there is a lower bound for D. Hence we conclude that there can be only finitely many interchanges of b_{k-1} and b_k during the algorithm and therefore algorithm terminates. Thus we proved claim (1) and hence theorem. □

4.4 Some important properties of reduced basis

Theorem 4.4.1. *If $b_1, b_2, \cdots, b_n$ is some reduced basis for a lattice L in $\mathbb{R}^n$, then we have following :*
(1) $|b_j|^2 \leq 2^{i-1}|b_i^*|^2$ for $1 \leq j \leq i \leq n$.
(2) $d(L) \leq \prod_{i=1}^{n} |b_i| \leq 2^{n(n-1)/4} d(L)$
(3) $|b_1| \leq 2^{(n-1)/4} d(L)^{\frac{1}{n}}$
(4) For any linearly independent set of vectors $x_1, x_2, \cdots, x_t \in L$, we have $|b_j| \geq 2^{(n-1)/2} max(|x_1|, |x_2|, \cdots, |x_t|)$ for $1 \leq j \leq t$.

Proof. (1) : We know that $|b_i|^2 = |b_i^* + \sum_{j=1}^{i-1} \mu_{i,j} b_j^*|^2$
Since all the b_i^* are orthogonal for $1 \leq i \leq n$ and $|\mu_{i,j}| \leq \frac{1}{2}$.

So, we have :

$$\begin{aligned}|b_i|^2 &= |b_i^*|^2 + \sum_{j=1}^{i-1} \mu_{i,i-1}^2 |b_j^*|^2 \\ &= |b_i^*|^2 + \mu_{i,1}^2 |b_1^*|^2 + \cdots + \mu_{i,i-1}^2 |b_{i-1}^*|^2 \\ &\leq |b_i^*|^2 + \frac{1}{4}|b_{i-1}^*|^2 + \cdots + \frac{1}{4}|b_1^*|^2\end{aligned}$$

By Lovasz's condition (which is described after equation (4.9)), we know that : $|b_j^*|^2 \leq 2^{i-j}|b_i^*|^2$ for all j $\leq$ i. Hence for all i, we get :

$$\begin{aligned}|b_i|^2 &\leq (1 + \frac{1}{4}(2 + 2^2 + \cdots + 2^{i-1}))|b_i^*|^2 \\ &= \frac{2^{i-1}+1}{2}|b_i^*|^2 \\ &\leq 2^{i-1}|b_i^*|^2.\end{aligned}$$

Thus we get : $|b_j^*|^2 \leq 2^{j-1}|b_j^*|^2$
Now again using Lovasz's condition,
$|b_i^*|^2 \leq 2^{i-j}2^{j-1}|b_i^*|^2 = 2^{i-1}|b_i^*|^2$ for $1 \leq$ j $\leq$ i $\leq$ n, which proves (1).

(2) : We now, d(L) = $\prod_{i=1}^{n} |\mathrm{b}_i^*|$
Since, $|b_i^*| \leq |b_i|$, so,
d(L) $\leq \prod_{i=1}^{n} |\mathrm{b}_i|$
Now, using part (1), we get :

$$\begin{aligned}d(L) &\leq \prod_{i=1}^{n} 2^{(i-1)/2}|b_i^*| \\ &= 2^{\frac{n(n-1)}{4}} \prod_{i=1}^{n} |b_i^*| \\ &= 2^{\frac{n(n-1)}{4}} d(L),\end{aligned}$$

which proves (2).

(3) : We know by Lovasz's condition that for j =1 that $|b_1^*|^2 \leq 2^{i-1}|b_i^*|^2$ and since

$|b_1^*| = |b_1|$, Hence we have :

$$\begin{aligned} |b_1|^n &\leq \prod_{i=1}^{n} 2^{(i-1)/2} |b_i^*| \\ &= 2^{\frac{n(n-1)}{4}} d(L), \text{ which implies :} \\ |b_1| &\leq 2^{\frac{(n-1)}{4}} d(L)^{\frac{1}{n}}, \end{aligned}$$

which proves (3).

(4) : If any $x_j \in L$ then we can write $x_j = \sum_{i=1}^{n} c_{i,j} b_i = \sum_{i=1}^{n} c'_{i,j} b_i^*$ with $c_{i,j} \in \mathbb{Z}$. Now, let $1 \leq j \leq n$ be fixed and k be the largest index such that $c_{k,j} \neq 0$.
Claim : $c_{k,j} = c'_{k,j}$.
Proof of the claim : Since $x_j = \sum_{i=1}^{k} c_{i,j} b_i = \sum_{i=1}^{k} c'_{i,j} (b_i - \sum_{i=1}^{i-1} \mu_{i,l} b_l^*)$
Since, all the b_i are independent and comparing the coefficient of b_k on both sides, we get $c_{k,j} = c'_{k,j}$.
Since all the b_i^* are pairwise orthogonal, hence we have : $|x_j| = \sum_{i=1}^{k} |c'_{i,j}||b_i^*| \geq |c'_{l,j}||b_l^*|$ for all $l \in \{1, \cdots, k\}$ and so,
$|x_j^2|^2 \geq |c'_{k,j}|^2 |b_k^*|^2 \geq |b_k^*|^2$ (since $c'_{k,l} = c_{k,l}$ which is an integer).
Now, using part (1), we have :

$$\begin{aligned} |b_i|^2 &\leq 2^{k-1} |b_k^*|^2 \\ &\leq 2^{k-1} |x_j|^2 \\ &\leq 2^{n-1} max(|x_1^2|, |x_2|^2, \cdots, |x_t|^2) \text{ for } 1 \leq i \leq k, \text{ and since } k \leq n. \end{aligned}$$

Moreover, since $t \leq k$, hence this inequality holds for $1 \leq i \leq t$, which we wanted for proving. □

4.5 Polynomial factorization method using LLL algorithm

Using above discussed LLL - lattice reduction algorithm, we can create an algorithm for finding factors of any arbitrary polynomials $f \in \mathbb{Z}[X]$.

Procedure for finding factors of any arbitrary polynomial f $\in \mathbb{Z}[X]$:

Let f(x) = $a_n x^n + a_{n-1}x^{n-1} \cdots + a_0$ be any arbitrary polynomial such that all the coefficients $\in \mathbb{Z}[X]$.
Step (1) :
Firstly, we make f(x) to be primitive, because if it is not primitive then we can easily calculate the gcd of its coefficients and take $f_1(x)$ to be f(x) divided through the gcd to get a primitive polynomial. Thus, we have primitive polynomial function.

Step (2) :
Now, we make polynomial with no multiple factors. We know that f has a multiple root if and only if gcd(f(x), f'(x)) = 0 or if f(x) and f'(x) have a common root. We also know from properties of resultant that f(x) and f'(x) have a common root if and only if R(f(x), f'(x)) = resultant(f(x), f'(x)) = 0 or by lemma (4.2.1), if and only if Disc(f(x)) = 0.
If f(x) and f'(x) have a common root, means R(f, f') = 0, then we will calculate r(x) = gcd(f(x), f'(x)), which is then the set of multiple roots of f. Now, set $f_1(x) = \frac{f(x)}{r(x)}$, then obviously $f_1(x)$ has no multiple roots. After factoring $f_1(x)$, it will be very easy to find the factorization of r(x), since r(x) only has factors that appear in the factorization of $f_1(x)$, and there are only finitely many factors in $f_1(x)$.

Step (3) :
Now, we will find out all the irreducible factors of f (mod p), where p is some prime (we will see later that how we will choose prime p). These factors can be determined with the use of Berlekamp's Algorithm.

Berlekamp's Algorithm :

Aim : To find the complete factorization of (f(x) mod p) into irreducible factors in $\mathbb{Z}/p\mathbb{Z}[X]$.
Assume that f(x) is reduced modulo p and square free. Moreover assume that there exists a polynomial $f_0(x) = \prod_{a \in \mathbb{Z}/p\mathbb{Z}} (g(x) - a) \in \mathbb{Z}/p\mathbb{Z}[X]$ such that f(x) divides $f_0(x)$.

Then every irreducible factor of f(x) is also a irreducible factor of $f_0(x)$ and we have :

$$\begin{aligned} f(x) &= gcd(f(x), f_0(x)) \\ &= gcd(f(x), \prod_{a \in \mathbb{Z}/p\mathbb{Z}} (g(x) - a)) \\ &= \prod_{a \in \mathbb{Z}/p\mathbb{Z}} gcd(f(x), g(x) - a). \end{aligned}$$

Clearly, we can observe that not every such gcd will be an irreducible factor of f(x) mod p, so, we need to find enough polynomials g(x) to factor out f(x) mod p completely into irreducible factors.
We know that $\prod_{a \in \mathbb{F}} (X - a) = X^p - X$ *for a finite field* $\mathbb{F}$*, so, for* $f_0(x)$*, we get* $\prod_{a \in \mathbb{Z}/p\mathbb{Z}} (g(x) - a) = g(x)^p - g(x)$*. Since* $f_0(x)$ *is divisible by f(x), hence we get that* $(g(x)^p - g(x))$ *is divisible by f(x) and therefore, we have :*

$$g(x)^p \equiv g(x) \; mod \; (f(x)) \tag{4.17}$$

Hence, we can restrict the search of the g(x) to the set with this above property. This set is also said to be Berlekamp subalgebra and it has some very nice properties, which help us in finding out g(x).
Define a matrix $Q = \{c_{k,l}\}_{0 \leq k,l \leq n}$ *with entries* $c_{k,l}$ *given by the equation.*
$x^{ip} = (c_{n,i}x^n + c_{n-1,i}x^{n-1} + \cdots + c_{0,i}) \; mod \; f(x)$ *for* $i \in [0, 1, \cdots, n]$.
Claim : g(x) will satisfy the equation (4.17) if and only if g(x) is a eigenvector of Q with eigenvalue one.
Proof of the claim : We know that $(x+y)^p = x^p + y^p$ *in* $\mathbb{Z}/p\mathbb{Z}$ *using the expansion of the binomial theorem and fact that* $p|\binom{p}{i}$*. Moreover,* $b^p = b$ *for* $b \in \mathbb{Z}/p\mathbb{Z}$.
Assume now that $g(x) = g_n x^n + g_{n-1}x^{n-1} + \cdots + g_0$ *and it is an eigen vector of Q with eigenvalue one, then :*

$$\begin{aligned} g(x) &= \sum_{i=1}^{n} g_i x^i \\ &= \sum_{i=0}^{n} (\sum_{j=0}^{n} c_{j,i} g_j) x^i \end{aligned}$$

$$\iff \sum_{j=0}^{n} g_j(x^{jp} mod f(x))$$
$$= \sum_{j=0}^{n} (g_j x^{j,p}) mod f(x)$$
$$= (\sum_{j=0}^{n} g_j x^j)^p mod f(x)$$
$$= g(x)^p mod(f(x)),$$

which proves the claim.

With the above claim, we are now able to describe an algorithm that factors the polynomial f(x) in $\mathbb{Z}/p\mathbb{Z}[X]$:

- *First, we calculate Q. This can be done by calculating x^{jp} mod f(x) for all $i \in \{0, 1, 2, \cdots, n\}$.*
- *Now, we calculate all the eigenvectors $g_j(x)$ for $j \in [1, 2, \cdots,$ rank(Q - Id)] (since, there are exactly rank(Q-I) linearly independent vectors satisfying the equation (Q-I)g = 0).*
- *For all $g_j(x)$ and for all $a \in \mathbb{Z}/p\mathbb{Z}$, we now have to calculate gcd(f(x), $g_j(x)$-a), which can be done by the Euclidean algorithm (when the algorithm finds rank(Q-I) different factors, algorithm can stop).*
- *Repeat this procedure for all factors of f(x) which we have found so far, until all the factors are irreducible.*

Hence, we can observe that, these factors h (mod p) of f (mod p) have the following properties :

$$(h \ mod \ p) \ is \ monic \ irreducible \ in \ \mathbb{Z}/p\mathbb{Z}[X] \tag{4.18}$$

$$(h \ mod \ p)^2 \ does \ not \ divide \ (f \ mod \ p) \ in \ \mathbb{Z}/p\mathbb{Z}[X]. \tag{4.19}$$

The equation (4.19) holds, because we know that f(x) is square free and we will select the value of p also such that both of the above equations hold.

Step (4) :
Now, we will have some properties of f and f (mod p) that will allow us to find an

irreducible factor h_0 by taking one irreducible factor h (mod p) of f (mod p) using LLL lattice reduction method.
Now, we choose an monic irreducible factor h (mod p) of f (mod p), let deg(h) = l, where $l < n$, which fulfils the following condition :

$$(h \bmod p^k) \text{ divides } (f \bmod p^k) \text{ in } (\mathbb{Z}/p^k\mathbb{Z})[X]. \tag{4.20}$$

Note (1) : We will describe the value of k at later stage. Now, k is some integer.
Note (2) : From (4.20), we get that (h mod p)|(f mod p). We see this by considering that if x|y then (x mod p)|(y mod p) and ((h mod p^k) mod p) = (h mod p).
Note (3) : We can get such a factor h (mod p) of f (mod p) which satisfy above equation using Hensal's lifting lemma.

Using Hensal lifting lemma (theorem 4.2.2), we can modify h such that (h mod p) does not change but equation (4.20) is true.
Hence, now, we have a factor (h mod p) of (f mod p) which has degree l and satisfy all the conditions (4.18), (4.19) and (4.20).

Proposition 4.5.1. *Let f and h are the functions as above. Then there is a polynomial $h_0 \in \mathbb{Z}[X]$ such that h_0 is an irreducible factor of f, (h mod p) divides (h_0 mod p) and h_0 is unique up to sign.*
Moreover, if g|f in $\mathbb{Z}[X]$, then the following are equivalent :

1. *(h mod p) divides (g mod p) in $(\mathbb{Z}/p\mathbb{Z})[X]$.*
2. *(h mod p^k) divides (g mod p^k) in $(\mathbb{Z}/p^k\mathbb{Z})[X]$.*
3. *h_0 divides g in $\mathbb{Z}[X]$.*

Proof. The existence of such an h_0 follows from equation (4.18) and the note (2). If h itself is a divisor of f, then h_0 = h, and irreducibility follows from equation (4.18). If h does not divide f in $\mathbb{Z}[x]$, there is an irreducible factor h_0 such that (h_0 mod p) factors into (h mod p) and (h_1 mod p) in $\mathbb{Z}/p\mathbb{Z}$, and from equation (4.19), we get the uniqueness. Now we will show that if g|f, then above three statements are equivalent.

$2 \Rightarrow 1$: This is obvious, as, we have discussed it in Note (2).

$3 \Rightarrow 1$: h_0|g which means, (h_0 mod p)|(g mod p) $\Rightarrow$ (h mod p)|(g mod p).

$1 \Rightarrow 3$: Given (h mod p)|(g mod p) and (h mod p)^2 $\nmid$(f mod p). Hence (h mod p)$\nmid$(f/g mod p) in $\mathbb{Z}/p\mathbb{Z}$[X]. Since, (h mod p) is a factor of (h_0 mod p), so, (h_0 mod p)$\nmid$(f/g mod p) and $h_0 \nmid$(f/g). Hence h_0 has to be a divisor of g.

$3 \Rightarrow 2$: Now, we have $h_0|g \Rightarrow$ (h_0 mod p)|(g mod p) $\Rightarrow$ (h mod p)|(g mod p) and also (h mod P)|(f mod p) but (h mod P)^2 $\nmid$(f mod p), so, (h mod p)$\nmid$($\frac{f}{g}$ mod p), which means, (h mod p) and (f/g mod p) have no common divisor in $\mathbb{Z}/p\mathbb{Z}$. Hence there exists s and t $\in \mathbb{Z}$[X] such that :
(s mod p)(h mod p) + (t mod p)(f/g mod p) = 1
$\Rightarrow$ sh + t(f/g) = 1 - rp with r $\in \mathbb{Z}$[X].
Now, by multiplying both sides with g and v(r) = 1 + pr + p^{2}r^2 + $\cdots$ + p^{k-1}r^{k-1}, we get :
sgv(r)h + tv(r)f = (1 - pr)v(r)g = (1 - p^{k}r^k)g
$\Rightarrow$ (s_1)h + t_1f) mod p^k = g mod p^k.
Now, since we know that (h mod p^k)|(f mod p^k), so, left hand side is divisible by h mod p^k, hence, right hand side (g mod p^k) is divisible by (h mod p^k) which we wanted. □

Note : If we choose g to equal h_0 then third statement is true, and so, by the equivalence of the three statements, we get that (h mod p^k)|(h_0 mod p^k).
Now with all of the above quantities, our aim is now to find a way to calculate h_0. Now for finding h_0, we will use lattice-reduction algorithm.

Defining lattice : *To apply the results from the section 4.3, we need to introduce a lattice L representing all the possible polynomials for h_0. Let dim(L) = dimension of L = m. Clearly m $\geq$ l because the degree of h_0 is greater than or equal to the degree of h. An upper bound for m would be n-1 because a factor (not necessarily irreducible) of a polynomial has at most degree one less than that the polynomial itself. We will set the actual value of m later for shorter running time of the algorithm, but we consider it fixed.*
Set L = set of all polynomials in $\mathbb{Z}[X]$ with property that if they are taken modulo p then they are divisible by (h mod p) in $\mathbb{Z}/p\mathbb{Z}$.

Claim : A basis of L is given by : $\{p^k X^i$; $0 \leq i < l\} \cup \{hX^j$; $0 \leq j \leq$ m-l$\}$.
Proof of the claim : Since for basis of L, (h mod p) has to be divide each of these polynomials mod p. We can see that h divides hX^j, hence (h mod p) divides (hX^j mod

p) as well. The polynomials in the first set are zero when they are taken mod p and zero can be divided by everything. Hence, all the linear combinations of these basis elements satisfy the desired property. Now, for checking that these two sets indeed cover all the polynomials, we can see that there are l + (m-l+1) = m + 1 elements in the basis of L and they are linearly independent. Hence the claim follows.

Now, we can also calculate the determinant of the lattice L :

$$d_L = det \begin{bmatrix} p^k & 0 & \cdots & 0 & h_0 & 0 & \cdots & 0 \\ 0 & p^k & \cdots & 0 & h_1 & h_0 & \cdots & 0 \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ 0 & 0 & \cdots & p^k & h_{l-1} & h_{l-2} & \cdots & \\ 0 & 0 & \cdots & 0 & 1 & h_{l-1} & \cdots & \\ 0 & 0 & \cdots & 0 & 0 & 1 & \cdots & \\ . & . & . & . & . & . & & \\ . & . & . & . & . & . & & \\ 0 & 0 & \cdots & 0 & 0 & \cdots & & 1 \end{bmatrix} = p^{kl}$$

Claim (A) : h_0 can be calculated as the greatest common divisor of some basis elements of a reduced basis of L.
Proof of the claim : For proving above claim, we need three propositions, which are discussed below :

Proposition 4.5.2. *Let $b \in L$ such that $p^{kl} > |f|^m |b|^n$. Then h_0 divides b in $\mathbb{Z}[X]$ and therefore gcd(f ,b) $\neq$ 1.*

Note : $|f|$ = Euclidean norm of $(a_n, a_{n-1}, \cdots, a_0) = \sqrt{|a_0|^2 + |a_1|^2 + \cdots + |a_n|^2}$.

Proof. Let gcd (f, b) = g. It is enough to show that (h mod p)|(g mod p) because then by proposition (4.5.1), it follows that $h_0|b$ in $\mathbb{Z}[X]$ and hence h_0 divides b.
Claim (1) : (h mod p)|(g mod p).
proof of the claim : To prove this claim, we assume to the contrary that let (h mod p)∤(g mod p). Since (h mod p) is irreducible in $\mathbb{Z}/p\mathbb{Z}$, so, we have that (h mod p) and (g mod p) are relatively prime in $\mathbb{Z}/p\mathbb{Z}$ and therefore

$$sh + tg = 1 - rp \tag{4.21}$$

for some s, t, r $\in \mathbb{Z}$[X].
Let deg(g) = e and deg(b) = e'
Define M = {sf + tb; s, t $\in \mathbb{Z}$[X], deg(s) < e' - e, deg(t) < n - e}
We can observe that M is the subset of the set of all polynomials with integer coefficients and degree lesser than or equal to n + e' - e - 1, i.e., M $\subset$ ($\mathbb{Z}$ + $\mathbb{Z}$X + $\cdots$ + $\mathbb{Z}X^{n+e'-e-1}$
Claim (2) : M is a lattice of rank n + e' - 2e.
Proof of the claim : We can observe that we can define a basis for M by projections such that it has rank n + e' - 2e.
Define M' to be the projection of M on ($\mathbb{Z}X^e$ + $\mathbb{Z}X^{e+1}$ + $\cdots$ + $\mathbb{Z}X^{n+e'-e-1}$).
Claim (3) : Kernel of this above projection is trivial, and so, its image has the same rank as M itself.
proof of the claim : Suppose (sf + tb) $\in$ M projects to zero in M'. Then we have deg(sf + tb) < e. Since g divides f and b, so, it will also divide (sf + tb) and we get : (sf + tb) = 0 and hence, s(f/g) = -t(b/g). Since gcd(f, b) = g, it follows that (f/g) and (b/g) has no common divisor and thus, (f/g) has to divide t and by the analysis of the degrees of (f/g) and t, we see that t needs to be zero. Similarly, s needs to be zero which proves that kernel is zero, and hence claim (3) follows.

From the claim (3), we have that M has the same rank as M'. Now, we have that the projections of {X^if ; 0$\leq$ i < e'-e} $\cup$ {X^jb ; 0 $\leq$ j < n-e} on M' are linearly independent and span M'. Hence, M' is a lattice of rank n+e-2e and from the claim (3), we have that M has the same rank n + e' - 2e, which proves claim (2).

Now, from the second part of the theorem (4.4.1), we get that :

$$d(M') \leq \prod_{i=0}^{e'-e-1} |X^i f| \prod_{j=0}^{n-e-1} |X^j b| = |f|^{e'-e}|b|^{n-e} \leq |f|^m|b|^n < p^{kl} \tag{4.22}$$

In order to derive a contradiction :
Claim (4) : The set {$\theta \in$ M; deg(θ) < e + l} is a subset of $p^k\mathbb{Z}$.
Proof of the claim : let θ be an element of the above set. Then by the definition of M, g divides θ. So, we can multiply the equation (4.21) by (θ/g) and v(r) = 1 + pr + p^2r^2 + $\cdots$ + $p^{k-1}r^{k-1}$. We have done the same calculation in the proof of proposition (4.5.1), hence we get :
(s_1h + $t_1\theta$) mod p^k = (θ/g) mod p^k.

where $s_1 = s(\theta/g)v(r)$ and $t_1 = tv(r)$ and hence both are in $\mathbb{Z}[X]$.
Since $b \in L$, hence (h mod p^k) divides (b mod p^k), and we know that (h mod p^k) divides (f mod p^k) from equation (4.20). $\theta \in M$, so, it a sum of multiple of f and b. Hence, we have (h mod p^k) divides (θ mod p^k). Therefore, from the above inequality, we have that (h mod p^k) divides ((θ/g) mod p^k). By observing the degrees, deg(h mod p^k) = l and deg((θ/g) mod p^k) < e + l - e = l. Hence, ((θ/g) mod p^k) has to be zero, and so, (θ mod p^k) has to be zero, which proves claim (4).

Now, choose a basis b_e, b_{e+1}, $\cdots$, $b_{n+e'-e-1}$ of M' such that deg(b_i) = i. Then we can observe that the matrix representing M' corresponding to this chosen basis has upper triangular form and we can calculate d(M') easily by multiplying the leading coefficients. By the observation above, the set ($\{\theta \in M$; deg(θ) < e + $l\} \subseteq p^k\mathbb{Z}[X]$). So, we have b_e, b_{e+1}, $\cdots$, b_{e+l-1} are all divisible by p^k and so are the leading coefficients of b_e, b_{e+1}, $\cdots$, b_{e+l-1}. Hence, we get that d(M') $\geq p^{kl}$ but by equation (4.22), we have d(M') < p^{kl}. Hence, we get a contradiction. Thus claim (1) follows and hence proposition. □

By the next proposition, we will check that deg(h_0) is smaller than m, i.e., $h_0 \in L$. This will be useful to determine the value of m in the algorithm. Clearly, if we can take m = n-1 to be sure that $h_0 \in L$, but we can shorten the algorithm running time if we can choose m as small as possible.
Before proceeding in the next proposition, first, we will see one another useful theorem which is useful in the proof of the next proposition.

Theorem 4.5.3. *(Landau-Mignotte) : Let f(x) $\in \mathbb{Z}[X]$, which have degree n and g(x) $\in \mathbb{Z}[X]$, which is a divisor of f(x) of degree m. Then we have*

$$|g| \leq \binom{2m}{m}^{1/2} |f| \tag{4.23}$$

Proof. For proving this, firstly, we need following claim :
Claim : |(x - a)h| = $|a||(x - \bar{a}^{-1}h|$ for a $\in \mathbb{C}$ and h $\in \mathbb{C}[X]$.
Proof of the theorem :

To prove the above claim, define $h = \sum_{i=0}^{n} h_i x^i$ and $h_{-1} = h_{n+1} = 0$ and now we calculate :

$$
\begin{aligned}
|(x-a)h|^2 &= \sum_{i=0}^{n+1} |h_{i-1}|x^i|^2 - ah_i|^2|x^i|^2 \\
&= \sum_{i=0}^{n+1} (h_{i-1} - ah_i)\overline{(h_{i-1} - ah_i)}|x^i|^2 \\
&= \sum_{i=0}^{n+1} (h_{i-1} - ah_i)(\overline{h_{i-1}} - \bar{a}\bar{h_i})|x^i|^2 \\
&= \sum_{i=0}^{n+1} (|h_{i-1}|^2 - ah_i\overline{h_{i-1}} - \bar{a}\bar{h_i}h_{i-1} + |ah_i|^2|x^i|^2)
\end{aligned}
$$

$$
\text{Since } \sum_{i=0}^{n+1} |h_i|^2 = \sum_{i=0}^{n+1} |h_{i-1}|^2 + |h_{n+1}|^2 - |h_{-1}|^2 = \sum_{i=0}^{n+1} |h_{i-1}|^2
$$

$$
\text{and similarly } \sum_{i=0}^{n+1} |ah_i|^2 = \sum_{i=0}^{n+1} |ah_{i-1}|^2.
$$

So, after using this, we get :

$$
\begin{aligned}
|(x-a)h|^2 &= \sum_{i=0}^{n+1} |\bar{a}h_{i-1} - h_i|^2|x^i|^2 \\
&= |(\bar{a}x - 1)h|^2 \\
&= |a|^2|(x - \bar{a}^{-1})h|^2
\end{aligned}
$$

and hence we proves the claim.

Now, let a_1, a_2, $\cdots$, a_s be the set of the roots of f inside the unit disk and a_{s+1}, a_{s+2}, $\cdots$, a_n be the set of roots of f outside the unit disk which all are in decreasing order in absolute value and f_n be the leading coefficients of f. Then we have :

$$
|f|^2 = |f_n \prod_{i=1}^{s} (x - a_i) \prod_{i=s+1}^{n} (x - a_i)|^2
$$

Now using the above claim, we get :

$$
|f|^2 = |a_1 a_2 \cdots a_s|^2 |f_n \prod_{i=1}^{s} (x - \bar{a}_i) \prod_{i=s+1}^{n} (x - a_i)|^2
$$

$$= |a_1 a_2 \cdots a_s|^2 |f_n x^n + \cdots + (-1)^n f_n \prod_{i=1}^{s} \bar{a_i}^{-1} \prod_{i=s+1}^{n} a_i|^2$$

$$\geq |a_1 a_2 \cdots a_s|^2 |f_n \prod_{i=1}^{s} \bar{a_i}^{-1} \prod_{i=s+1}^{n} a_i|^2$$

$$= |f_n \prod_{i=s+1}^{n} a_i|^2.$$

Now, let $b_1, b_2, \cdots, b_m$ are the roots of g(x) ordered such that $b_i \geq b_{i+1}$ for all 1 $\leq$ i $\leq$ m-1 and $g(x) = g_m \prod_{i=1}^{m}(x-b_i) = \sum_{i=0}^{m} g_i x^i$.
Define S_i = Set of all subsets of $b_1, b_2, \cdots, b_m$ with m-i elements. Then :
$|g_i| = |g_m \sum_{S_i}(\prod_{b_j \in S_i} b_j)|$
Since there are $\binom{m}{i} = \binom{m}{m-i}$ such subsets in S_i and the absolute value of such a subset can be at most $|b_1 b_2 \cdots b_{m-i}|$, hence we get :
$|g_i| \leq g_m \binom{m}{i} |b_1 b_2 \cdots b_{m-i}|$
Since we have that g|f, so, $|b_1 b_2 \cdots b_{m-i}| \leq |a_{s+1} a_{s+2} \cdots a_{s+m-i}|$ and
$|g_i| \leq g_m \binom{m}{i} |a_{s+1} a_{s+2} \cdots a_{s+m-i}| \leq g_m \binom{m}{i} |a_{s+1} a_{s+2} \cdots a_n| \leq \binom{m}{i} \frac{|g_m|}{|f_n|} |f|$
Since g|f, so, we also have that $g_m | f_n$ and therefore $\frac{g_m}{f_n} \leq 1$, and we get :
$|g_i| \leq \binom{m}{i} |f|$
Thus, we have :
$|g| = (\sum_{i=0}^{m} |g_i|^2)^{\frac{1}{2}} \leq (\sum_{i=0}^{n} \binom{m}{i}^2 |f|^2)^{\frac{1}{2}} = \binom{2m}{m}^{\frac{1}{2}} |f|$, which proves the theorem. □

Proposition 4.5.4. *Let $b_1, b_2, \cdots, b_{m+1}$ be a reduced basis for L and assume $p^{kl} > 2^{\frac{mn}{2}} \binom{2m}{m}^{\frac{n}{2}} |f|^{m+n}$, Then we have :*
deg(h_0) $\leq m \iff |b_1| < (p^{kl}/|f|^m)^{\frac{1}{n}}$

Proof. We will show the proof of the proposition both sides.
$\Rightarrow$ Given deg(h_0) $\leq$ m, so, $h_0 \in$ L. Hence, applying theorem (4.4.1) part (4) to b_1 and h_0, we get : $|b_1| \leq 2^{\frac{m}{2}} |h_0|$.
Since we know that h_0|f, so, deg(h_0)$\leq$m and by the above theorem, we have : $|h_0| \leq \binom{2m}{m}^{\frac{1}{2}} |f|$. So, we get :
$|b_1| \leq 2^{\frac{m}{2}} |h_0| \leq 2^{\frac{m}{2}} \binom{2m}{m}^{\frac{1}{2}} |f| = (2^{\frac{mn}{2}} \binom{2m}{m}^{\frac{n}{2}} |f|^n \frac{|f|^m}{|f|^m})^{\frac{1}{n}}$
$< (\frac{p^{\frac{kl}{n}}}{|f|^{m/n}}) = (\frac{p^{kl}}{|f|^m})^{\frac{1}{n}}$

$\Leftarrow$ Given $|b_1| < (p^{kl}/|f|^m)^{\frac{1}{n}} \Rightarrow p^{kl} > |b_1|^n |f|^m$. Hence, by previous proposition, we have that $h_0 | b_1$ in $\mathbb{Z}[X]$ and since deg(b_1) $\leq$ m, so, we have deg(h_0) $\leq$ m. □

Proposition 4.5.5. *Similar to above proposition, let b_1, b_2, $\cdots$, b_{m+1} be a reduced basis for L and assume $p^{kl} > 2^{\frac{mn}{2}}\binom{2m}{m}^{\frac{n}{2}}|f|^{m+n}$. Let t be the largest integer in {1, 2, $\cdots$, m+1} such that $|b_t| < (\frac{p^{kl}}{|f|^m})^{\frac{1}{n}}$.*
Then we have :
deg(h_0) = m + 1 - t and h_0 = gcd(b_1, b_2, $\cdots$, b_t).

Proof. Deinfe J = set of all indices j such that $|b_j| < (\frac{p^{kl}}{|f|^m})^{\frac{1}{n}}$. Hence, by the above proposition (4.5.2), we have $h_0|b_j$ for all j $\in$ J. So, let us define $h_1 = \gcd(\{b_j ; j \in J\})$.
Claim (1): $h_0 = h_1$
Proof of the claim : Clearly, we cansee that h_0 divides h_1.
Claim (2) : There are at most m + 1 - deg(h_1) elements in J.
proof of the claim : Since h_1 divides all b_j (j $\in$ J) and the degree(b_j) is smaller than m. So, b_j is an element of the lattice L_1, which defined through the basis :
Basis for $L_1 = \{h_1X^i ; 0 \leq i \leq m - \deg(h_1)\}$.
Since, by the definition, all the b_j's are linearly independent and there are at most m + 1 - deg(h_1) linearly independent elements in the lattice L_1, so, there are at most m + 1 - deg(h_1) elements in J. Thus claim (2) proves.

Since in the proof of the above claim (2), we have $h_0X^i \in L$ for i $\in$ {0, 1, $\cdots$, m - deg(h_0)}. Hence, by using the forth part of the theorem (4.4.1), we get :
$|b_k| \leq 2^{\frac{m}{2}}\max\{|x_i|; 0 \leq i \leq m - \deg(h_0)\} = 2^{\frac{m}{2}}|h_0X^i| = 2^{\frac{m}{2}}|h_0|$.
Now, using Landau - Mignotte theorem, we have that $|X^ih_0| \leq \binom{2m}{m}^{\frac{1}{2}}|f|$ for all i $\in$ {0, 1, $\cdots$, m - deg(h_0)}, so, we get :
$|b_k \leq 2^{\frac{m}{2}}\binom{2m}{m}^{\frac{1}{2}}|f| < (\frac{p^{kl}}{|f|^m})^{\frac{1}{n}}$ for all k $\in$ {1, 2, $\cdots$, m + 1 - deg(h_0)}.
Since, J was defined to be all the indices such that exactly above inequality holds. So, {1, 2, $\cdots$, m + 1 - deg(h_0)} $\subset$ J. Since deg(h_1) $\geq$ deg(h_0) and with the above observation from claim (2) about the upper bound for the number of elements in J, we get following inequality :
#{1, 2, $\cdots$, m + 1 - deg(h_1)} = m + 1 - deg(h_1) $\leq$ #{1, 2, $\cdots$, m + 1 - deg(h_0)} $\leq$ # J $\leq$ m + 1 - deg(h_1).
Hence we can conclude that deg(h_0) = deg(h_1).
Since J = {1, 2, $\cdots$, m + 1 - deg(h_0)}, hence, t = m + 1 - deg(h_0), and so, deg(h_0) = m + 1 - t.
Now, we know that h_0 and h_1 have the same degree and $h_0|h_1$, so, we know that they are equal up to a factor in $\mathbb{Z}$, so, we will show that this factor equals to one, i.e., if content of h_1 is one. Then $h_0 = h_1$.

Claim (3) : h_1 is primitive.
Proof of the claim : Choose some arbitrary j ∈ J and let c_j be the content of b_j. We know that all b_j are divisible by h_0 and h_0 is primitive, so, $h_0|\frac{b_j}{c_j}$. By the definition of L, we have that $b_j/c_j \in L$. But b_j is the basis element of L, so, $c_j = 1$ for all j ∈ {1, 2, $\cdots$,t} and hence the content of $h_1 = \gcd(b_1, b_2, \cdots, b_t)$ is one. So, h_1 is primitive, which proves the claim (3), and hence claim (1) follows. Therefore, proposition follows. □

Thus, from the above proposition, we can find an irreducible factor of f(x), and by repeating this algorithm for different (h mod p) and taking all the reducible factors of (f(x)) as a new f(x), we can find out all the irreducible factors of f(x).

Remark 4.5.6. ***Deciding the value of p :*** *Define p to be the smallest prime not dividing R(f(x), f'(x)) because we choose p such that equation (4.19) and (4.20) holds.*
Claim : The above chosen value of p is reasonable, i.e., for this value of p, equation (4.19) and (4.20) holds.
Proof of the claim : Since we know that R(f(x), f'(x)) is up to sign equal to the product of the leading coefficients f_n and the discriminant D(f) of f (using lemma (4.2.1)). Since, we choose p such that R(f(x), f'(x)) ≠ 0 (mod p), hence, we have $f_n D(f) \neq 0 \pmod p$ and therefore $f_n \neq 0 \pmod p$ and $D(f) \neq 0 \pmod p$.
Now, we see that, if we choose any two roots of f, say x_i and x_j, then since $D(f) \neq 0 \pmod p$, hence $(x_i - x_j) \neq (0 \bmod p)$ which means that x_i and x_j are not differ by a multiple of p. Hence $(x_i \bmod p) \neq (x_j \bmod p)$ and $(x - x_i \bmod p) \neq (x - x_j \bmod p)$.
For proving the contradiction of equation (4.19), i.e., there are multiple roots in (f mod p), we would need $(x - (x_i \bmod p))$ to be equal to $(x - (x_j \bmod p))$ for some choice of x_i and x_j, but from the above discussion, we can say that there are none.
Since $(h \bmod p)|(f \bmod p)$, hence equation (4.20) satisfies for the choice of p, which proves our claim.

Remark 4.5.7. ***Deciding the value of k :*** *We decide k such that equation (4.18), (4.19) and (4.20) is satisfied for the prime p which we specified in the above remark. If we set, k =1 then we can observe that equation (4.18), (4.19) and (4.20) holds.*
Since for using the results from the three propositions, we need the equation $p^{kl} > 2^{\frac{mn}{2}} \binom{2m}{m}^{\frac{n}{2}} |f|^{m+n}$ (proposition 4.5.4) to hold. Recall that $l \leq deg(h_0) \leq m$. Degree of h_0 can not be grater than m, because $h_0 \in L$, otherwise, h_0 is not in the lattice, and we can not find out h_0 with the results above. Since in the worst case $deg(h_0) = n-1$,

hence set m = n - 1 in the above inequality, and define k to be the least integer such that the inequality holds, i.e.

$$k = min\{k \in \mathbb{Z};\ p^{kl} > 2^{\frac{n(n-1)}{2}} \binom{2n-2}{n-1}^{\frac{n}{2}} |f|^{2n-1}\} \tag{4.24}$$

Clearly, equation (4.18) and (4.19) satisfy and we have already shown that equation (4.20) is satisfied by using Hensal's lifting lemma for any integer $k \geq 2$.

Remark 4.5.8. *We can choose m smaller such that the running time of the algorithm becomes shorter. For this we can follow this way :*
Let v be such that $l \leq \lfloor \frac{n-1}{2^v} \rfloor$. Choose $m_i = \lfloor \frac{(n-1)}{2^{v-i}} \rfloor$ for $0 \leq i \leq v$ and check that if $deg(h_0) \leq m_i$, by the result of proposition (4.5.4) for every value of m_i. As soon as $deg(h_0) \leq m_j$ for some m_j, then calculate h_0 using proposition (4.5.5). So, for every choice of m_i, we first determine a reduced basis $b_1, b_2, \cdots, b_{m+1}$ by the LLL-Algorithm and then we check if $deg(h_0) \leq m_i$ by checking if $|b_1| < (p^{kl}/|f|^m)^{\frac{1}{n}}$. If the inequality holds, we can calculate h_0 by the equation $h_0 = gcd(b_1, b_2, \cdots, b_j)$, if not, continue with the next value of m_{j+1}. Since we know that m goes up to n - 1, it is guaranteed that we will find h_0 sooner or later.

4.5.1 Lenstra - Lenstra - Lovasz (LLL) Algorithm for factoring a nonconstant polynomial:

Let R(f(x), f'(x)) = r ;
g(x) = 1 ;
If (r = 0) {
g(x) = gcd(f(x), f'(x)) ; f = f/g ; r = R(f(x), f'(x));
}
Choose p = smallest prime number such that p∤r.
H = {h mod p ; (h mod p) is irreducible factor of (f mod p)} (using Berlekamp's Algorithm)
Define $f_1 = 1$; $f_2 = f$;
While ($f_2 \neq 1$) { h = some arbitrary element of H; l = degree of h;
If (l = n) {
$h_0 = f$ };
break ;
$k = min\{k \in \mathbb{N};\ p^{kl} > 2^{\frac{n(n-1)}{2}} \binom{2n-2}{n-1}^{\frac{n}{2}} |f|^{2n-1}\}$;

Modify h by using Hensal lifting lemma.
(**) m = n - 1;
Call LLL algorithm (4) of section (4.3) to $\{p^k X^i \ ; \ 0 \leq i < l\} \cup \{hX^j \ ; \ 0 \leq j \leq m\text{-}l\}$
j = greatest integer such that $|b_j| < (p^{kl}/|f|^m)^{\frac{1}{n}}$ *(b_j means - one of the reduced basis from above chosen basis)*
$h_0 = gcd(b_1, \ b_2, \cdots, \ b_j)$ *(**)*
$f_1 = f_1 h_0$ *;* $f_2 = f_2/h_0$*;*
$H_0 = \{h \ mod \ p \ ; \ (h \ mod \ p) | (h_0 \ mod \ p)\}$ *;*
$H = H \backslash H_0$ *;*
}
Now, let $g_1 = 1$*;* $g_2 = g$*;*
while $(g_2 \neq 1)$ {
Check all factors of f if they divide g_2*. Let* $\bar{f}$ *divide* g_2*.*
$g_1 = g_1 \bar{f}$ *;* $g_2 = g_2/\bar{f}$ *;*
}

If we use remark 4.5.8 for optimizing the running time of the algorithm by smaller value of m, then we can replace line 16 to 20 (means (**) to (**)) by the following algorithm :

Define $u = max\{ \ u \in \mathbb{N}; \ l < \frac{(n-1)}{2^u}\}$*;*
While $(m \leq (n\text{-}1)$ {
$m = \lfloor \frac{(n-1)}{2^u} \rfloor$ *;*
Call LLL algorithm (4) of section (4.3) to $\{p^k X^i \ ; \ 0 \leq i < l\} \cup \{hX^j \ ; \ 0 \leq j \leq m\text{-}l\}$
If $(|b_j| < (\frac{p^{kl}}{|f|})^{\frac{1}{n}})$ {
$h_0 = gcd(b_1, \ b_2, \cdots, \ b_j)$*;*
$m = n$*;*
}
$u = u$ *- 1;*
}

Bibliography

[1] *Dickson, L. E.,"History of the theory of numbers", Vol. I, G. E. Stechert and company, 1934.*

[2] *Neil Koblitz, A Course in Number Theory and Cryptography, Springer-Verlag, New York, second edition, 2010.*

[3] *Kapil H. Paranjape, Notes on Miller-Rabin primality test, available at http://www.imsc.res.in/ kapil/crypto/index.html.*

[4] *Abhijit Das, Computational Number Theory, CRC Press, first edition, 2013.*

[5] *David M. Burton, Elementary Number Theory, Tata McGraw-Hill Education Private Limited, sixth edition, 2011.*

[6] *M.Agrawal, N. Kayal and N. Saxena, "Primes is in P",Annals of Mathematics 160(2), 2004, 781-793.*

[7] *A. Schnhage and V. Strassen, Schnele multiplikation, grosser zahlen. Computing 7 (1971), 281-292.*

[8] *Henri Cohen, A Course in computational algebraic number theory, Springer-Verlag, Graduate texts in mathematics;138, 1996.*

[9] *Dummit and Foote, Abstract Algebra, second edition, John Wiley and Sons, Inc., 1999.*

[10] *M. Dietzfelbinger, Primality Testing in Polynomial Time, Springer, 2004.*

[11] *Alford, Granville and Pomerance, there are infinitely many carmichael numbers, Ann. of Mathematics 140(3), 1994, 703-722.*

[12] *Joachim von zur Gathen and Jurgen Gerhard, Modern Computer Algebra, Cambridge University Press, 1999.*

[13] R. Stanley, Enumerative Combinatorics Volume I, Cambridge University Press, 1997.

[14] H. L. Montgomery and R. C. Vaughan, The exceptional set in Goldbachs problem, Acta Arith. (27) 1975, 353370.

[15] Manasse, Pollard, Lenstra, Lenstra, The number field Sieve, Lecture notes in Mathematics, 1993.

[16] J. M. Couveigne, Computing a square root for the number field Sieve. Lecture notes in Mathematics, 1993.

[17] D. E. Knuth, The art of Computer programming, Semi numerical algorithms, second edition, volume - 2.

[18] Pomerance, Buhler, Lenstra, Factoring integers with number field Sieve, Lecture notes in Mathematics, 1993.

[19] Manasse, Pollard, Lenstra, Lenstra, The factorization of the ninth Fermat number, Mathematics of computation, 1993.

[20] D. H. Wiedemann, Solving sparse linear equations over finite fields, IEEE transactions on information theory, issue - 1, 1986.

[21] Victor Shop, A Computational Introduction to number theory and algebra, Cambridge university press, 2005.

[22] Buchmann, Loho and Zayer, An implementation of the general number field Sieve, Advances in Cryptology (Proceeding of Crpto '93)(Berlin), Lecture Notes in computer science, no 773, Springer-Verlag, 1994, pp. 50 - 94.

[23] Jean-Marc Couveignes, Computing a square root for the number field Sieve, In Lenstra and Lenstra [29], pp. 95-102.

[24] Henri Cohen, A Course in computational algebraic number theory, Graduate Texts in Mathematics, vol. 138, Springer-Verlag, Berlin, 1993.

[25] A. K. Lenstra, D. J. Bernstein, A general number field Sieve implementation, In Lenstra and Lenstra [29], pp. 103-126.

[26] Thomas W. Hungerford, first edition, Algebra, Springer-Verlag, New York, 2004.

[27] Huizing, An implementation of the number field Sieve, Experimental mathematics (5) 1996, no. 3, 231-253.

[28] Friedberg, Insel, Spence, Linear algebra, Prentice-Hall, Englewood Cliffs, New Jersey, Second edition, 1989

[29] A.K. Lenstra, H.W. Lenstra, The development of the number field Sieve, Lecture notes in mathematics, vol. 1554, springer-verlag, berlin, 1993.

[30] Niven, Zuckerman, Montgomery, An introduction to the theory of numbers, fifth edition, John Wiley and Sons, New York, 1991.

[31] Peter Montgomery, Square roots of products of algebraic numbers, Mathematics of computation 1943-1993, 1993.

[32] J. M. Pollard, The lattice Sieve, In lenstra and lenstra [29], pp. 43-49.

[33] McCurley, Golliver, Lenstra, Lattice sieving and trial division, Algorithmic number theory, Lecture notes in computer science, Springer-Verlag, Berlin, 1994.

[34] Hans Riesel, Prime numbers and computer methods of factorization, second edition, Progress in mathematics, vol. 126, Birkhausar, Boston, 1994.

[35] Stewart, Tall, Algebraic number theory, second edition, Chapman and Hall, London, 1987.

[36] Cohen, Cuypers, Sterk, Some Tapas of computer algebra, pp. 66-90, Springer-Verlag, 1999.

[37] E.R. Berlekamp, Factoring polynomials over finite fields, Bell systems technical journal 46, 1967.

[38] A.K. Lenstra, H.W. Lenstra, L. Lovansz, Factoring polynomials with rational coefficients, pp. 515-534, 1982.

[39] A. Schnhage and V. Srassen, Schnele multiplication grosserr zahlen, Computing 7 (1971), 281-292.

[40] Joachim von zur Gathen and Jurgen Gerhard, Modern Computer Algebra, Cambridge University Press, 1999.