量子計算的數學基礎 MA5501*

Chapter 6. Shor's Factoring Algorithm

Chapter 6. Shor's Factoring Algorithm

Suppose that N is the product of two unknown prime numbers p, q. Then a classical way of factoring N is to run a routine check to see which natural number not greater than $\sqrt{N}$ is a factor of N. The worse case scenario is to try this division $\sqrt{N}$ times in order to find the correct factors. The current encryption system is designed based on the fact that "it is much easier to compute the product of two prime numbers than to factor a number which is the product of two prime numbers". In the following, we quickly review the current encryption system and the mathematics behind it, and study the most famous quantum algorithm to factor large numbers, the Shor algorithm.

Chapter 6. Shor's Factoring Algorithm

Suppose that N is the product of two unknown prime numbers p, q. Then a classical way of factoring N is to run a routine check to see which natural number not greater than $\sqrt{N}$ is a factor of N. The worse case scenario is to try this division $\sqrt{N}$ times in order to find the correct factors. The current encryption system is designed based on the fact that "it is much easier to compute the product of two prime numbers than to factor a number which is the product of two prime numbers". In the following, we quickly review the current encryption system and the mathematics behind it, and study the most famous quantum algorithm to factor large numbers, the Shor algorithm.

Chapter 6. Shor's Factoring Algorithm

Suppose that N is the product of two unknown prime numbers p, q. Then a classical way of factoring N is to run a routine check to see which natural number not greater than $\sqrt{N}$ is a factor of N. The worse case scenario is to try this division $\sqrt{N}$ times in order to find the correct factors. The current encryption system is designed based on the fact that "it is much easier to compute the product of two prime numbers than to factor a number which is the product of two prime numbers". In the following, we quickly review the current encryption system and the mathematics behind it, and study the most famous quantum algorithm to factor large numbers, the Shor algorithm.

§6.1 RSA Encryption

RSA is an asymmetric encryption (非對稱式加密) technique that uses two different keys as public and private keys to perform the encryption and decryption. The public key is represented by the integers n and e, and the private key by the integer d. A basic principle behind RSA is to find three very large positive integers e, d, and n, such that with modular exponentiation all messages $m \in \mathbb{N}$ with $0 \leqslant m < n$ satisfies

$$(m^e)^d \equiv m \pmod{n}$$

and that knowing e and n, or even m, it can be extremely difficult to find d.

§6.1 RSA Encryption

RSA is an asymmetric encryption (非對稱式加密) technique that uses two different keys as public and private keys to perform the encryption and decryption. The public key is represented by the integers n and e, and the private key by the integer d. A basic principle behind RSA is to find three very large positive integers e, d, and n, such that with modular exponentiation all messages $m \in \mathbb{N}$ with $0 \leqslant m < n$ satisfies

$$(m^e)^d \equiv m \pmod{n}$$

and that knowing e and n, or even m, it can be extremely difficult to find d.

§6.1 RSA Encryption

§6.1.1 Mathematical foundation

Definition (Greatest common divisor)

Let a and b be non-zero integers. We say the integer d is the **greatest common divisor (gcd)** of a and b, and write $d = \gcd(a, b)$, if

1. d is a common divisor of a and b.
2. every common divisor c of a and b is not greater than d.

§6.1 RSA Encryption

Theorem

Let a and b be positive integers with $a \leqslant b$. *Suppose that* $b = aq_0 + r_1$, $a = r_1 q_1 + r_2$, $r_{j-1} = r_j q_j + r_{j+1}$ *for* $2 \leqslant j \leqslant k$, *where* $0 = r_{k+1} < r_k < \cdots < r_2 < r_1 < a$ *and* $q_j \in \mathbb{N}$ *for all* $0 \leqslant j \leqslant k$.

1. $\gcd(a, b) = r_k$, *the last non-zero remainder in the list.*
2. *If* $\{s_j\}_{j=-1}^{k}$ *and* $\{t_j\}_{j=-1}^{k}$ *are defined by*

$$s_j = \begin{cases} 1 & \text{if } j = -1\,, \\ 0 & \text{if } j = 0\,, \\ s_{j-2} - q_{j-1}s_{j-1} & \text{if } j \geqslant 1\,, \end{cases}$$

$$t_j = \begin{cases} 0 & \text{if } j = -1\,, \\ 1 & \text{if } j = 0\,, \\ t_{j-2} - q_{j-1}t_{j-1} & \text{if } j \geqslant 1\,, \end{cases}$$

then

$$at_j + bs_j = r_j \qquad \forall\, 1 \leqslant j \leqslant k\,.$$

§6.1 RSA Encryption

Theorem

Let a and b be positive integers with $a \leqslant b$. Suppose that $b = aq_0 + r_1$, $a = r_1 q_1 + r_2$, $r_{j-1} = r_j q_j + r_{j+1}$ for $2 \leqslant j \leqslant k$, where $0 = r_{k+1} < r_k < \cdots < r_2 < r_1 < a$ and $q_j \in \mathbb{N}$ for all $0 \leqslant j \leqslant k$.

1. $\gcd(a, b) = r_k$, *the last non-zero remainder in the list.*
2. *If $\{s_j\}_{j=-1}^{k}$ and $\{t_j\}_{j=-1}^{k}$ are defined by*

$$s_j = \begin{cases} 1 & \text{if } j = -1, \\ 0 & \text{if } j = 0, \\ s_{j-2} - q_{j-1}s_{j-1} & \text{if } j \geqslant 1, \end{cases}$$

$$t_j = \begin{cases} 0 & \text{if } j = -1, \\ 1 & \text{if } j = 0, \\ t_{j-2} - q_{j-1}t_{j-1} & \text{if } j \geqslant 1, \end{cases}$$

then

$$at_j + bs_j = r_j \qquad \forall\, 1 \leqslant j \leqslant k.$$

§6.1 RSA Encryption

Theorem

Let a and b be positive integers with $a \leqslant b$. *Suppose that* $b = aq_0 + r_1$, $a = r_1 q_1 + r_2$, $r_{j-1} = r_j q_j + r_{j+1}$ *for* $2 \leqslant j \leqslant k$, *where* $0 = r_{k+1} < r_k < \cdots < r_2 < r_1 < a$ *and* $q_j \in \mathbb{N}$ *for all* $0 \leqslant j \leqslant k$.

1. $\gcd(a, b) = r_k$, *the last non-zero remainder in the list.*
2. *If* $\{s_j\}_{j=-1}^{k}$ *and* $\{t_j\}_{j=-1}^{k}$ *are defined by*

$$s_j = \begin{cases} 1 & \text{if } j = -1\,, \\ 0 & \text{if } j = 0\,, \\ s_{j-2} - q_{j-1}s_{j-1} & \text{if } j \geqslant 1\,, \end{cases}$$

$$t_j = \begin{cases} 0 & \text{if } j = -1\,, \\ 1 & \text{if } j = 0\,, \\ t_{j-2} - q_{j-1}t_{j-1} & \text{if } j \geqslant 1\,, \end{cases}$$

then

$$at_j + bs_j = r_j \qquad \forall\, 1 \leqslant j \leqslant k\,.$$

§6.1 RSA Encryption

Proof.

Let a and b be positive integers with $a \leqslant b$. By the Division Algorithm, there exists positive integer q_1 and non-negative integer r_1 such that $b = aq_0 + r_1$ and $0 \leqslant r_1 < a$. If $r_1 = 0$, the lists terminate; otherwise, for $0 < r_1 < a$, there exists positive integer q_1 and non-negative integer r_2 such that $a = r_1 q_1 + r_2$ and $0 \leqslant r_2 < r_1$. If $r_2 = 0$, the lists terminate; otherwise, for $0 < r_2 < r_1$, there exists positive integer q_2 and non-negative integer r_3 such that $r_1 = r_2 q_2 + r_3$ and $0 \leqslant r_3 < r_2$. Continuing in this fashion, we obtain a strictly decreasing sequence of non-negative integers $r_1, r_2, r_3, \cdots$. This lists must end, so there is an integer k such that $r_{k+1} = 0$.

Therefore, with r_{-1} and r_0 denoting b and a respectively, we have

$$r_{-1} \geqslant r_0 > r_1 > r_2 > \cdots > r_k > r_{k+1} = 0\,,$$

$$r_{j-1} = r_j q_j + r_{j+1} \quad \text{for all } 0 \leqslant j \leqslant k\,.$$

□

§6.1 RSA Encryption

Proof.

Let a and b be positive integers with $a \leqslant b$. By the Division Algorithm, there exists positive integer q_1 and non-negative integer r_1 such that $b = aq_0 + r_1$ and $0 \leqslant r_1 < a$. If $r_1 = 0$, the lists terminate; otherwise, for $0 < r_1 < a$, there exists positive integer q_1 and non-negative integer r_2 such that $a = r_1 q_1 + r_2$ and $0 \leqslant r_2 < r_1$. If $r_2 = 0$, the lists terminate; otherwise, for $0 < r_2 < r_1$, there exists positive integer q_2 and non-negative integer r_3 such that $r_1 = r_2 q_2 + r_3$ and $0 \leqslant r_3 < r_2$. Continuing in this fashion, we obtain a strictly decreasing sequence of non-negative integers $r_1, r_2, r_3, \cdots$. This lists must end, so there is an integer k such that $r_{k+1} = 0$.

Therefore, with r_{-1} and r_0 denoting b and a respectively, we have

$$r_{-1} \geqslant r_0 > r_1 > r_2 > \cdots > r_k > r_{k+1} = 0\,,$$
$$r_{j-1} = r_j q_j + r_{j+1} \quad \text{for all } 0 \leqslant j \leqslant k\,.$$

□

§6.1 RSA Encryption

Proof (cont'd).

1. We now show that $r_k = d \equiv \gcd(a, b)$.
   (a) First we note that r_k divides r_{k-1} since $r_{k-1} = r_k q_k$. Therefore, the fact that $r_{j-1} = r_j q_j + r_{j+1}$ for all $0 \leqslant j \leqslant k$ implies that r_k divides r_{j-1} for all $0 \leqslant j \leqslant k$.
   (b) On the other hand, d divides r_{-1} and r_0. Therefore, by the fact that $r_{j+1} = r_{j-1} - r_j q_j$ for all $0 \leqslant j \leqslant k$, we find that d divides r_{j+1} for all $0 \leqslant j \leqslant k$.

By (a), r_k is a common divisor of a and b. By (b), the greatest common divisor of a and b must divide r_k; thus we conclude that $r_k = \gcd(a, b)$. □

§6.1 RSA Encryption

Proof (cont'd).

1. We now show that $r_k = d \equiv \gcd(a, b)$.
   (a) First we note that r_k divides r_{k-1} since $r_{k-1} = r_k q_k$. Therefore, the fact that $r_{j-1} = r_j q_j + r_{j+1}$ for all $0 \leqslant j \leqslant k$ implies that r_k divides r_{j-1} for all $0 \leqslant j \leqslant k$.
   (b) On the other hand, d divides r_{-1} and r_0. Therefore, by the fact that $r_{j+1} = r_{j-1} - r_j q_j$ for all $0 \leqslant j \leqslant k$, we find that d divides r_{j+1} for all $0 \leqslant j \leqslant k$.

   By (a), r_k is a common divisor of a and b. By (b), the greatest common divisor of a and b must divide r_k; thus we conclude that $r_k = \gcd(a, b)$. □

§6.1 RSA Encryption

Proof (cont'd).

1. We now show that $r_k = d \equiv \gcd(a, b)$.
   - ⓐ First we note that r_k divides r_{k-1} since $r_{k-1} = r_k q_k$. Therefore, the fact that $r_{j-1} = r_j q_j + r_{j+1}$ for all $0 \leqslant j \leqslant k$ implies that r_k divides r_{j-1} for all $0 \leqslant j \leqslant k$.
   - ⓑ On the other hand, d divides r_{-1} and r_0. Therefore, by the fact that $r_{j+1} = r_{j-1} - r_j q_j$ for all $0 \leqslant j \leqslant k$, we find that d divides r_{j+1} for all $0 \leqslant j \leqslant k$.

   By ⓐ, r_k is a common divisor of a and b. By ⓑ, the greatest common divisor of a and b must divide r_k; thus we conclude that $r_k = \gcd(a, b)$. □

§6.1 RSA Encryption

Proof (cont'd).

2. To see that for all $1 \leqslant j \leqslant k$,

$$at_j + bs_j = r_j\,, \qquad (\star)$$

we note that

ⓐ $(\star)$ holds for the case $k = 1$ since $(s_1, t_1) = (1, -q_0)$ and $b = aq_0 + r_1$.

ⓑ $(\star)$ holds for the case $k = 2$ since $(s_2, t_2) = (-q_1, 1 + q_0q_1)$ and $at_2 + bs_2 = a(1 + q_0q_1) - bq_1 = a - q_1(b - aq_0) = r_0 - q_1r_1 = r_2$.

ⓒ Suppose that $(\star)$ holds for $k = \ell, \ell - 1$, $\ell \geqslant 2$. Then

$$\begin{aligned} at_{\ell+1} + bs_{\ell+1} &= a(t_{\ell-1} - q_\ell t_\ell) + b(s_{\ell-1} - q_\ell s_\ell) \\ &= at_{\ell-1} + bs_{\ell-1} - q_\ell(at_\ell + bs_\ell) \\ &= r_{\ell-1} - q_\ell r_\ell = r_{\ell+1}\,. \end{aligned}$$

By induction, we conclude that $(\star)$ holds for $1 \leqslant j \leqslant k$. □

§6.1 RSA Encryption

Proof (cont'd).

2. To see that for all $1 \leqslant j \leqslant k$,

$$at_j + bs_j = r_j\,, \tag{$\star$}$$

we note that

ⓐ ($\star$) holds for the case $k = 1$ since $(s_1, t_1) = (1, -q_0)$ and $b = aq_0 + r_1$.

ⓑ ($\star$) holds for the case $k = 2$ since $(s_2, t_2) = (-q_1, 1 + q_0 q_1)$ and $at_2 + bs_2 = a(1+q_0 q_1) - bq_1 = a - q_1(b - aq_0) = r_0 - q_1 r_1 = r_2$.

ⓒ Suppose that ($\star$) holds for $k = \ell, \ell - 1$, $\ell \geqslant 2$. Then

$$\begin{aligned} at_{\ell+1} + bs_{\ell+1} &= a(t_{\ell-1} - q_\ell t_\ell) + b(s_{\ell-1} - q_\ell s_\ell) \\ &= at_{\ell-1} + bs_{\ell-1} - q_\ell(at_\ell + bs_\ell) \\ &= r_{\ell-1} - q_\ell r_\ell = r_{\ell+1}\,. \end{aligned}$$

By induction, we conclude that ($\star$) holds for $1 \leqslant j \leqslant k$. □

§6.1 RSA Encryption

Proof (cont'd).

2. To see that for all $1 \leqslant j \leqslant k$,
$$at_j + bs_j = r_j\,, \qquad (\star)$$
we note that
   ⓐ $(\star)$ holds for the case $k = 1$ since $(s_1, t_1) = (1, -q_0)$ and $b = aq_0 + r_1$.
   ⓑ $(\star)$ holds for the case $k = 2$ since $(s_2, t_2) = (-q_1, 1 + q_0 q_1)$ and $at_2 + bs_2 = a(1 + q_0 q_1) - bq_1 = a - q_1(b - aq_0) = r_0 - q_1 r_1 = r_2$.
   ⓒ Suppose that $(\star)$ holds for $k = \ell, \ell - 1$, $\ell \geqslant 2$. Then
$$\begin{aligned} at_{\ell+1} + bs_{\ell+1} &= a(t_{\ell-1} - q_\ell t_\ell) + b(s_{\ell-1} - q_\ell s_\ell) \\ &= at_{\ell-1} + bs_{\ell-1} - q_\ell(at_\ell + bs_\ell) \\ &= r_{\ell-1} - q_\ell r_\ell = r_{\ell+1}\,. \end{aligned}$$
By induction, we conclude that $(\star)$ holds for $1 \leqslant j \leqslant k$. □

§6.1 RSA Encryption

Proof (cont'd).

2. To see that for all $1 \leqslant j \leqslant k$,
$$at_j + bs_j = r_j \,, \tag{$\star$}$$
we note that
   - ⓐ ($\star$) holds for the case $k = 1$ since $(s_1, t_1) = (1, -q_0)$ and $b = aq_0 + r_1$.
   - ⓑ ($\star$) holds for the case $k = 2$ since $(s_2, t_2) = (-q_1, 1 + q_0 q_1)$ and $at_2 + bs_2 = a(1 + q_0 q_1) - bq_1 = a - q_1(b - aq_0) = r_0 - q_1 r_1 = r_2$.
   - ⓒ Suppose that ($\star$) holds for $k = \ell, \ell - 1$, $\ell \geqslant 2$. Then
$$\begin{aligned} at_{\ell+1} + bs_{\ell+1} &= a(t_{\ell-1} - q_\ell t_\ell) + b(s_{\ell-1} - q_\ell s_\ell) \\ &= at_{\ell-1} + bs_{\ell-1} - q_\ell(at_\ell + bs_\ell) \\ &= r_{\ell-1} - q_\ell r_\ell = r_{\ell+1} \,. \end{aligned}$$

By induction, we conclude that ($\star$) holds for $1 \leqslant j \leqslant k$. □

§6.1 RSA Encryption

Proof (cont'd).

2. To see that for all $1 \leqslant j \leqslant k$,

$$at_j + bs_j = r_j\,, \qquad (\star)$$

we note that

ⓐ $(\star)$ holds for the case $k = 1$ since $(s_1, t_1) = (1, -q_0)$ and $b = aq_0 + r_1$.

ⓑ $(\star)$ holds for the case $k = 2$ since $(s_2, t_2) = (-q_1, 1 + q_0 q_1)$ and $at_2 + bs_2 = a(1+q_0 q_1) - bq_1 = a - q_1(b - aq_0) = r_0 - q_1 r_1 = r_2$.

ⓒ Suppose that $(\star)$ holds for $k = \ell, \ell - 1$, $\ell \geqslant 2$. Then

$$\begin{aligned} at_{\ell+1} + bs_{\ell+1} &= a(t_{\ell-1} - q_\ell t_\ell) + b(s_{\ell-1} - q_\ell s_\ell) \\ &= at_{\ell-1} + bs_{\ell-1} - q_\ell(at_\ell + bs_\ell) \\ &= r_{\ell-1} - q_\ell r_\ell = r_{\ell+1}\,. \end{aligned}$$

By induction, we conclude that $(\star)$ holds for $1 \leqslant j \leqslant k$. □

§6.1 RSA Encryption

Remark: Let $a, b \in \mathbb{N}$ with $a \leqslant b$. The algorithm to compute $\gcd(a, b)$ given in part 1 of the previous theorem is caleed **Euclid's Algorithm** (輾轉相除法), and the algorithm to compute $x, y \in \mathbb{Z}$ so that $ax + by = \gcd(a, b)$ given in part 2 of the previous theorem is called **Extended Euclid's Algorithm**.

§6.1 RSA Encryption

Example

We compute $\gcd(32, 12)$ using Euclid's algorithm as follows:

$$32 = 12 \times 2 + 8\,, \quad 12 = 8 \times 1 + 4\,, \quad 8 = 4 \times 2 + 0\,.$$

Therefore, $4 = \gcd(12, 32)$. Moreover, by working backward,

$$4 = 12 - 8 \times 1 = 12 - (32 - 12 \times 2) \times 1 = 12 \times 3 + 32 \times (-1)\,.$$

One can also obtain the "coefficients" 3 and -1 using Extended Euclid's Algorithm:

j	r_j	q_j	s_j	t_j
-1	32		1	0
0	12	2	0	1
1	8	1	1	-2
2	4	2	-1	3

§6.1 RSA Encryption

Example

We compute $\gcd(32, 12)$ using Euclid's algorithm as follows:

$$32 = 12 \times 2 + 8\,, \quad 12 = 8 \times 1 + 4\,, \quad 8 = 4 \times 2 + 0\,.$$

Therefore, $4 = \gcd(12, 32)$. Moreover, by working backward,

$$4 = 12 - 8 \times 1 = 12 - (32 - 12 \times 2) \times 1 = 12 \times 3 + 32 \times (-1)\,.$$

One can also obtain the "coefficients" 3 and -1 using Extended Euclid's Algorithm:

j	r_j	q_j	s_j	t_j
-1	32		1	0
0	12	2	0	1
1	8	1	1	-2
2	4	2	-1	3

§6.1 RSA Encryption

Example

We compute $\gcd(32, 12)$ using Euclid's algorithm as follows:

$$32 = 12 \times 2 + 8\,, \quad 12 = 8 \times 1 + 4\,, \quad 8 = 4 \times 2 + 0\,.$$

Therefore, $4 = \gcd(12, 32)$. Moreover, by working backward,

$$4 = 12 - 8 \times 1 = 12 - (32 - 12 \times 2) \times 1 = 12 \times 3 + 32 \times (-1)\,.$$

One can also obtain the "coefficients" 3 and -1 using Extended Euclid's Algorithm:

j	r_j	q_j	s_j	t_j
-1	32		1	0
0	12	2	0	1
1	8	1	1	-2
2	4	2	-1	3

§6.1 RSA Encryption

Theorem

Let a and b be non-zero integers. The gcd of a and b is the smallest positive linear combination of a and b; that is,

$$\gcd(a, b) = \min\{am + bn \,|\, am + bn > 0\,, m, n \in \mathbb{Z}\}\,.$$

Proof.

Let $d = am + bn$ be the smallest positive linear combination of a and b.

1. By the Division Algorithm, there exist integers q and r such that $a = dq + r$, where $0 \leqslant r < d$. Then

$$r = a - dq = a - (am + bn)q = a(1 - m) + b(-nq)\,;$$

thus r is a linear combination of a and b. Since $0 \leqslant r < d$, we must have $r = 0$. Therefore, $a = dq$; thus $d|a$. Similarly, $d|b$; thus d is a common divisor of a and b. □

§6.1 RSA Encryption

Theorem

Let a and b be non-zero integers. The gcd of a and b is the smallest positive linear combination of a and b; that is,

$$\gcd(a, b) = \min\{am + bn \mid am + bn > 0\,, m, n \in \mathbb{Z}\}\,.$$

Proof.

Let $d = am + bn$ be the smallest positive linear combination of a and b.

1. By the Division Algorithm, there exist integers q and r such that $a = dq + r$, where $0 \leqslant r < d$. Then

$$r = a - dq = a - (am + bn)q = a(1 - m) + b(-nq)\,;$$

thus r is a linear combination of a and b. Since $0 \leqslant r < d$, we must have $r = 0$. Therefore, $a = dq$; thus $d|a$. Similarly, $d|b$; thus d is a common divisor of a and b. □

§6.1 RSA Encryption

Proof (cont'd).

2. Let c be a common divisor of a and b. Then c divides d since $d = am + bn$. Therefore, $c \leqslant d$.

By ① and ②, we find that $d = \gcd(a, b)$. □

Definition (Euler function)

Let $n \in \mathbb{N}$. The function $\varphi : \mathbb{N} \to \mathbb{N}$ defined by

$$\varphi(n) = \#\left\{k \in \mathbb{N} \,\middle|\, 1 \leqslant k \leqslant n \text{ and } \gcd(k, n) = 1\right\}$$

is called the Euler (phi) function. In other words, the Euler function counts the positive integers up to a given integer n that are coprime to n.

§6.1 RSA Encryption

Proof (cont'd).

2. Let c be a common divisor of a and b. Then c divides d since $d = am + bn$. Therefore, $c \leqslant d$.

By ① and ②, we find that $d = \gcd(a, b)$. □

Definition (Euler function)

Let $n \in \mathbb{N}$. The function $\varphi : \mathbb{N} \to \mathbb{N}$ defined by

$$\varphi(n) = \#\{k \in \mathbb{N} \,|\, 1 \leqslant k \leqslant n \text{ and } \gcd(k, n) = 1\}$$

is called the Euler (phi) function. In other words, the Euler function counts the positive integers up to a given integer n that are coprime to n.

§6.1 RSA Encryption

Proof (cont'd).

2. Let c be a common divisor of a and b. Then c divides d since $d = am + bn$. Therefore, $c \leqslant d$.

By ① and ②, we find that $d = \gcd(a, b)$. □

Definition (Euler function)

Let $n \in \mathbb{N}$. The function $\varphi : \mathbb{N} \to \mathbb{N}$ defined by

$$\varphi(n) = \#\{k \in \mathbb{N} \,|\, 1 \leqslant k \leqslant n \text{ and } \gcd(k, n) = 1\}$$

is called the Euler (phi) function. In other words, the Euler function counts the positive integers up to a given integer n that are coprime to n.

§6.1 RSA Encryption

Proposition

For each $n \in \mathbb{N}$,

$$\varphi(n) = n \prod_{\substack{p|n \\ p \text{ prime}}} \left(1 - \frac{1}{p}\right).$$

In particular, by writing $n = \prod_{j=1}^{r} p_j^{k_j} = p_1^{k_1} p_2^{k_2} \cdots p_r^{k_r}$, *where* $p_1, \cdots, p_r$ *are distinct prime numbers and* $k_1, \cdots, k_r \in \mathbb{N}$, *one has*

$$\varphi(n) = \prod_{j=1}^{r} p_j^{k_j - 1}(p_j - 1).$$

Corollary

Let $m, n \in \mathbb{N}$ *be such that* $\gcd(m, n) = 1$. *Then* $\varphi(mn) = \varphi(m)\varphi(n)$.

§6.1 RSA Encryption

Proposition

For each $n \in \mathbb{N}$,

$$\varphi(n) = n \prod_{\substack{p|n \\ p \text{ prime}}} \left(1 - \frac{1}{p}\right).$$

In particular, by writing $n = \prod_{j=1}^{r} p_j^{k_j} = p_1^{k_1} p_2^{k_2} \cdots p_r^{k_r}$, *where* $p_1, \cdots, p_r$ *are distinct prime numbers and* $k_1, \cdots, k_r \in \mathbb{N}$, *one has*

$$\varphi(n) = \prod_{j=1}^{r} p_j^{k_j - 1}(p_j - 1).$$

Corollary

Let $m, n \in \mathbb{N}$ *be such that* $\gcd(m, n) = 1$. *Then* $\varphi(mn) = \varphi(m)\varphi(n)$.

§6.1 RSA Encryption

Proposition

For each $n \in \mathbb{N}$,

$$\varphi(n) = n \prod_{\substack{p|n \\ p \text{ prime}}} \left(1 - \frac{1}{p}\right).$$

In particular, by writing $n = \prod_{j=1}^{r} p_j^{k_j} = p_1^{k_1} p_2^{k_2} \cdots p_r^{k_r}$, *where* $p_1, \cdots, p_r$ *are distinct prime numbers and* $k_1, \cdots, k_r \in \mathbb{N}$, *one has*

$$\varphi(n) = \prod_{j=1}^{r} p_j^{k_j - 1}(p_j - 1).$$

Corollary

Let $m, n \in \mathbb{N}$ *be such that* $\gcd(m, n) = 1$. *Then* $\varphi(mn) = \varphi(m)\varphi(n)$.

§6.1 RSA Encryption

Definition

Given $a \in \mathbb{Z}$ and $n \in \mathbb{N}$, a modulo n (abbreviated as a mod n) is the remainder of the Euclidean division of a by n. In other words, a mod n outputs r if $a = qn + r$ for some $q \in \mathbb{Z}$ and $r \in \{0, 1, \cdots, n-1\}$. For $a, b \in \mathbb{Z}$, the notation $a \equiv b \pmod{n}$ denotes the fact that $n|(a-b)$; that is, there exists $m \in \mathbb{Z}$ such that $a - b = mn$.

Definition

The addition $\oplus$ on $\mathbb{Z}_n$ is defined by

$$c = a \oplus b \qquad \text{if and only if} \qquad (a+b) \text{ mod } n \text{ outputs } c,$$

and the multiplication $\odot$ on $\mathbb{Z}_n$ is defined by

$$c = a \odot b \qquad \text{if and only if} \qquad (a \cdot b) \text{ mod } n \text{ outputs } c,$$

where $+$ and $\cdot$ are the usual addition and multiplication on $\mathbb{Z}$.

§6.1 RSA Encryption

Definition

Given $a \in \mathbb{Z}$ and $n \in \mathbb{N}$, a modulo n (abbreviated as a mod n) is the remainder of the Euclidean division of a by n. In other words, a mod n outputs r if $a = qn + r$ for some $q \in \mathbb{Z}$ and $r \in \{0, 1, \cdots, n-1\}$. For $a, b \in \mathbb{Z}$, the notation $a \equiv b \pmod{n}$ denotes the fact that $n|(a-b)$; that is, there exists $m \in \mathbb{Z}$ such that $a - b = mn$.

Definition

The addition $\oplus$ on $\mathbb{Z}_n$ is defined by

$$c = a \oplus b \qquad \text{if and only if} \qquad (a+b) \text{ mod } n \text{ outputs } c,$$

and the multiplication $\odot$ on $\mathbb{Z}_n$ is defined by

$$c = a \odot b \qquad \text{if and only if} \qquad (a \cdot b) \text{ mod } n \text{ outputs } c,$$

where $+$ and $\cdot$ are the usual addition and multiplication on $\mathbb{Z}$.

§6.1 RSA Encryption

Definition

Given $a \in \mathbb{Z}$ and $n \in \mathbb{N}$, a modulo n (abbreviated as a mod n) is the remainder of the Euclidean division of a by n. In other words, a mod n outputs r if $a = qn + r$ for some $q \in \mathbb{Z}$ and $r \in \{0, 1, \cdots, n-1\}$. For $a, b \in \mathbb{Z}$, the notation $a \equiv b \pmod{n}$ denotes the fact that $n \big| (a-b)$; that is, there exists $m \in \mathbb{Z}$ such that $a - b = mn$.

Definition

The addition $\oplus$ on $\mathbb{Z}_n$ is defined by

$$c = a \oplus b \qquad \text{if and only if} \qquad (a+b) \text{ mod } n \text{ outputs } c\,,$$

and the multiplication $\odot$ on $\mathbb{Z}_n$ is defined by

$$c = a \odot b \qquad \text{if and only if} \qquad (a \cdot b) \text{ mod } n \text{ outputs } c\,,$$

where $+$ and $\cdot$ are the usual addition and multiplication on $\mathbb{Z}$.

§6.1 RSA Encryption

Proposition

$(\mathbb{Z}_n, \oplus)$ *is a group; that is,*

1. $\mathbb{Z}_n$ *is closed under addition* $\oplus$;
2. *there exists an additive identity* 0 (*that is,* $a \oplus 0 = a$ *for all* $a \in \mathbb{Z}_n$), *and*
3. *every element in* $\mathbb{Z}_n$ *has an additive inverse* (*that is, for each* $a \in \mathbb{Z}_n$ *there exists* $b \in \mathbb{Z}_n$ *such that* $a \oplus b = 0$).

Proposition

Let $a, b, c, d \in \mathbb{Z}$ *and* $n \in \mathbb{N}$ *be such that* $a \equiv c \pmod{n}$ *and* $b \equiv d \pmod{n}$. *Then* $a \cdot b \equiv c \cdot d \pmod{n}$.

Proposition (Cancellation law in $\mathbb{Z}_n$)

Let $a, n \in \mathbb{N}$ *be such that* $\gcd(a, n) = 1$. *If* $a \cdot b \equiv a \cdot c \pmod{n}$, *then* $b \equiv c \pmod{n}$.

§6.1 RSA Encryption

Proposition

$(\mathbb{Z}_n, \oplus)$ *is a group; that is,*

1. $\mathbb{Z}_n$ *is closed under addition* $\oplus$*;*
2. *there exists an additive identity* 0 (*that is,* $a \oplus 0 = a$ *for all* $a \in \mathbb{Z}_n$), *and*
3. *every element in* $\mathbb{Z}_n$ *has an additive inverse* (*that is, for each* $a \in \mathbb{Z}_n$ *there exists* $b \in \mathbb{Z}_n$ *such that* $a \oplus b = 0$).

Proposition

Let $a, b, c, d \in \mathbb{Z}$ *and* $n \in \mathbb{N}$ *be such that* $a \equiv c \pmod{n}$ *and* $b \equiv d \pmod{n}$. *Then* $a \cdot b \equiv c \cdot d \pmod{n}$.

Proposition (Cancellation law in $\mathbb{Z}_n$)

Let $a, n \in \mathbb{N}$ *be such that* $\gcd(a, n) = 1$. *If* $a \cdot b \equiv a \cdot c \pmod{n}$, *then* $b \equiv c \pmod{n}$.

§6.1 RSA Encryption

Proposition

$(\mathbb{Z}_n, \oplus)$ *is a group; that is,*

1. $\mathbb{Z}_n$ *is closed under addition* $\oplus$*;*
2. *there exists an additive identity* 0 *(that is,* $a \oplus 0 = a$ *for all* $a \in \mathbb{Z}_n$*), and*
3. *every element in* $\mathbb{Z}_n$ *has an additive inverse (that is, for each* $a \in \mathbb{Z}_n$ *there exists* $b \in \mathbb{Z}_n$ *such that* $a \oplus b = 0$*).*

Proposition

Let $a, b, c, d \in \mathbb{Z}$ *and* $n \in \mathbb{N}$ *be such that* $a \equiv c \pmod n$ *and* $b \equiv d \pmod n$*. Then* $a \cdot b \equiv c \cdot d \pmod n$*.*

Proposition (Cancellation law in $\mathbb{Z}_n$)

Let $a, n \in \mathbb{N}$ *be such that* $\gcd(a, n) = 1$*. If* $a \cdot b \equiv a \cdot c \pmod n$*, then* $b \equiv c \pmod n$*.*

§6.1 RSA Encryption

Proposition

Let $n \geqslant 2$ be an integer, and $a, b \in \mathbb{Z}$ satisfy $a \equiv b \pmod{n}$. Then $\gcd(a, n) = 1$ if and only if $\gcd(b, n) = 1$.

Proof.

It suffices to shows that if $\gcd(a, n) \neq 1$, then $\gcd(b, n) \neq 1$. Suppose that $\gcd(a, n) = p > 1$. Then $a = pq_1$ and $n = pq_2$ for some $q_1, q_2 \in \mathbb{Z}$. Since $a \equiv b \pmod{n}$, there exists $m \in \mathbb{Z}$ such that $a - b = mn$. Therefore, $b = a - mn = pq_1 - pq_2m = p(q_1 - q_2m)$ which shows that $\gcd(b, n) \geqslant p$. □

The proposition above shows that if $a \in \mathbb{Z}$ satisfies $\gcd(a, n) = 1$, then $(a \bmod n)$ is coprime to n.

§6.1 RSA Encryption

Proposition

Let $n \geqslant 2$ be an integer, and $a, b \in \mathbb{Z}$ satisfy $a \equiv b \pmod{n}$. Then $\gcd(a, n) = 1$ if and only if $\gcd(b, n) = 1$.

Proof.

It suffices to shows that if $\gcd(a, n) \neq 1$, then $\gcd(b, n) \neq 1$.

Suppose that $\gcd(a, n) = p > 1$. Then $a = pq_1$ and $n = pq_2$ for some $q_1, q_2 \in \mathbb{Z}$. Since $a \equiv b \pmod{n}$, there exists $m \in \mathbb{Z}$ such that $a - b = mn$. Therefore, $b = a - mn = pq_1 - pq_2m = p(q_1 - q_2m)$ which shows that $\gcd(b, n) \geqslant p$. □

The proposition above shows that if $a \in \mathbb{Z}$ satisfies $\gcd(a, n) = 1$, then $(a \bmod n)$ is coprime to n.

§6.1 RSA Encryption

Proposition

Let $n \geqslant 2$ be an integer, and $a, b \in \mathbb{Z}$ satisfy $a \equiv b \pmod{n}$. Then $\gcd(a, n) = 1$ if and only if $\gcd(b, n) = 1$.

Proof.

It suffices to shows that if $\gcd(a, n) \neq 1$, then $\gcd(b, n) \neq 1$. Suppose that $\gcd(a, n) = p > 1$. Then $a = pq_1$ and $n = pq_2$ for some $q_1, q_2 \in \mathbb{Z}$. Since $a \equiv b \pmod{n}$, there exists $m \in \mathbb{Z}$ such that $a - b = mn$. Therefore, $b = a - mn = pq_1 - pq_2 m = p(q_1 - q_2 m)$ which shows that $\gcd(b, n) \geqslant p$. □

The proposition above shows that if $a \in \mathbb{Z}$ satisfies $\gcd(a, n) = 1$, then $(a \bmod n)$ is coprime to n.

§6.1 RSA Encryption

Proposition

Let $n \geqslant 2$ be an integer, and $a, b \in \mathbb{Z}$ satisfy $a \equiv b \pmod{n}$. Then $\gcd(a, n) = 1$ if and only if $\gcd(b, n) = 1$.

Proof.

It suffices to shows that if $\gcd(a, n) \neq 1$, then $\gcd(b, n) \neq 1$. Suppose that $\gcd(a, n) = p > 1$. Then $a = pq_1$ and $n = pq_2$ for some $q_1, q_2 \in \mathbb{Z}$. Since $a \equiv b \pmod{n}$, there exists $m \in \mathbb{Z}$ such that $a - b = mn$. Therefore, $b = a - mn = pq_1 - pq_2 m = p(q_1 - q_2 m)$ which shows that $\gcd(b, n) \geqslant p$. □

The proposition above shows that if $a \in \mathbb{Z}$ satisfies $\gcd(a, n) = 1$, then $(a \bmod n)$ is coprime to n.

§6.1 RSA Encryption

Proposition

Let $n \geqslant 2$ be an integer, and $a, b \in \mathbb{Z}$ satisfy $a \equiv b \pmod{n}$. Then $\gcd(a, n) = 1$ if and only if $\gcd(b, n) = 1$.

Proof.

It suffices to shows that if $\gcd(a, n) \neq 1$, then $\gcd(b, n) \neq 1$. Suppose that $\gcd(a, n) = p > 1$. Then $a = pq_1$ and $n = pq_2$ for some $q_1, q_2 \in \mathbb{Z}$. Since $a \equiv b \pmod{n}$, there exists $m \in \mathbb{Z}$ such that $a - b = mn$. Therefore, $b = a - mn = pq_1 - pq_2 m = p(q_1 - q_2 m)$ which shows that $\gcd(b, n) \geqslant p$. □

The proposition above shows that if $a \in \mathbb{Z}$ satisfies $\gcd(a, n) = 1$, then $(a \bmod n)$ is coprime to n.

§6.1 RSA Encryption

Theorem

The integers coprime to n from the set $\{0, 1, \cdots, n-1\}$ of n non-negative integers form a group under multiplication modulo n. In other words, let S be a subset of $\mathbb{Z}_n$ consisting of numbers coprime to n; that is, $S = \{k \in \mathbb{N} \,|\, 1 \leqslant k \leqslant n \text{ and } \gcd(k, n) = 1\}$. Then $(S, \odot)$ is a group; that is,

1. *S is closed under multiplication $\odot$;*
2. *there exists an multiplicative identity 1 (that is, $a \odot 1 = a$ for all $a \in S$), and*
3. *every element in S has an multiplicative inverse element (that is, for each $a \in S$ there exists $b \in S$ such that $a \odot b = 1$).*

§6.1 RSA Encryption

Theorem

The integers coprime to n from the set $\{0, 1, \cdots, n-1\}$ *of n non-negative integers form a group under multiplication modulo n. In other words, let S be a subset of* $\mathbb{Z}_n$ *consisting of numbers coprime to n; that is,* $S = \{k \in \mathbb{N} \mid 1 \leqslant k \leqslant n \text{ and } \gcd(k, n) = 1\}$. *Then* $(S, \odot)$ *is a group; that is,*

1. *S is closed under multiplication* $\odot$;
2. *there exists an multiplicative identity* 1 *(that is,* $a \odot 1 = a$ *for all* $a \in S$*), and*
3. *every element in S has an multiplicative inverse element (that is, for each* $a \in S$ *there exists* $b \in S$ *such that* $a \odot b = 1$*).*

§6.1 RSA Encryption

Proof.

It suffices to prove 1 and 3.

1. Let $a, b \in S$. Then $a \cdot b$ is coprime to n; thus the previous proposition implies that $a \cdot b \bmod n$ is coprime to n as well. Therefore, $a \odot b \in S$.
3. Let $a \in S$. Then the set $a \odot S \equiv \{a \odot s \mid s \in S\}$ is a subset of S. Moreover, if $s_1, s_2 \in S$ satisfying that $a \odot s_1 = a \odot s_2$; that is, $a \cdot s_1 \equiv a \cdot s_2 \pmod{n}$, then $s_1 = s_2$; thus $\#(a \odot S) = \varphi(n)$. This fact shows that there exists $s \in S$ such that $a \odot s = 1$. □

§6.1 RSA Encryption

Proof.

It suffices to prove 1 and 3.

1. Let $a, b \in S$. Then $a \cdot b$ is coprime to n; thus the previous proposition implies that $a \cdot b$ mod n is coprime to n as well. Therefore, $a \odot b \in S$.
3. Let $a \in S$. Then the set $a \odot S \equiv \{a \odot s \,|\, s \in S\}$ is a subset of S. Moreover, if $s_1, s_2 \in S$ satisfying that $a \odot s_1 = a \odot s_2$; that is, $a \cdot s_1 \equiv a \cdot s_2 \pmod{n}$, then $s_1 = s_2$; thus $\#(a \odot S) = \varphi(n)$. This fact shows that there exists $s \in S$ such that $a \odot s = 1$. □

§6.1 RSA Encryption

Proof.

It suffices to prove 1 and 3.

1. Let $a, b \in S$. Then $a \cdot b$ is coprime to n; thus the previous proposition implies that $a \cdot b \bmod n$ is coprime to n as well. Therefore, $a \odot b \in S$.
3. Let $a \in S$. Then the set $a \odot S \equiv \{a \odot s \,|\, s \in S\}$ is a subset of S. Moreover, if $s_1, s_2 \in S$ satisfying that $a \odot s_1 = a \odot s_2$; that is, $a \cdot s_1 \equiv a \cdot s_2 \pmod{n}$, then $s_1 = s_2$; thus $\#(a \odot S) = \varphi(n)$. This fact shows that there exists $s \in S$ such that $a \odot s = 1$. □

§6.1 RSA Encryption

Definition

The multiplicative group of integers modulo n (given in the previous theorem) is denoted by $(\mathbb{Z}_n^*, \odot)$.

Theorem

Let $n \in \mathbb{N}$ and $a \in \mathbb{Z}_n^*$. If $a \cdot x + n \cdot y = 1$ for some $x, y \in \mathbb{Z}$, then

$$a^{-1} \equiv x \pmod{n},$$

where a^{-1} denotes the unique number in $\mathbb{Z}_n^*$ satisfying

$$a \odot a^{-1} = a^{-1} \odot a = 1.$$

§6.1 RSA Encryption

Definition

The multiplicative group of integers modulo n (given in the previous theorem) is denoted by $(\mathbb{Z}_n^*, \odot)$.

Theorem

Let $n \in \mathbb{N}$ and $a \in \mathbb{Z}_n^*$. If $a \cdot x + n \cdot y = 1$ for some $x, y \in \mathbb{Z}$, then

$$a^{-1} \equiv x \pmod{n},$$

where a^{-1} denotes the unique number in $\mathbb{Z}_n^*$ satisfying

$$a \odot a^{-1} = a^{-1} \odot a = 1.$$

§6.1 RSA Encryption

Theorem

Let $a, n \in \mathbb{N}$ *be such that* $\gcd(a, n) = 1$. *Then* $a^{\varphi(n)} \equiv 1 \pmod{n}$.

Proof.

Let $a\mathbb{Z}_n^*$ be the set $a\mathbb{Z}_n^* \equiv \{a \cdot s \,|\, s \in \mathbb{Z}_n^*\}$. Then the set $a\mathbb{Z}_n^*$ mod $n \equiv \{(a \cdot s) \bmod n \,|\, s \in \mathbb{Z}_n^*\}$ is identical to $\mathbb{Z}_n^*$. Therefore,

$$\prod_{k \in \mathbb{Z}_n^*} k \equiv \prod_{k \in a\mathbb{Z}_n^*} k \pmod{n}.$$

Since $\prod_{k \in a\mathbb{Z}_n^*} k = a^{\varphi(n)} \prod_{k \in \mathbb{Z}_n^*} k$ and $\prod_{k \in \mathbb{Z}_n^*} k$ is coprime to n, by the cancellation law for $\mathbb{Z}_n$ we find that $a^{\varphi(n)} \equiv 1 \pmod{n}$. □

§6.1 RSA Encryption

Theorem

Let $a, n \in \mathbb{N}$ *be such that* $\gcd(a, n) = 1$. *Then* $a^{\varphi(n)} \equiv 1 \pmod{n}$.

Proof.

Let $a\mathbb{Z}_n^*$ be the set $a\mathbb{Z}_n^* \equiv \{a \cdot s \,|\, s \in \mathbb{Z}_n^*\}$. Then the set $a\mathbb{Z}_n^*$ mod $n \equiv \{(a \cdot s) \text{ mod } n \,|\, s \in \mathbb{Z}_n^*\}$ is identical to $\mathbb{Z}_n^*$. Therefore,

$$\prod_{k \in \mathbb{Z}_n^*} k \equiv \prod_{k \in a\mathbb{Z}_n^*} k \pmod{n}.$$

Since $\prod_{k \in a\mathbb{Z}_n^*} k = a^{\varphi(n)} \prod_{k \in \mathbb{Z}_n^*} k$ and $\prod_{k \in \mathbb{Z}_n^*} k$ is coprime to n, by the cancellation law for $\mathbb{Z}_n$ we find that $a^{\varphi(n)} \equiv 1 \pmod{n}$. □

§6.1 RSA Encryption

Corollary (Fermat little theorem)

Let p be a prime number, and $a \in \mathbb{N}$ satisfy $\gcd(a, p) = 1$. Then $a^{p-1} \equiv 1 \pmod{p}$.

§6.1 RSA Encryption

§6.1.2 Encryption based on factoring large numbers

The RSA algorithm involves four steps: key generation, key distribution, encryption, and decryption.

- **Key generation**: The keys for the RSA algorithm are generated in the following way:

1. Choose two distinct prime numbers p and q.
   (a) For security purposes, p and q should be chosen at random and should be similar in magnitude but differ in length by a few digits to make factoring harder.
   (b) p and q are kept secret.
2. Compute $n = pq$.
   (a) n is used as the modulus for both the public and private keys. Its length, usually expressed in bits, is the key length.
   (b) n is released as part of the public key.

§6.1 RSA Encryption

§6.1.2 Encryption based on factoring large numbers

The RSA algorithm involves four steps: key generation, key distribution, encryption, and decryption.

- **Key generation**: The keys for the RSA algorithm are generated in the following way:

1. Choose two distinct prime numbers p and q.
   (a) For security purposes, p and q should be chosen at random and should be similar in magnitude but differ in length by a few digits to make factoring harder.
   (b) p and q are kept secret.
2. Compute $n = pq$.
   (a) n is used as the modulus for both the public and private keys. Its length, usually expressed in bits, is the key length.
   (b) n is released as part of the public key.

§6.1 RSA Encryption

§6.1.2 Encryption based on factoring large numbers

The RSA algorithm involves four steps: key generation, key distribution, encryption, and decryption.

- **Key generation**: The keys for the RSA algorithm are generated in the following way:
  1. Choose two distinct prime numbers p and q.
     - (a) For security purposes, p and q should be chosen at random and should be similar in magnitude but differ in length by a few digits to make factoring harder.
     - (b) p and q are kept secret.
  2. Compute $n = pq$.
     - (a) n is used as the modulus for both the public and private keys. Its length, usually expressed in bits, is the key length.
     - (b) n is released as part of the public key.

§6.1 RSA Encryption

§6.1.2 Encryption based on factoring large numbers

The RSA algorithm involves four steps: key generation, key distribution, encryption, and decryption.

- **Key generation**: The keys for the RSA algorithm are generated in the following way:

1. Choose two distinct prime numbers p and q.
   (a) For security purposes, p and q should be chosen at random and should be similar in magnitude but differ in length by a few digits to make factoring harder.
   (b) p and q are kept secret.
2. Compute $n = pq$.
   (a) n is used as the modulus for both the public and private keys. Its length, usually expressed in bits, is the key length.
   (b) n is released as part of the public key.

§6.1 RSA Encryption

3. Compute $\varphi(n)$, where φ is the Euler function. By previous proposition, $\varphi(n) = (p-1)(q-1)$. $\varphi(n)$ is kept secret.
4. Choose an integer e such that $1 < e < \varphi(n)$ and $\gcd(e, \varphi(n)) = 1$; that is, e and $\varphi(n)$ are coprime.
   (a) e having a short bit-length and small Hamming weight results in more efficient encryption - the most commonly chosen value for e is $2^{16} + 1 = 65537$. The smallest (and fastest) possible value for e is 3, but such a small value for e has been shown to be less secure in some settings.
   (b) e is released as part of the public key.

§6.1 RSA Encryption

3. Compute $\varphi(n)$, where φ is the Euler function. By previous proposition, $\varphi(n) = (p-1)(q-1)$. $\varphi(n)$ is kept secret.
4. Choose an integer e such that $1 < e < \varphi(n)$ and $\gcd(e, \varphi(n)) = 1$; that is, e and $\varphi(n)$ are coprime.
   - (a) e having a short bit-length and small Hamming weight results in more efficient encryption - the most commonly chosen value for e is $2^{16} + 1 = 65537$. The smallest (and fastest) possible value for e is 3, but such a small value for e has been shown to be less secure in some settings.
   - (b) e is released as part of the public key.

§6.1 RSA Encryption

5. Determine d as $d \equiv e^{-1} \pmod{\varphi(n)}$; that is, d is the modular multiplicative inverse of e modulo $\varphi(n)$.
   - (a) This means: solve for d the equation $d \cdot e \equiv 1 \pmod{\varphi(n)}$; d can be computed efficiently by using the extended Euclidean algorithm.
   - (b) d is kept secret as the private key exponent.

The public key consists of the modulus n and the public (or encryption) exponent e. The private key consists of the private (or decryption) exponent d, which must be kept secret. p, q, and $\varphi(n)$ must also be kept secret because they can be used to calculate d. In fact, they can all be discarded after d has been computed.

§6.1 RSA Encryption

5. Determine d as $d \equiv e^{-1} \pmod{\varphi(n)}$; that is, d is the modular multiplicative inverse of e modulo $\varphi(n)$.
   - (a) This means: solve for d the equation $d \cdot e \equiv 1 \pmod{\varphi(n)}$; d can be computed efficiently by using the extended Euclidean algorithm.
   - (b) d is kept secret as the private key exponent.

The public key consists of the modulus n and the public (or encryption) exponent e. The private key consists of the private (or decryption) exponent d, which must be kept secret. p, q, and $\varphi(n)$ must also be kept secret because they can be used to calculate d. In fact, they can all be discarded after d has been computed.

§6.1 RSA Encryption

Remark:

1. In modern RSA implementation the use of Euler function φ is replaced by Carmichael's totient function λ defined by
$$\lambda(n) = \min\left\{k \in \mathbb{N} \,\middle|\, a^k \equiv 1 \pmod{n} \text{ for all } a \in \mathbb{Z}_n^*\right\}.$$
If p and q are prime numbers and $n = pq$, then
$$\lambda(n) = \mathrm{lcm}(p-1, q-1),$$
the least common multiple（最小公倍數）of $p-1$ and $q-1$.

2. If both n and $\varphi(n)$ are known, then two primes p and q satisfying
$$n = pq, \quad \varphi(n) = (p-1)(q-1)$$
can be solved easily since p and q are zeros of
$$x^2 + \left[\varphi(n) - (n+1)\right]x + n = 0.$$

§6.1 RSA Encryption

Remark:

1. In modern RSA implementation the use of Euler function φ is replaced by Carmichael's totient function λ defined by
$$\lambda(n) = \min\left\{k \in \mathbb{N} \,\middle|\, a^k \equiv 1 \pmod{n} \text{ for all } a \in \mathbb{Z}_n^*\right\}.$$
If p and q are prime numbers and $n = pq$, then
$$\lambda(n) = \mathrm{lcm}(p-1, q-1),$$
the least common multiple（最小公倍數）of $p-1$ and $q-1$.

2. If both n and $\varphi(n)$ are known, then two primes p and q satisfying
$$n = pq, \quad \varphi(n) = (p-1)(q-1)$$
can be solved easily since p and q are zeros of
$$x^2 + \big[\varphi(n) - (n+1)\big]x + n = 0.$$

§6.1 RSA Encryption

- **Key distribution**: Suppose that Bob wants to send information to Alice. To enable Bob to send his encrypted messages, Alice transmits her public key (n, e) to Bob via a reliable, but not necessarily secret, route. Alice's private key (d) is never distributed.

- **Encryption**: After obtaining Alice's public key, Bob first turns the message M into an integer m, such that $0 \leqslant m < n$. He then computes the ciphertext c using Alice's public key e by

$$c \equiv m^e \pmod{n}.$$

This can be done reasonably quickly, even for very large numbers, using modular exponentiation. Bob then transmits c to Alice. Note that some values of m will yield a ciphertext c equal to m, but this is very unlikely to occur in practice.

§6.1 RSA Encryption

- **Key distribution**: Suppose that Bob wants to send information to Alice. To enable Bob to send his encrypted messages, Alice transmits her public key (n, e) to Bob via a reliable, but not necessarily secret, route. Alice's private key (d) is never distributed.

- **Encryption**: After obtaining Alice's public key, Bob first turns the message M into an integer m, such that $0 \leqslant m < n$. He then computes the ciphertext c using Alice's public key e by

$$c \equiv m^e \pmod{n}.$$

This can be done reasonably quickly, even for very large numbers, using modular exponentiation. Bob then transmits c to Alice. Note that some values of m will yield a ciphertext c equal to m, but this is very unlikely to occur in practice.

§6.1 RSA Encryption

- **Key distribution**: Suppose that Bob wants to send information to Alice. To enable Bob to send his encrypted messages, Alice transmits her public key (n, e) to Bob via a reliable, but not necessarily secret, route. Alice's private key (d) is never distributed.

- **Encryption**: After obtaining Alice's public key, Bob first turns the message M into an integer m, such that $0 \leqslant m < n$. He then computes the ciphertext c using Alice's public key e by

$$c \equiv m^e \pmod{n}.$$

This can be done reasonably quickly, even for very large numbers, using modular exponentiation. Bob then transmits c to Alice. Note that some values of m will yield a ciphertext c equal to m, but this is very unlikely to occur in practice.

§6.1 RSA Encryption

- **Decryption**: Alice can recover m from c by using her private key exponent d by computing

$$c^d \equiv (m^e)^d \equiv m \pmod{n}.$$

Given m, she can recover the original message M by reversing the padding scheme.

Example

Here is an toy example of RSA encryption and decryption.

1. Choose two prime numbers $p = 11$ and $q = 31$.
2. Compute $n = pq = 341$.
3. Compute $\varphi(n) = (p-1)(q-1) = 300$ / $(\lambda(n) = \mathrm{lcm}(10, 30) = 30)$.
4. Choose the encryption key $e = 17$ so that $1 < e < \varphi(n)$ and $\gcd(e, \varphi(n)) = 1$ / $(1 < e < \lambda(n)$ and $\gcd(e, \lambda(n)) = 1)$.

§6.1 RSA Encryption

- **Decryption**: Alice can recover m from c by using her private key exponent d by computing

$$c^d \equiv (m^e)^d \equiv m \pmod{n}.$$

Given m, she can recover the original message M by reversing the padding scheme.

Example

Here is an toy example of RSA encryption and decryption.

1. Choose two prime numbers $p = 11$ and $q = 31$.
2. Compute $n = pq = 341$.
3. Compute $\varphi(n) = (p-1)(q-1) = 300$ / $(\lambda(n) = \mathrm{lcm}(10, 30) = 30)$.
4. Choose the encryption key $e = 17$ so that $1 < e < \varphi(n)$ and $\gcd(e, \varphi(n)) = 1$ / $(1 < e < \lambda(n)$ and $\gcd(e, \lambda(n)) = 1)$.

§6.1 RSA Encryption

Example (cont'd)

5. Compute the decryption key d by Extended Euclid's algorithm:

j	r_j	q_j	s_j	t_j
-1	300		1	0
0	17	17	0	1
1	11	1	1	-17
2	6	1	-1	18
3	5	1	2	-35
4	1	5	-3	53

j	r_j	q_j	s_j	t_j
-1	30		1	0
0	17	1	0	1
1	13	1	1	-1
2	4	3	-1	2
3	1	4	4	-7

which implies that $300\times(-3)+17\times 53 = 1$ $(30\times 4+17\times(-7) = 1)$; thus $d = 53$ ($d \equiv -7 \pmod{30}$ or $d = 23$).

§6.1 RSA Encryption

Example (cont'd)

Therefore, to encrypt $m = 30$, we raise to the power of 17 and obtain the encrypted message:

$$30^{17} \equiv 123 \pmod{341}.$$

To decrypt the encrypted message, we raise it to the power of 53 (23) and obtain that

$$123^{53} \equiv (123^3)^{17} \cdot 123^2 \equiv 30^{17} \cdot 125 \equiv 123 \cdot 125 \equiv 30 \pmod{341}$$
$$(123^{23} \equiv (123^3)^7 \cdot 123^2 \equiv 30^7 \cdot 125 \equiv 123 \cdot 125 \equiv 30 \pmod{341}).$$

§6.2 Reduction from Factoring to Period-finding

The crucial observation of Shor was that there is an efficient quantum algorithm for the problem of period-finding and that factoring can be reduced to this, in the sense that an efficient algorithm for period-finding implies an efficient algorithm for factoring. We first explain the reduction. Suppose we want to find factors of the composite number $N > 1$. We may assume N is odd and not a prime power, since those cases can easily be filtered out by a classical algorithm. Now randomly choose some integer $x \in \{2, \cdots, N-1\}$ which is coprime to N. If x is not coprime to N, then the greatest common divisor of x and N is a nontrivial factor of N, so then we are already done. From now on consider x and N are coprime, so x is an element of the multiplicative group $\mathbb{Z}_N^*$.

§6.2 Reduction from Factoring to Period-finding

The crucial observation of Shor was that there is an efficient quantum algorithm for the problem of period-finding and that factoring can be reduced to this, in the sense that an efficient algorithm for period-finding implies an efficient algorithm for factoring. We first explain the reduction. Suppose we want to find factors of the composite number $N > 1$. We may assume N is odd and not a prime power, since those cases can easily be filtered out by a classical algorithm. Now randomly choose some integer $x \in \{2, \cdots, N-1\}$ which is coprime to N. If x is not coprime to N, then the greatest common divisor of x and N is a nontrivial factor of N, so then we are already done. From now on consider x and N are coprime, so x is an element of the multiplicative group $\mathbb{Z}_N^*$.

§6.2 Reduction from Factoring to Period-finding

The crucial observation of Shor was that there is an efficient quantum algorithm for the problem of period-finding and that factoring can be reduced to this, in the sense that an efficient algorithm for period-finding implies an efficient algorithm for factoring. We first explain the reduction. Suppose we want to find factors of the composite number $N > 1$. We may assume N is odd and not a prime power, since those cases can easily be filtered out by a classical algorithm. Now randomly choose some integer $x \in \{2, \cdots, N-1\}$ which is coprime to N. If x is not coprime to N, then the greatest common divisor of x and N is a nontrivial factor of N, so then we are already done. From now on consider x and N are coprime, so x is an element of the multiplicative group $\mathbb{Z}_N^*$.

§6.2 Reduction from Factoring to Period-finding

The crucial observation of Shor was that there is an efficient quantum algorithm for the problem of period-finding and that factoring can be reduced to this, in the sense that an efficient algorithm for period-finding implies an efficient algorithm for factoring. We first explain the reduction. Suppose we want to find factors of the composite number $N > 1$. We may assume N is odd and not a prime power, since those cases can easily be filtered out by a classical algorithm. Now randomly choose some integer $x \in \{2, \cdots, N-1\}$ which is coprime to N. If x is not coprime to N, then the greatest common divisor of x and N is a nontrivial factor of N, so then we are already done. From now on consider x and N are coprime, so x is an element of the multiplicative group $\mathbb{Z}_N^*$.

§6.2 Reduction from Factoring to Period-finding

Consider the sequence

$$1 = x^0 \bmod N, \quad x^1 \bmod N, \quad x^2 \bmod N, \cdots$$

This sequence will cycle after a while: there is a least $0 < r \leqslant N$ such that $x^r \equiv 1 \pmod{N}$. The **smallest** such number r is called the period of the sequence (a.k.a. the **order** of the element x in the group $(\mathbb{Z}_N^*, \odot)$). If r is even,

$$\begin{aligned} x^r \equiv 1 \pmod{N} &\Leftrightarrow (x^{r/2})^2 \equiv 1 \pmod{N} \\ &\Leftrightarrow (x^{r/2}+1)(x^{r/2}-1) \equiv 0 \pmod{N} \\ &\Leftrightarrow (x^{r/2}+1)(x^{r/2}-1) = kN \text{ for some } k \in \mathbb{N}. \end{aligned}$$

Because both $x^{r/2}+1 > 0$ and $x^{r/2}-1 > 0$ (due to the fact that $x > 1$), we must have $k \neq 0$. Hence $x^{r/2}+1$ or $x^{r/2}-1$ will share a factor with N.

§6.2 Reduction from Factoring to Period-finding

Consider the sequence

$$1 = x^0 \bmod N, \quad x^1 \bmod N, \quad x^2 \bmod N, \cdots$$

This sequence will cycle after a while: there is a least $0 < r \leqslant N$ such that $x^r \equiv 1 \pmod{N}$. The **smallest** such number r is called the period of the sequence (a.k.a. the **order** of the element x in the group $(\mathbb{Z}_N^*, \odot)$). If r is even,

$$\begin{aligned} x^r \equiv 1 \pmod{N} &\Leftrightarrow (x^{r/2})^2 \equiv 1 \pmod{N} \\ &\Leftrightarrow (x^{r/2}+1)(x^{r/2}-1) \equiv 0 \pmod{N} \\ &\Leftrightarrow (x^{r/2}+1)(x^{r/2}-1) = kN \text{ for some } k \in \mathbb{N}. \end{aligned}$$

Because both $x^{r/2}+1 > 0$ and $x^{r/2}-1 > 0$ (due to the fact that $x > 1$), we must have $k \neq 0$. Hence $x^{r/2}+1$ or $x^{r/2}-1$ will share a factor with N.

§6.2 Reduction from Factoring to Period-finding

Consider the sequence

$$1 = x^0 \bmod N, \quad x^1 \bmod N, \quad x^2 \bmod N, \cdots$$

This sequence will cycle after a while: there is a least $0 < r \leqslant N$ such that $x^r \equiv 1 \pmod{N}$. The **smallest** such number r is called the period of the sequence (a.k.a. the **order** of the element x in the group $(\mathbb{Z}_N^*, \odot)$). If r is even,

$$\begin{aligned} x^r \equiv 1 \pmod{N} &\Leftrightarrow (x^{r/2})^2 \equiv 1 \pmod{N} \\ &\Leftrightarrow (x^{r/2}+1)(x^{r/2}-1) \equiv 0 \pmod{N} \\ &\Leftrightarrow (x^{r/2}+1)(x^{r/2}-1) = kN \text{ for some } k \in \mathbb{N}. \end{aligned}$$

Because both $x^{r/2}+1 > 0$ and $x^{r/2}-1 > 0$ (due to the fact that $x > 1$), we must have $k \neq 0$. Hence $x^{r/2}+1$ or $x^{r/2}-1$ will share a factor with N.

§6.2 Reduction from Factoring to Period-finding

Note that $x^{r/2} \neq 1 \bmod N$ for otherwise $r/2$ is a period of f. In other words, $\gcd(x^{r/2} - 1, N) \neq N$. It is still possible that $\gcd(x^{r/2} - 1, N) = 1$ and this is equivalent to that $\gcd(x^{r/2} + 1, N) = N$. Therefore, we are able to factor N if $\gcd(x^{r/2} + 1, N) < N$.

Assuming that N is odd and not a prime power, it can be shown that with probability not less than $1/2$, the period r is even and $x^{r/2} + 1$ and $x^{r/2} - 1$ are not multiples of N.

Accordingly, with high probability we can obtain an even period r so that $\gcd(x^{r/2} + 1, N)$ is a non-trivial factor of N. If we are unlucky we might have chosen an x that does not give a factor (which we can detect efficiently), but trying a few different random x gives a high probability of finding a factor.

§6.2 Reduction from Factoring to Period-finding

Note that $x^{r/2} \neq 1 \mod N$ for otherwise $r/2$ is a period of f. In other words, $\gcd(x^{r/2} - 1, N) \neq N$. It is still possible that $\gcd(x^{r/2} - 1, N) = 1$ and this is equivalent to that $\gcd(x^{r/2} + 1, N) = N$. Therefore, we are able to factor N if $\gcd(x^{r/2} + 1, N) < N$.

Assuming that N is odd and not a prime power, it can be shown that with probability not less than $1/2$, the period r is even and $x^{r/2} + 1$ and $x^{r/2} - 1$ are not multiples of N.

Accordingly, with high probability we can obtain an even period r so that $\gcd(x^{r/2} + 1, N)$ is a non-trivial factor of N. If we are unlucky we might have chosen an x that does not give a factor (which we can detect efficiently), but trying a few different random x gives a high probability of finding a factor.

§6.2 Reduction from Factoring to Period-finding

Note that $x^{r/2} \neq 1 \mod N$ for otherwise $r/2$ is a period of f. In other words, $\gcd(x^{r/2} - 1, N) \neq N$. It is still possible that $\gcd(x^{r/2} - 1, N) = 1$ and this is equivalent to that $\gcd(x^{r/2} + 1, N) = N$. Therefore, we are able to factor N if $\gcd(x^{r/2} + 1, N) < N$.

Assuming that N is odd and not a prime power, it can be shown that with probability not less than $1/2$, the period r is even and $x^{r/2} + 1$ and $x^{r/2} - 1$ are not multiples of N.

Accordingly, with high probability we can obtain an even period r so that $\gcd(x^{r/2} + 1, N)$ is a non-trivial factor of N. If we are unlucky we might have chosen an x that does not give a factor (which we can detect efficiently), but trying a few different random x gives a high probability of finding a factor.

§6.2 Reduction from Factoring to Period-finding

Note that $x^{r/2} \neq 1 \bmod N$ for otherwise $r/2$ is a period of f. In other words, $\gcd(x^{r/2} - 1, N) \neq N$. It is still possible that $\gcd(x^{r/2} - 1, N) = 1$ and this is equivalent to that $\gcd(x^{r/2} + 1, N) = N$. Therefore, we are able to factor N if $\gcd(x^{r/2} + 1, N) < N$.

Assuming that N is odd and not a prime power, it can be shown that with probability not less than $1/2$, the period r is even and $x^{r/2} + 1$ and $x^{r/2} - 1$ are not multiples of N.

Accordingly, with high probability we can obtain an even period r so that $\gcd(x^{r/2} + 1, N)$ is a non-trivial factor of N. If we are unlucky we might have chosen an x that does not give a factor (which we can detect efficiently), but trying a few different random x gives a high probability of finding a factor.

§6.2 Reduction from Factoring to Period-finding

Factorization Algorithm: Let N be an odd natural number N that has at least two distinct prime factors.

Step 1: Choose $x \in \{2, \cdots, N-1\}$ and compute $\gcd(x, N)$.

1. If $\gcd(x, N) > 1$, then $\gcd(x, N)$ is a non-trivial factor of N and we are done.
2. If $\gcd(x, N) = 1$, then goto **Step 2**.

Step 2: Determine the period r of the function $f(a) = x^a \bmod N$.

1. If r is odd, goto **Step 1**.
2. If r is even, goto **Step 3**.

Step 3: Determine $\gcd(x^{r/2} + 1, N)$.

1. If $\gcd(x^{r/2} + 1, N) = N$, then goto **Step 1**.
2. If $\gcd(x^{r/2} + 1, N) < N$, then $\gcd(x^{r/2} + 1, N)$ is a non-trivial factor of N and we are done.

§6.2 Reduction from Factoring to Period-finding

Factorization Algorithm: Let N be an odd natural number N that has at least two distinct prime factors.

Step 1: Choose $x \in \{2, \cdots, N-1\}$ and compute $\gcd(x, N)$.

1. If $\gcd(x, N) > 1$, then $\gcd(x, N)$ is a non-trivial factor of N and we are done.
2. If $\gcd(x, N) = 1$, then goto **Step 2**.

Step 2: Determine the period r of the function $f(a) = x^a \bmod N$.

1. If r is odd, goto **Step 1**.
2. If r is even, goto **Step 3**.

Step 3: Determine $\gcd(x^{r/2} + 1, N)$.

1. If $\gcd(x^{r/2} + 1, N) = N$, then goto **Step 1**.
2. If $\gcd(x^{r/2} + 1, N) < N$, then $\gcd(x^{r/2} + 1, N)$ is a non-trivial factor of N and we are done.

§6.2 Reduction from Factoring to Period-finding

Factorization Algorithm: Let N be an odd natural number N that has at least two distinct prime factors.

Step 1: Choose $x \in \{2, \cdots, N-1\}$ and compute $\gcd(x, N)$.

1. If $\gcd(x, N) > 1$, then $\gcd(x, N)$ is a non-trivial factor of N and we are done.
2. If $\gcd(x, N) = 1$, then goto **Step 2**.

Step 2: Determine the period r of the function $f(a) = x^a \mod N$.

1. If r is odd, goto **Step 1**.
2. If r is even, goto **Step 3**.

Step 3: Determine $\gcd(x^{r/2} + 1, N)$.

1. If $\gcd(x^{r/2} + 1, N) = N$, then goto **Step 1**.
2. If $\gcd(x^{r/2} + 1, N) < N$, then $\gcd(x^{r/2} + 1, N)$ is a non-trivial factor of N and we are done.

§6.2 Reduction from Factoring to Period-finding

Thus the problem of factoring reduces to finding the period r of the function given by modular exponentiation $f(a) = x^a \bmod N$. In general, the period-finding problem can be stated as follows:

The period-finding problem: We are given some function $f : \mathbb{N} \to \{0, 1, \cdots, N-1\}$ with the property that there is some unknown $r \in \{0, 1, \cdots, N-1\}$ such that $f(a) = f(b)$ if and only if $a \equiv b \bmod r$. The goal is to find r.

§6.2 Reduction from Factoring to Period-finding

A naive algorithm is to compute $f(0)$, $f(1)$, $f(2)$, $\cdots$ until we encounter the value $f(0)$ for the second time. The input at which this happens is the period r that we are trying to find; however, r could be huge, polynomial in N. To be efficient, we would like a runtime that is polynomial in $\log_2 N$, since that is the bitsize of the inputs to f. It is generally believed that classical computers cannot solve period-finding problems efficiently. This problem can be solved efficiently on a quantum computer, using only $\mathcal{O}(\log\log N)$ evaluations of f (query) and $\mathcal{O}(\log\log N)$ quantum Fourier transforms. Even a somewhat more general kind of period-finding can be solved by **Shor's algorithm** with very few f-evaluations, whereas any classical bounded-error algorithm would need to evaluate the function $\Omega(N^{1/3}/\sqrt{\log N})$ times in order to find the period.

§6.2 Reduction from Factoring to Period-finding

A naive algorithm is to compute $f(0)$, $f(1)$, $f(2)$, $\cdots$ until we encounter the value $f(0)$ for the second time. The input at which this happens is the period r that we are trying to find; however, r could be huge, polynomial in N. To be efficient, we would like a runtime that is polynomial in $\log_2 N$, since that is the bitsize of the inputs to f. It is generally believed that classical computers cannot solve period-finding problems efficiently. This problem can be solved efficiently on a quantum computer, using only $\mathcal{O}(\log\log N)$ evaluations of f (query) and $\mathcal{O}(\log\log N)$ quantum Fourier transforms. Even a somewhat more general kind of period-finding can be solved by **Shor's algorithm** with very few f-evaluations, whereas any classical bounded-error algorithm would need to evaluate the function $\Omega(N^{1/3}/\sqrt{\log N})$ times in order to find the period.

§6.2 Reduction from Factoring to Period-finding

A naive algorithm is to compute $f(0)$, $f(1)$, $f(2)$, $\cdots$ until we encounter the value $f(0)$ for the second time. The input at which this happens is the period r that we are trying to find; however, r could be huge, polynomial in N. To be efficient, we would like a runtime that is polynomial in $\log_2 N$, since that is the bitsize of the inputs to f. It is generally believed that classical computers cannot solve period-finding problems efficiently. This problem can be solved efficiently on a quantum computer, using only $\mathcal{O}(\log\log N)$ evaluations of f (query) and $\mathcal{O}(\log\log N)$ quantum Fourier transforms. Even a somewhat more general kind of period-finding can be solved by **Shor's algorithm** with very few f-evaluations, whereas any classical bounded-error algorithm would need to evaluate the function $\Omega(N^{1/3}/\sqrt{\log N})$ times in order to find the period.

§6.2 Reduction from Factoring to Period-finding

A naive algorithm is to compute $f(0)$, $f(1)$, $f(2)$, $\cdots$ until we encounter the value $f(0)$ for the second time. The input at which this happens is the period r that we are trying to find; however, r could be huge, polynomial in N. To be efficient, we would like a runtime that is polynomial in $\log_2 N$, since that is the bitsize of the inputs to f. It is generally believed that classical computers cannot solve period-finding problems efficiently. This problem can be solved efficiently on a quantum computer, using only $\mathcal{O}(\log\log N)$ evaluations of f (query) and $\mathcal{O}(\log\log N)$ quantum Fourier transforms. Even a somewhat more general kind of period-finding can be solved by **Shor's algorithm** with very few f-evaluations, whereas any classical bounded-error algorithm would need to evaluate the function $\Omega(N^{1/3}/\sqrt{\log N})$ times in order to find the period.

§6.2 Reduction from Factoring to Period-finding

A naive algorithm is to compute $f(0)$, $f(1)$, $f(2)$, $\cdots$ until we encounter the value $f(0)$ for the second time. The input at which this happens is the period r that we are trying to find; however, r could be huge, polynomial in N. To be efficient, we would like a runtime that is polynomial in $\log_2 N$, since that is the bitsize of the inputs to f. It is generally believed that classical computers cannot solve period-finding problems efficiently. This problem can be solved efficiently on a quantum computer, using only $\mathcal{O}(\log\log N)$ evaluations of f (query) and $\mathcal{O}(\log\log N)$ quantum Fourier transforms. Even a somewhat more general kind of period-finding can be solved by **Shor's algorithm** with very few f-evaluations, whereas any classical bounded-error algorithm would need to evaluate the function $\Omega(N^{1/3}/\sqrt{\log N})$ times in order to find the period.

§6.2 Reduction from Factoring to Period-finding

A naive algorithm is to compute $f(0)$, $f(1)$, $f(2)$, $\cdots$ until we encounter the value $f(0)$ for the second time. The input at which this happens is the period r that we are trying to find; however, r could be huge, polynomial in N. To be efficient, we would like a runtime that is polynomial in $\log_2 N$, since that is the bitsize of the inputs to f. It is generally believed that classical computers cannot solve period-finding problems efficiently. This problem can be solved efficiently on a quantum computer, using only $\mathcal{O}(\log\log N)$ evaluations of f (query) and $\mathcal{O}(\log\log N)$ quantum Fourier transforms. Even a somewhat more general kind of period-finding can be solved by **Shor's algorithm** with very few f-evaluations, whereas any classical bounded-error algorithm would need to evaluate the function $\Omega(N^{1/3}/\sqrt{\log N})$ times in order to find the period.

§6.3 Shor's Period-Finding Algorithm

Before proceeding to the discussion of Shor's algorithm, let us point out that the period-finding problem in §6.2 can be related to the **phase estimation** problem in the following sense: given $x \in \mathbb{Z}_N^*$, the (unitary) map

$$U|y\rangle = |x \odot y\rangle \equiv |x \cdot y \bmod N\rangle$$

has eigenvectors

$$|\psi_s\rangle \equiv \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\Big(-\frac{2\pi isk}{r}\Big) |x^k \bmod N\rangle$$

with $0 \leqslant s < r$ since

$$U|\psi_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\Big(-\frac{2\pi isk}{r}\Big) U|x^k \bmod N\rangle$$

Therefore, the phase estimation algorithm introduced in Section 5.5 can be applied to find r as long as the eigenvector $|\psi_s\rangle$ is known (even though we do not know $|\psi_s\rangle$ for $s \neq 0$).

§6.3 Shor's Period-Finding Algorithm

Before proceeding to the discussion of Shor's algorithm, let us point out that the period-finding problem in §6.2 can be related to the **phase estimation** problem in the following sense: given $x \in \mathbb{Z}_N^*$, the (unitary) map

$$U|y\rangle = |x \odot y\rangle \equiv |x \cdot y \bmod N\rangle$$

has eigenvectors

$$|\psi_s\rangle \equiv \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\Big(-\frac{2\pi i s k}{r}\Big) |x^k \bmod N\rangle$$

with $0 \leqslant s < r$ since

$$U|\psi_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\Big(-\frac{2\pi i s k}{r}\Big) |x^{k+1} \bmod N\rangle$$

Therefore, the phase estimation algorithm introduced in Section 5.5 can be applied to find r as long as the eigenvector $|\psi_s\rangle$ is known (even though we do not know $|\psi_s\rangle$ for $s \neq 0$).

§6.3 Shor's Period-Finding Algorithm

Before proceeding to the discussion of Shor's algorithm, let us point out that the period-finding problem in §6.2 can be related to the **phase estimation** problem in the following sense: given $x \in \mathbb{Z}_N^*$, the (unitary) map

$$U|y\rangle = |x \odot y\rangle \equiv |x \cdot y \bmod N\rangle$$

has eigenvectors

$$|\psi_s\rangle \equiv \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\Big(-\frac{2\pi i s k}{r}\Big) |x^k \bmod N\rangle$$

with $0 \leqslant s < r$ since

$$U|\psi_s\rangle = \exp\Big(\frac{2\pi i s}{r}\Big) \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\Big(-\frac{2\pi i s(k+1)}{r}\Big) |x^{k+1} \bmod N\rangle$$

Therefore, the phase estimation algorithm introduced in Section 5.5 can be applied to find r as long as the eigenvector $|\psi_s\rangle$ is known (even though we do not know $|\psi_s\rangle$ for $s \neq 0$).

§6.3 Shor's Period-Finding Algorithm

Before proceeding to the discussion of Shor's algorithm, let us point out that the period-finding problem in §6.2 can be related to the **phase estimation** problem in the following sense: given $x \in \mathbb{Z}_N^*$, the (unitary) map

$$U|y\rangle = |x \odot y\rangle \equiv |x \cdot y \bmod N\rangle$$

has eigenvectors

$$|\psi_s\rangle \equiv \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\Big(-\frac{2\pi i s k}{r}\Big) |x^k \bmod N\rangle$$

with $0 \leqslant s < r$ since

$$U|\psi_s\rangle = \exp\Big(\frac{2\pi i s}{r}\Big) \frac{1}{\sqrt{r}} \sum_{\ell=1}^{r} \exp\Big(-\frac{2\pi i s \ell}{r}\Big) |x^\ell \quad \bmod N\rangle$$

Therefore, the phase estimation algorithm introduced in Section 5.5 can be applied to find r as long as the eigenvector $|\psi_s\rangle$ is known (even though we do not know $|\psi_s\rangle$ for $s \neq 0$).

§6.3 Shor's Period-Finding Algorithm

Before proceeding to the discussion of Shor's algorithm, let us point out that the period-finding problem in §6.2 can be related to the **phase estimation** problem in the following sense: given $x \in \mathbb{Z}_N^*$, the (unitary) map

$$U|y\rangle = |x \odot y\rangle \equiv |x \cdot y \bmod N\rangle$$

has eigenvectors

$$|\psi_s\rangle \equiv \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\Big(-\frac{2\pi isk}{r}\Big) |x^k \bmod N\rangle$$

with $0 \leqslant s < r$ since

$$U|\psi_s\rangle = \exp\Big(\frac{2\pi is}{r}\Big) \frac{1}{\sqrt{r}} \sum_{\ell=0}^{r-1} \exp\Big(-\frac{2\pi is\ell}{r}\Big) |x^\ell \quad \bmod N\rangle$$

Therefore, the phase estimation algorithm introduced in Section 5.5 can be applied to find r as long as the eigenvector $|\psi_s\rangle$ is known (even though we do not know $|\psi_s\rangle$ for $s \neq 0$).

§6.3 Shor's Period-Finding Algorithm

Before proceeding to the discussion of Shor's algorithm, let us point out that the period-finding problem in §6.2 can be related to the **phase estimation** problem in the following sense: given $x \in \mathbb{Z}_N^*$, the (unitary) map

$$U|y\rangle = |x \odot y\rangle \equiv |x \cdot y \bmod N\rangle$$

has eigenvectors

$$|\psi_s\rangle \equiv \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\Big(-\frac{2\pi i s k}{r}\Big) |x^k \bmod N\rangle$$

with $0 \leqslant s < r$ since

$$U|\psi_s\rangle = \exp\Big(\frac{2\pi i s}{r}\Big) \frac{1}{\sqrt{r}} \sum_{\ell=0}^{r-1} \exp\Big(-\frac{2\pi i s \ell}{r}\Big) |x^\ell \bmod N\rangle$$

Therefore, the phase estimation algorithm introduced in Section 5.5 can be applied to find r as long as the eigenvector $|\psi_s\rangle$ is known (even though we do not know $|\psi_s\rangle$ for $s \neq 0$).

§6.3 Shor's Period-Finding Algorithm

Before proceeding to the discussion of Shor's algorithm, let us point out that the period-finding problem in §6.2 can be related to the **phase estimation** problem in the following sense: given $x \in \mathbb{Z}_N^*$, the (unitary) map

$$U|y\rangle = |x \odot y\rangle \equiv |x \cdot y \bmod N\rangle$$

has eigenvectors

$$|\psi_s\rangle \equiv \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\Big(-\frac{2\pi i s k}{r}\Big) |x^k \bmod N\rangle$$

with $0 \leqslant s < r$ since

$$U|\psi_s\rangle = \exp\Big(\frac{2\pi i s}{r}\Big) \frac{1}{\sqrt{r}} \sum_{\ell=0}^{r-1} \exp\Big(-\frac{2\pi i s \ell}{r}\Big) |x^\ell \bmod N\rangle$$

Therefore, the phase estimation algorithm introduced in Section 5.5 can be applied to find r as long as the eigenvector $|\psi_s\rangle$ is known (even though we do not know $|\psi_s\rangle$ for $s \neq 0$).

§6.3 Shor's Period-Finding Algorithm

Before proceeding to the discussion of Shor's algorithm, let us point out that the period-finding problem in §6.2 can be related to the **phase estimation** problem in the following sense: given $x \in \mathbb{Z}_N^*$, the (unitary) map

$$U|y\rangle = |x \odot y\rangle \equiv |x \cdot y \bmod N\rangle$$

has eigenvectors

$$|\psi_s\rangle \equiv \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\Big(-\frac{2\pi i s k}{r}\Big) |x^k \bmod N\rangle$$

with $0 \leqslant s < r$ since

$$U|\psi_s\rangle = \exp\Big(\frac{2\pi i s}{r}\Big)|\psi_s\rangle\,.$$

Therefore, the phase estimation algorithm introduced in Section 5.5 can be applied to find r as long as the eigenvector $|\psi_s\rangle$ is known (even though we do not know $|\psi_s\rangle$ for $s \neq 0$).

§6.3 Shor's Period-Finding Algorithm

Before proceeding to the discussion of Shor's algorithm, let us point out that the period-finding problem in §6.2 can be related to the **phase estimation** problem in the following sense: given $x \in \mathbb{Z}_N^*$, the (unitary) map

$$U|y\rangle = |x \odot y\rangle \equiv |x \cdot y \bmod N\rangle$$

has eigenvectors

$$|\psi_s\rangle \equiv \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\left(-\frac{2\pi i s k}{r}\right) |x^k \bmod N\rangle$$

with $0 \leqslant s < r$ since

$$U|\psi_s\rangle = \exp\left(\frac{2\pi i s}{r}\right)|\psi_s\rangle .$$

Therefore, the phase estimation algorithm introduced in Section 5.5 can be applied to find r as long as the eigenvector $|\psi_s\rangle$ is known (even though we do not know $|\psi_s\rangle$ for $s \neq 0$).

§6.3 Shor's Period-Finding Algorithm

Now we will show how Shor's algorithm finds the period r of the function f, given a "black-box" that maps $|a\rangle|0^K\rangle \mapsto |a\rangle|f(a)\rangle$. We can always efficiently pick some $q = 2^L$ such that $N^2 < q \leqslant 2N^2$. Then we can implement the Fourier transform QFT using $\mathcal{O}((\log_2 N)^2)$ gates. Let O_f denote the unitary that maps $|a\rangle|0^K\rangle \mapsto |a\rangle|f(a)\rangle$, where the first register consists of L qubits, and the second of $K = [\log_2 N] + 1$ qubits.

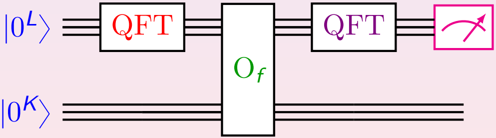

Figure 1: Shor's period-finding algorithm

§6.3 Shor's Period-Finding Algorithm

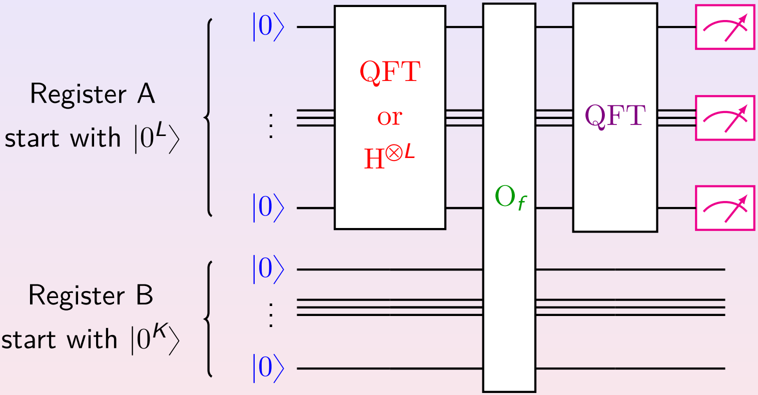

Figure 2: Shor's period-finding algorithm

§6.3 Shor's Period-Finding Algorithm

Shor's period-finding algorithm is illustrated in previous figures. Start with $|\psi_0\rangle = |0^L\rangle|0^K\rangle$. Apply the QFT (or just L Hadamard gates) to the first register to build the uniform superposition

$$|\psi_1\rangle = (\mathrm{H}^{\otimes L} \otimes \mathrm{I}_K)|\psi_0\rangle = \frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a\rangle|0^K\rangle,$$

where I_K denotes the identity map on the second register. The second register still consists of zeroes. Now use the "black-box" to compute $f(a)$ in quantum parallel:

$$|\psi_2\rangle = \mathrm{O}_f|\psi_1\rangle = \frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a\rangle|f(a)\rangle.$$

Next we apply the quantum Fourier transform **QFT** to the first register to obtain the quantum state $|\psi_3\rangle = (F_q \otimes \mathrm{I}_K)|\psi_2\rangle$. Finally, we measure the first register and obtain a number b and wish to find the period of f based on this observation.

§6.3 Shor's Period-Finding Algorithm

Shor's period-finding algorithm is illustrated in previous figures. Start with $|\psi_0\rangle = |0^L\rangle|0^K\rangle$. Apply the QFT (or just L Hadamard gates) to the first register to build the uniform superposition

$$|\psi_1\rangle = (\mathrm{H}^{\otimes L} \otimes \mathrm{I}_K)|\psi_0\rangle = \frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a\rangle|0^K\rangle,$$

where I_K denotes the identity map on the second register. The second register still consists of zeroes. Now use the "black-box" to compute $f(a)$ in quantum parallel:

$$|\psi_2\rangle = \mathrm{O}_f|\psi_1\rangle = \frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a\rangle|f(a)\rangle.$$

Next we apply the quantum Fourier transform **QFT** to the first register to obtain the quantum state $|\psi_3\rangle = (F_q \otimes \mathrm{I}_K)|\psi_2\rangle$. Finally, we measure the first register and obtain a number b and wish to find the period of f based on this observation.

§6.3 Shor's Period-Finding Algorithm

Shor's period-finding algorithm is illustrated in previous figures. Start with $|\psi_0\rangle = |0^L\rangle|0^K\rangle$. Apply the QFT (or just L Hadamard gates) to the first register to build the uniform superposition

$$|\psi_1\rangle = (\mathrm{H}^{\otimes L} \otimes \mathrm{I}_K)|\psi_0\rangle = \frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a\rangle|0^K\rangle,$$

where I_K denotes the identity map on the second register. The second register still consists of zeroes. Now use the "black-box" to compute $f(a)$ in quantum parallel:

$$|\psi_2\rangle = \mathrm{O}_f|\psi_1\rangle = \frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a\rangle|f(a)\rangle.$$

Next we apply the quantum Fourier transform **QFT** to the first register to obtain the quantum state $|\psi_3\rangle = (F_q \otimes \mathrm{I}_K)|\psi_2\rangle$. Finally, we measure the first register and obtain a number b and wish to find the period of f based on this observation.

§6.3 Shor's Period-Finding Algorithm

Shor's period-finding algorithm is illustrated in previous figures. Start with $|\psi_0\rangle = |0^L\rangle|0^K\rangle$. Apply the QFT (or just L Hadamard gates) to the first register to build the uniform superposition

$$|\psi_1\rangle = (\mathrm{H}^{\otimes L} \otimes \mathrm{I}_K)|\psi_0\rangle = \frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a\rangle|0^K\rangle,$$

where I_K denotes the identity map on the second register. The second register still consists of zeroes. Now use the "black-box" to compute $f(a)$ in quantum parallel:

$$|\psi_2\rangle = \mathrm{O}_f|\psi_1\rangle = \frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a\rangle|f(a)\rangle.$$

Next we apply the quantum Fourier transform **QFT** to the first register to obtain the quantum state $|\psi_3\rangle = (F_q \otimes \mathrm{I}_K)|\psi_2\rangle$. Finally, we measure the first register and obtain a number b and wish to find the period of f based on this observation.

§6.3 Shor's Period-Finding Algorithm

Some of the measurement b obtained by Shor's algorithm above are useless. The measurement b becomes useful for us to determine the period r if b belongs to the set

$$E = \left\{ b \in \mathbb{Z} \,\middle|\, 0 \leqslant b \leqslant q-1 \text{ and } \left|\frac{b}{q} - \frac{c}{r}\right| < \frac{1}{2r^2} \text{ for some } c \in \mathbb{Z}_r^* \right\},$$

where we recall that $\mathbb{Z}_r^*$ is the collection of numbers from $\{1, \cdots, r-1\}$ that are coprime to r so that $\#\mathbb{Z}_r^* = \varphi(r)$. We note that E is indeed unknown to us (since r is unknown to us) but it exists and is a non-empty set. We will show in Section 6.5 that the probability of obtaining $b \in E$ by Shor's algorithm is not less than $\frac{1}{10 \ln L}$. This implies that if we apply Shor's algorithm k times, the probability of obtaining **no** $b \in E$ is less than $\left(1 - \frac{1}{10 \ln L}\right)^k$ which is quite small when k is large.

§6.3 Shor's Period-Finding Algorithm

Some of the measurement b obtained by Shor's algorithm above are useless. The measurement b becomes useful for us to determine the period r if b belongs to the set

$$E = \left\{ b \in \mathbb{Z} \,\middle|\, 0 \leqslant b \leqslant q-1 \text{ and } \left|\frac{b}{q} - \frac{c}{r}\right| < \frac{1}{2r^2} \text{ for some } c \in \mathbb{Z}_r^* \right\},$$

where we recall that $\mathbb{Z}_r^*$ is the collection of numbers from $\{1, \cdots, r-1\}$ that are coprime to r so that $\#\mathbb{Z}_r^* = \varphi(r)$. We note that E is indeed unknown to us (since r is unknown to us) but it exists and is a non-empty set. We will show in Section 6.5 that the probability of obtaining $b \in E$ by Shor's algorithm is not less than $\dfrac{1}{10 \ln L}$. This implies that if we apply Shor's algorithm k times, the probability of obtaining **no** $b \in E$ is less than $\left(1 - \dfrac{1}{10 \ln L}\right)^k$ which is quite small when k is large.

§6.3 Shor's Period-Finding Algorithm

Some of the measurement b obtained by Shor's algorithm above are useless. The measurement b becomes useful for us to determine the period r if b belongs to the set

$$E = \left\{ b \in \mathbb{Z} \,\middle|\, 0 \leqslant b \leqslant q-1 \text{ and } \left|\frac{b}{q} - \frac{c}{r}\right| < \frac{1}{2r^2} \text{ for some } c \in \mathbb{Z}_r^* \right\},$$

where we recall that $\mathbb{Z}_r^*$ is the collection of numbers from $\{1, \cdots, r-1\}$ that are coprime to r so that $\#\mathbb{Z}_r^* = \varphi(r)$. We note that E is indeed unknown to us (since r is unknown to us) but it exists and is a non-empty set. We will show in Section 6.5 that the probability of obtaining $b \in E$ by Shor's algorithm is not less than $\dfrac{1}{10 \ln L}$. This implies that if we apply Shor's algorithm k times, the probability of obtaining **no** $b \in E$ is less than $\left(1 - \dfrac{1}{10 \ln L}\right)^k$ which is quite small when k is large.

§6.3 Shor's Period-Finding Algorithm

Suppose that we apply Shor's algorithm and obtain one such $b \in E$. Then there exists $c \in \mathbb{Z}_r^*$ such that $\left|\frac{b}{q} - \frac{c}{r}\right| < \frac{1}{2r^2}$. We note that in this inequality we only know b and q (so is the number $x = b/q$), and both c and r are unknown to us. Even though c and r are unknown to us, the fact that $c \in \mathbb{Z}_r^*$ implies that $\frac{c}{r}$ is an irreducible fraction（最簡分數）. Therefore, if there is a fast algorithm to find all irreducible fractions $\frac{n}{m}$ satisfying

$$\left|x - \frac{n}{m}\right| < \frac{1}{2m^2} \quad \text{and} \quad m < N, \tag{1}$$

we can check whether the denominators m of all such irreducible fractions is the period of f. In Section 6.4 an efficient algorithm is proposed to find all irreducible fractions $\frac{n}{m}$ satisfying (1).

§6.3 Shor's Period-Finding Algorithm

Suppose that we apply Shor's algorithm and obtain one such $b \in E$. Then there exists $c \in \mathbb{Z}_r^*$ such that $\left|\frac{b}{q} - \frac{c}{r}\right| < \frac{1}{2r^2}$. We note that in this inequality we only know b and q (so is the number $x = b/q$), and both c and r are unknown to us. Even though c and r are unknown to us, the fact that $c \in \mathbb{Z}_r^*$ implies that $\frac{c}{r}$ is an irreducible fraction（最簡分數）. Therefore, if there is a fast algorithm to find all irreducible fractions $\frac{n}{m}$ satisfying

$$\left|x - \frac{n}{m}\right| < \frac{1}{2m^2} \quad \text{and} \quad m < N, \tag{1}$$

we can check whether the denominators m of all such irreducible fractions is the period of f. In Section 6.4 an efficient algorithm is proposed to find all irreducible fractions $\frac{n}{m}$ satisfying (1).

§6.3 Shor's Period-Finding Algorithm

Suppose that we apply Shor's algorithm and obtain one such $b \in E$. Then there exists $c \in \mathbb{Z}_r^*$ such that $\left|\frac{b}{q} - \frac{c}{r}\right| < \frac{1}{2r^2}$. We note that in this inequality we only know b and q (so is the number $x = b/q$), and both c and r are unknown to us. Even though c and r are unknown to us, the fact that $c \in \mathbb{Z}_r^*$ implies that $\frac{c}{r}$ is an irreducible fraction（最簡分數）. Therefore, if there is a fast algorithm to find all irreducible fractions $\frac{n}{m}$ satisfying

$$\left|x - \frac{n}{m}\right| < \frac{1}{2m^2} \quad \text{and} \quad m < N, \tag{1}$$

we can check whether the denominators m of all such irreducible fractions is the period of f. In Section 6.4 an efficient algorithm is proposed to find all irreducible fractions $\frac{n}{m}$ satisfying (1).

§6.3 Shor's Period-Finding Algorithm

Suppose that we apply Shor's algorithm and obtain one such $b \in E$. Then there exists $c \in \mathbb{Z}_r^*$ such that $\left|\frac{b}{q} - \frac{c}{r}\right| < \frac{1}{2r^2}$. We note that in this inequality we only know b and q (so is the number $x = b/q$), and both c and r are unknown to us. Even though c and r are unknown to us, the fact that $c \in \mathbb{Z}_r^*$ implies that $\frac{c}{r}$ is an irreducible fraction（最簡分數）. Therefore, if there is a fast algorithm to find all irreducible fractions $\frac{n}{m}$ satisfying

$$\left|x - \frac{n}{m}\right| < \frac{1}{2m^2} \quad \text{and} \quad m < N, \tag{1}$$

we can check whether the denominators m of all such irreducible fractions is the period of f. In Section 6.4 an efficient algorithm is proposed to find all irreducible fractions $\frac{n}{m}$ satisfying (1).

§6.3 Shor's Period-Finding Algorithm

Shor's period-finding algorithm: Let $f : \mathbb{N} \cup \{0\} \to \{0,1\}^K$ be a periodic function with period r satisfying $19 \leqslant r < 2^{L/2}$ for some $L \in \mathbb{N}$ such that f is injective within one period.

Step 1: Measure the first register of the quantum state

$$(F_{2^L} \otimes \mathrm{I}_K) U_f (\mathrm{H}^{\otimes L} \otimes \mathrm{I}_K)(|0^L\rangle \otimes |0^K\rangle) .$$

and obtain b.

Step 2: Find all irreducible fractions $\frac{n}{m}$ satisfying

$$\left|\frac{b}{2^L} - \frac{n}{m}\right| < \frac{1}{2m^2} \qquad \text{and} \qquad m < 2^{L/2} .$$

1. If one of such denominator m is the period of f, we are done.
2. If none of these denominators m is the period of f, then $b \notin E$ and goto **Step 1**.

§6.3 Shor's Period-Finding Algorithm

Shor's period-finding algorithm: Let $f : \mathbb{N} \cup \{0\} \to \{0,1\}^K$ be a periodic function with period r satisfying $19 \leqslant r < 2^{L/2}$ for some $L \in \mathbb{N}$ such that f is injective within one period.

Step 1: Measure the first register of the quantum state

$$(F_{2^L} \otimes \mathrm{I}_K) U_f (\mathrm{H}^{\otimes L} \otimes \mathrm{I}_K)(|0^L\rangle \otimes |0^K\rangle) .$$

and obtain b.

Step 2: Find all irreducible fractions $\frac{n}{m}$ satisfying

$$\left|\frac{b}{2^L} - \frac{n}{m}\right| < \frac{1}{2m^2} \qquad \text{and} \qquad m < 2^{L/2} .$$

1. If one of such denominator m is the period of f, we are done.
2. If none of these denominators m is the period of f, then $b \notin E$ and goto **Step 1**.

§6.3 Shor's Period-Finding Algorithm

Shor's period-finding algorithm: Let $f : \mathbb{N} \cup \{0\} \to \{0,1\}^K$ be a periodic function with period r satisfying $19 \leqslant r < 2^{L/2}$ for some $L \in \mathbb{N}$ such that f is injective within one period.

Step 1: Measure the first register of the quantum state

$$(F_{2^L} \otimes \mathrm{I}_K) U_f (\mathrm{H}^{\otimes L} \otimes \mathrm{I}_K)(|0^L\rangle \otimes |0^K\rangle) .$$

and obtain b.

Step 2: Find all irreducible fractions $\dfrac{n}{m}$ satisfying

$$\Big|\frac{b}{2^L} - \frac{n}{m}\Big| < \frac{1}{2m^2} \qquad \text{and} \qquad m < 2^{L/2} .$$

1. If one of such denominator m is the period of f, we are done.
2. If none of these denominators m is the period of f, then $b \notin E$ and goto **Step 1**.

§6.4 Continued fractions

A continued fraction (連分數), or simply CF, is a real number of the form

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\cdots}}}.$$

The continued fraction above is denote by $[a_0; a_1, a_2, \cdots]$ (here the number of a_i's can be finite or infinite), and the a_i's are called the partial quotients. We assume these to be positive natural numbers. $[a_0; \cdots, a_n]$ is called the n-th convergent of the continued fraction $[a_0; a_1, a_2, \cdots]$, and can be simply computed by the following iterative scheme: $[a_0; \cdots, a_n]$, in its lowest terms, is p_n/q_n, where

$$\begin{aligned} p_0 &= a_0, & p_1 &= a_1 a_0 + 1, & p_n &= a_n p_{n-1} + p_{n-2}, \\ q_0 &= 1, & q_1 &= a_1, & q_n &= a_n q_{n-1} + q_{n-2}. \end{aligned} \tag{2}$$

§6.4 Continued fractions

A continued fraction (連分數), or simply CF, is a real number of the form

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\cdots}}}\,.$$

The continued fraction above is denote by $[a_0; a_1, a_2, \cdots]$ (here the number of a_i's can be finite or infinite), and the a_i's are called the partial quotients. We assume these to be positive natural numbers. $[a_0; \cdots, a_n]$ is called the n-th convergent of the continued fraction $[a_0; a_1, a_2, \cdots]$, and can be simply computed by the following iterative scheme: $[a_0; \cdots, a_n]$, in its lowest terms, is p_n/q_n, where

$$\begin{aligned} p_0 &= a_0, & p_1 &= a_1 a_0 + 1, & p_n &= a_n p_{n-1} + p_{n-2}\,, \\ q_0 &= 1, & q_1 &= a_1, & q_n &= a_n q_{n-1} + q_{n-2}\,. \end{aligned} \tag{2}$$

§6.4 Continued fractions

Note that q_n increases at least exponentially with n since $q_n \geqslant 2q_{n-2}$. Given a real number x, the following "algorithm" gives a continued fraction expansion of x:

$$\begin{aligned} a_0 &\equiv [x], & x_1 &\equiv 1/(x-a_0), \\ a_1 &\equiv [x_1], & x_2 &\equiv 1/(x_1-a_1), \\ a_2 &\equiv [x_2], & x_3 &\equiv 1/(x_2-a_2), \\ &\vdots \end{aligned}$$

Informally, we just take the integer part of the number as the partial quotient and continue with the inverse of the decimal part of the number.

Theorem

For an $x \in \mathbb{R}$, the sequence $\{a_j\}$ constructed from the algorithm above terminates if and only if x is rational.

§6.4 Continued fractions

Note that q_n increases at least exponentially with n since $q_n \geqslant 2q_{n-2}$. Given a real number x, the following "algorithm" gives a continued fraction expansion of x:

$$\begin{aligned} a_0 &\equiv [x], & x_1 &\equiv 1/(x - a_0), \\ a_1 &\equiv [x_1], & x_2 &\equiv 1/(x_1 - a_1), \\ a_2 &\equiv [x_2], & x_3 &\equiv 1/(x_2 - a_2), \\ & & \vdots & \end{aligned}$$

Informally, we just take the integer part of the number as the partial quotient and continue with the inverse of the decimal part of the number.

Theorem

For an $x \in \mathbb{R}$, the sequence $\{a_j\}$ constructed from the algorithm above terminates if and only if x is rational.

§6.4 Continued fractions

Note that q_n increases at least exponentially with n since $q_n \geqslant 2q_{n-2}$. Given a real number x, the following "algorithm" gives a continued fraction expansion of x:

$$\begin{aligned} a_0 &\equiv [x], & x_1 &\equiv 1/(x - a_0), \\ a_1 &\equiv [x_1], & x_2 &\equiv 1/(x_1 - a_1), \\ a_2 &\equiv [x_2], & x_3 &\equiv 1/(x_2 - a_2), \\ & & \vdots & \end{aligned}$$

Informally, we just take the integer part of the number as the partial quotient and continue with the inverse of the decimal part of the number.

Theorem

For an $x \in \mathbb{R}$, the sequence $\{a_j\}$ constructed from the algorithm above terminates if and only if x is rational.

§6.4 Continued fractions

Example

Let $x = \sqrt{2}$. Then $a_0 = 1$ and $a_k = 2$ for all $k \in \mathbb{N}$. To see this, we note that $x_1 = \frac{1}{\sqrt{2}-1} = \sqrt{2}+1$ so we have $a_1 = 2$. This then shows that

$$x_2 = \frac{1}{x_1 - a_1} = \frac{1}{\sqrt{2}+1-2} = \sqrt{2}+1$$

and as a consequence $a_2 = 2$. Repeating this process, we find that $x_k = \sqrt{2}+1$ and $a_k = 2$ for all $k \in \mathbb{N}$. Using (2), we obtain that

n	1	2	3	4	5	6
p_n	3	7	17	41	99	239
q_n	2	5	12	29	70	169
$\left\|x - \frac{p_n}{q_n}\right\|$	0.0858	0.0142	0.0025	4.2e-4	7.2e-5	1.2e-5

§6.4 Continued fractions

Example

Let $x = \sqrt{2}$. Then $a_0 = 1$ and $a_k = 2$ for all $k \in \mathbb{N}$. To see this, we note that $x_1 = \dfrac{1}{\sqrt{2}-1} = \sqrt{2}+1$ so we have $a_1 = 2$. This then shows that

$$x_2 = \frac{1}{x_1 - a_1} = \frac{1}{\sqrt{2}+1-2} = \sqrt{2}+1$$

and as a consequence $a_2 = 2$. Repeating this process, we find that $x_k = \sqrt{2}+1$ and $a_k = 2$ for all $k \in \mathbb{N}$. Using (2), we obtain that

n	1	2	3	4	5	6
p_n	3	7	17	41	99	239
q_n	2	5	12	29	70	169
$\left\lvert x - \frac{p_n}{q_n} \right\rvert$	0.0858	0.0142	0.0025	4.2e-4	7.2e-5	1.2e-5

§6.4 Continued fractions

Example

Let $x = \sqrt{2}$. Then $a_0 = 1$ and $a_k = 2$ for all $k \in \mathbb{N}$. To see this, we note that $x_1 = \dfrac{1}{\sqrt{2}-1} = \sqrt{2}+1$ so we have $a_1 = 2$. This then shows that

$$x_2 = \frac{1}{x_1 - a_1} = \frac{1}{\sqrt{2}+1-2} = \sqrt{2}+1$$

and as a consequence $a_2 = 2$. Repeating this process, we find that $x_k = \sqrt{2}+1$ and $a_k = 2$ for all $k \in \mathbb{N}$. Using (2), we obtain that

n	7	8	9	10	11	12
p_n	577	1393	3363	8119	19601	47321
q_n	408	985	2378	5741	13860	33461
$\left\lvert x - \frac{p_n}{q_n} \right\rvert$	2.1e-6	3.6e-7	6.3e-8	1.1e-8	1.8e-9	3.2e-10

§6.4 Continued fractions

The convergence of the CF approximate x follows from the fact that

$$\text{if } x = [a_0; a_1, \cdots], \text{ then } \left| x - \frac{p_n}{q_n} \right| \leqslant \frac{1}{q_n^2}.$$

Recall that q_n increases exponentially with n, so this convergence is quite fast. Moreover, p_n/q_n provides the **best approximation** of x among all fractions with denominator not greater than q_n:

$$\text{if } n \geqslant 1,\ q \leqslant q_n,\ \frac{p}{q} \neq \frac{p_n}{q_n}, \text{ then } \left| x - \frac{p_n}{q_n} \right| < \left| x - \frac{p}{q} \right|.$$

§6.4 Continued fractions

The convergence of the CF approximate x follows from the fact that

$$\text{if } x = [a_0; a_1, \cdots], \text{ then } \left| x - \frac{p_n}{q_n} \right| \leqslant \frac{1}{q_n^2}.$$

Recall that q_n increases exponentially with n, so this convergence is quite fast. Moreover, p_n/q_n provides the **best approximation** of x among all fractions with denominator not greater than q_n:

$$\text{if } n \geqslant 1,\ q \leqslant q_n,\ \frac{p}{q} \neq \frac{p_n}{q_n}, \text{ then } \left| x - \frac{p_n}{q_n} \right| < \left| x - \frac{p}{q} \right|.$$

§6.4 Continued fractions

Theorem

Let $b, q \in \mathbb{N}$ *be given and let* $[a_0; a_1, \cdots, a_n]$ *be the continued fraction of their quotient; that is*

$$\frac{b}{q} = [a_0; a_1, \cdots, a_n] .$$

If $c, r \in \mathbb{N}$ *are such that*

$$\left|\frac{b}{q} - \frac{c}{r}\right| < \frac{1}{2r^2} ,$$

then $\frac{c}{r}$ *is a convergent of the continued fraction of* $\frac{b}{q}$*; that is, there exists a* $j \in \{0, 1, \cdots, n\}$ *such that*

$$\frac{c}{r} = [a_0; a_1, \cdots, a_j] = \frac{p_j}{q_j}$$

where p_j *and* q_j *are as constructed by* (2).

§6.5 Efficiency of Shor's Algorithm

§6.5.1 Shor's period-finding algorithm

Shor's algorithm can be applied to find the period of a more general class of periodic functions.

Theorem

Let $f : \mathbb{N} \cup \{0\} \to \mathbb{N} \cup \{0\}$ be a periodic function with period r satisfying $19 \leqslant r < 2^{L/2}$ for some $L \in \mathbb{N}$ such that f is injective within one period and is bounded by $2^K - 1$, and U_f be an $(L+K)$-qubit quantum gate satisfying

$$U_f|a\rangle|b\rangle = |a\rangle|b \oplus f(a)\rangle\,, \qquad \forall\, a \in \{0,1\}^L, b \in \{0,1\}^K.$$

Then each application of Shor's algorithm provides the period r with a probability of at least $\dfrac{1}{10 \ln L}$.

§6.5 Efficiency of Shor's Algorithm

Proof.

Let $M = \max\{f(a) \,|\, 0 \leqslant a \leqslant 2^L - 1\}$ and $K \in \mathbb{N}$ with $M < 2^K$, and $\mathbb{H}$ be the usual qubit Hilbert space with basis $\{|0\rangle, |1\rangle\}$. Set $|\psi_0\rangle = |0^L\rangle \otimes |0^K\rangle$. With I_K denoting the identity map on $\mathbb{H}^{\otimes K}$,

$$|\psi_1\rangle \equiv (\mathrm{H}^{\otimes L} \otimes \mathrm{I}_K)|\psi_0\rangle = \frac{1}{\sqrt{2^L}} \sum_{a=0}^{2^L-1} |a\rangle \otimes |0^K\rangle .$$

Applying U_f to $|\psi_1\rangle$, we find that

$$|\psi_2\rangle = U_f|\psi_1\rangle = \frac{1}{\sqrt{2^L}} \sum_{a=0}^{2^L-1} |a\rangle \otimes |f(a)\rangle . \qquad \square$$

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Define $m \equiv \left[\frac{2^L - 1}{r}\right]$, the largest integer smaller than $\frac{2^L-1}{r}$, and $R \equiv (2^L - 1) \bmod r$. Then

$$|\psi_2\rangle = \frac{1}{\sqrt{2^L}} \sum_{a=0}^{2^L-1} |a\rangle \otimes |f(a)\rangle = \frac{1}{\sqrt{2^L}} \sum_{0 \leqslant jr+s < 2^L} |jr+s\rangle \otimes |f(jr+s)\rangle$$

$$= \frac{1}{\sqrt{2^L}} \sum_{j=0}^{m-1} \sum_{s=0}^{r-1} |jr+s\rangle \otimes |f(s)\rangle + \sum_{s=0}^{R} |mr+s\rangle \otimes |f(s)\rangle .$$

Define $m_s = m - \mathbf{1}_{(R,\infty)}(s)$; that is, $m_s = m$ if $s \leqslant R$ and $m_s = m-1$ if $s > R$. Then

$$|\psi_2\rangle = \frac{1}{\sqrt{2^L}} \sum_{s=0}^{r-1} \sum_{j=0}^{m_s} |jr+s\rangle \otimes |f(s)\rangle .$$

□

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Define $m \equiv \left[\frac{2^L-1}{r}\right]$, the largest integer smaller than $\frac{2^L-1}{r}$, and $R \equiv (2^L-1) \bmod r$. Then

$$|\psi_2\rangle = \frac{1}{\sqrt{2^L}} \sum_{a=0}^{2^L-1} |a\rangle \otimes |f(a)\rangle = \frac{1}{\sqrt{2^L}} \sum_{0\leqslant jr+s<2^L} |jr+s\rangle \otimes |f(jr+s)\rangle$$

$$= \frac{1}{\sqrt{2^L}} \sum_{j=0}^{m-1} \sum_{s=0}^{r-1} |jr+s\rangle \otimes |f(s)\rangle + \sum_{s=0}^{R} |mr+s\rangle \otimes |f(s)\rangle .$$

Define $m_s = m - \mathbf{1}_{(R,\infty)}(s)$; that is, $m_s = m$ if $s \leqslant R$ and $m_s = m-1$ if $s > R$. Then

$$|\psi_2\rangle = \frac{1}{\sqrt{2^L}} \sum_{s=0}^{r-1} \sum_{j=0}^{m_s} |jr+s\rangle \otimes |f(s)\rangle .$$

□

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Next we apply the quantum Fourier transform to the first L qubits of $|\psi_2\rangle$ and obtain that

$$|\psi_3\rangle \equiv (F_{2^L} \otimes \mathrm{I}_B)|\psi_2\rangle = \frac{1}{\sqrt{2^L}} \sum_{s=0}^{r-1} \sum_{j=0}^{m_s} (F_{2^L}|jr+s\rangle) \otimes |f(s)\rangle$$

$$= \frac{1}{2^L} \sum_{s=0}^{r-1} \sum_{j=0}^{m_s} \sum_{b=0}^{2^L-1} \exp\left(2\pi i \frac{(jr+s)b}{2^L}\right)|b\rangle \otimes |f(s)\rangle .$$

□

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Now we measure the input register, and let $P(b)$ denote the probability of observing $|b\rangle$ upon measurement. For $b \in \{0, \cdots, 2^L - 1\}$,

$$\begin{aligned} P(b) &= \frac{1}{2^{2L}} \sum_{s=0}^{r-1} \Big| \sum_{j=0}^{m_s} \exp\Big(2\pi i \frac{(jr+s)b}{2^L}\Big)\Big|^2 \\ &= \frac{1}{2^{2L}} \sum_{s=0}^{r-1} \Big[\sum_{j_1,j_2=0}^{m_s} \exp\Big(2\pi i \frac{(j_1 r+s)b}{2^L}\Big) \exp\Big(-2\pi i \frac{(j_2 r+s)b}{2^L}\Big)\Big] \\ &= \frac{1}{2^{2L}} \sum_{s=0}^{r-1} \Big[\sum_{j_1,j_2=0}^{m_s} \exp\Big(2\pi i \frac{j_1 rb}{2^L}\Big) \exp\Big(-2\pi i \frac{j_2 rb}{2^L}\Big)\Big] \\ &= \frac{1}{2^{2L}} \sum_{s=0}^{r-1} \Big| \Big[\sum_{j=0}^{m_s} \exp\Big(2\pi i \frac{jrb}{2^L}\Big)\Big|^2 . \end{aligned}$$

□

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Since

$$\sum_{j=0}^{d} a^j = \begin{cases} d+1 & \text{if } a = 1\,, \\ \dfrac{1-a^{d+1}}{1-a} & \text{if } a \neq 1\,, \end{cases}$$

we obtain that

$$\sum_{j=0}^{m_s} \exp\left(2\pi i \frac{jrb}{2^L}\right) = \begin{cases} m_s + 1 & \text{if } \dfrac{rb}{2^L} \in \mathbb{N} \cup \{0\}\,, \\ \dfrac{1 - e^{2\pi i \frac{(m_s+1)rb}{2^L}}}{1 - e^{2\pi i \frac{rb}{2^L}}} & \text{if } \dfrac{rb}{2^L} \notin \mathbb{N} \cup \{0\}\,; \end{cases}$$

thus

$$P(b) = \begin{cases} \dfrac{1}{2^{2L}} \displaystyle\sum_{s=0}^{r-1} (m_s + 1)^2 & \text{if } \dfrac{rb}{2^L} \in \mathbb{N} \cup \{0\}\,, \\ \dfrac{1}{2^{2L}} \displaystyle\sum_{s=0}^{r-1} \left| \dfrac{1 - e^{2\pi i \frac{(m_s+1)rb}{2^L}}}{1 - e^{2\pi i \frac{rb}{2^L}}} \right|^2 & \text{if } \dfrac{rb}{2^L} \notin \mathbb{N} \cup \{0\}\,. \end{cases}$$

□

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Define

$$E=\Big\{b\in\{0,\cdots,2^L-1\}\,\Big|\,\Big|\frac{b}{2^L}-\frac{c}{r}\Big|<\frac{1}{2r^2}\text{ for some (unique) } c\in\mathbb{Z}_r^*\Big\},$$

$$B=\Big\{b\in\{0,\cdots,2^L-1\}\,\Big|\,|rb-c2^L|\leqslant\frac{r}{2}\text{ for some (unique) } c\in\mathbb{Z}_r^*\Big\},$$

here we recall that $\mathbb{Z}_r^*$ is the collection of numbers in $\{1,\cdots,r-1\}$ that are coprime to r.

The fact that $r<2^{L/2}$ implies that if $b\in B$,

$$\Big|\frac{b}{2^L}-\frac{c}{r}\Big|=\frac{1}{r2^L}\big|rb-c2^L\big|\leqslant\frac{r}{2}\cdot\frac{1}{r2^L}=\frac{1}{2\cdot 2^L}<\frac{1}{2r^2}\,.$$

In other words, $B\subseteq E$. Our goal is to show that the probability of measuring a $b\in E$ is **not** "too small" by finding a lower bound for the probability of measuring a $b\in B$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Define

$$E = \left\{ b \in \{0, \cdots, 2^L - 1\} \,\middle|\, \left|\frac{b}{2^L} - \frac{c}{r}\right| < \frac{1}{2r^2} \text{ for some (unique) } c \in \mathbb{Z}_r^* \right\},$$

$$B = \left\{ b \in \{0, \cdots, 2^L - 1\} \,\middle|\, \left|rb - c2^L\right| \leqslant \frac{r}{2} \text{ for some (unique) } c \in \mathbb{Z}_r^* \right\},$$

here we recall that $\mathbb{Z}_r^*$ is the collection of numbers in $\{1, \cdots, r-1\}$ that are coprime to r.

The fact that $r < 2^{L/2}$ implies that if $b \in B$,

$$\left|\frac{b}{2^L} - \frac{c}{r}\right| = \frac{1}{r2^L}\left|rb - c2^L\right| \leqslant \frac{r}{2} \cdot \frac{1}{r2^L} = \frac{1}{2 \cdot 2^L} < \frac{1}{2r^2}\,.$$

In other words, $B \subseteq E$. Our goal is to show that the probability of measuring a $b \in E$ is **not** "too small" by finding a lower bound for the probability of measuring a $b \in B$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Define

$$E = \Big\{ b \in \{0, \cdots, 2^L - 1\} \,\Big|\, \Big|\frac{b}{2^L} - \frac{c}{r}\Big| < \frac{1}{2r^2} \text{ for some (unique) } c \in \mathbb{Z}_r^* \Big\},$$

$$B = \Big\{ b \in \{0, \cdots, 2^L - 1\} \,\Big|\, \big|rb - c2^L\big| \leqslant \frac{r}{2} \text{ for some (unique) } c \in \mathbb{Z}_r^* \Big\},$$

here we recall that $\mathbb{Z}_r^*$ is the collection of numbers in $\{1, \cdots, r-1\}$ that are coprime to r.

The fact that $r < 2^{L/2}$ implies that if $b \in B$,

$$\Big|\frac{b}{2^L} - \frac{c}{r}\Big| = \frac{1}{r2^L}\big|rb - c2^L\big| \leqslant \frac{r}{2} \cdot \frac{1}{r2^L} = \frac{1}{2 \cdot 2^L} < \frac{1}{2r^2}\,.$$

In other words, $B \subseteq E$. Our goal is to show that the probability of measuring a $b \in E$ is **not** "too small" by finding a lower bound for the probability of measuring a $b \in B$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Let $b \in B$.

1. The case $\dfrac{rb}{2^L} \in \mathbb{N} \cup \{0\}$: In this case

$$P(b) = \frac{1}{2^{2L}} \sum_{s=0}^{r-1} (m_s+1)^2 = \frac{1}{2^{2L}} \Big[\sum_{s=0}^{R} (m+1)^2 + \sum_{s=R+1}^{r-1} m^2 \Big]$$
$$\geqslant \frac{1}{2^{2L}} \Big[(R+1)m^2 + (r-1-R)m^2 \Big] = \frac{1}{r}\Big(\frac{rm}{2^L}\Big)^2 .$$

Since $m = \left[\dfrac{2^L-1}{r}\right]$, $r-1 \geqslant (2^L-1) \bmod r \geqslant 2^L - 1 - mr$; thus the fact that $r < 2^{\frac{L}{2}}$ implies that $\dfrac{mr}{2^L} \geqslant 1 - \dfrac{r}{2^L} > 1 - \dfrac{1}{\sqrt{2^L}}$. Therefore,

$$P(b) \geqslant \frac{1}{r}\Big(1 - \frac{1}{\sqrt{2^L}}\Big)^2 > \frac{1}{r}\Big(1 - \frac{1}{2^{L/2-1}}\Big).$$

□

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Let $b \in B$.

1. The case $\frac{rb}{2^L} \in \mathbb{N} \cup \{0\}$: In this case

$$P(b) = \frac{1}{2^{2L}} \sum_{s=0}^{r-1} (m_s + 1)^2 = \frac{1}{2^{2L}} \Big[\sum_{s=0}^{R} (m+1)^2 + \sum_{s=R+1}^{r-1} m^2 \Big]$$
$$\geqslant \frac{1}{2^{2L}} \Big[(R+1)m^2 + (r-1-R)m^2 \Big] = \frac{1}{r} \Big(\frac{rm}{2^L} \Big)^2 .$$

Since $m = \Big[\frac{2^L - 1}{r} \Big]$, $r - 1 \geqslant (2^L - 1) \bmod r \geqslant 2^L - 1 - mr$; thus the fact that $r < 2^{\frac{L}{2}}$ implies that $\frac{mr}{2^L} \geqslant 1 - \frac{r}{2^L} > 1 - \frac{1}{\sqrt{2^L}}$.

Therefore,

$$P(b) \geqslant \frac{1}{r} \Big(1 - \frac{1}{\sqrt{2^L}} \Big)^2 > \frac{1}{r} \Big(1 - \frac{1}{2^{L/2-1}} \Big) .$$

□

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Let $b \in B$.

1. The case $\dfrac{rb}{2^L} \in \mathbb{N} \cup \{0\}$: In this case

$$P(b) = \frac{1}{2^{2L}} \sum_{s=0}^{r-1} (m_s + 1)^2 = \frac{1}{2^{2L}} \Big[\sum_{s=0}^{R} (m+1)^2 + \sum_{s=R+1}^{r-1} m^2 \Big]$$
$$\geqslant \frac{1}{2^{2L}} \Big[(R+1)m^2 + (r-1-R)m^2 \Big] = \frac{1}{r} \Big(\frac{rm}{2^L} \Big)^2 .$$

Since $m = \Big[\dfrac{2^L - 1}{r}\Big]$, $r - 1 \geqslant (2^L - 1) \bmod r \geqslant 2^L - 1 - mr$; thus the fact that $r < 2^{\frac{L}{2}}$ implies that $\dfrac{mr}{2^L} \geqslant 1 - \dfrac{r}{2^L} > 1 - \dfrac{1}{\sqrt{2^L}}$. Therefore,

$$P(b) \geqslant \frac{1}{r} \Big(1 - \frac{1}{\sqrt{2^L}} \Big)^2 > \frac{1}{r} \Big(1 - \frac{1}{2^{L/2-1}} \Big) .$$

□

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

2. The case $\frac{rb}{2^L} \notin \mathbb{N} \cup \{0\}$: Suppose that $c \in \mathbb{Z}_r^*$ satisfies

$$\left|rb - c2^L\right| \leqslant \frac{r}{2}. \tag{3}$$

Then

$$P(b) = \frac{1}{2^{2L}} \sum_{s=0}^{r-1} \left| \frac{1 - e^{2\pi i \frac{(m_s+1)rb}{2^L}}}{1 - e^{2\pi i \frac{rb}{2^L}}} \right|^2 = \frac{1}{2^{2L}} \sum_{s=0}^{r-1} \left| \frac{1 - e^{2\pi i \frac{(m_s+1)(rb - c2^L)}{2^L}}}{1 - e^{2\pi i \frac{rb - c2^L}{2^L}}} \right|^2$$

$$= \frac{1}{2^{2L}} \sum_{s=0}^{r-1} \frac{\sin^2 \pi \frac{rb - c2^L}{2^L}(m_s + 1)}{\sin^2 \pi \frac{rb - c2^L}{2^L}},$$

where we have used the identity $|1 - e^{i\theta}| = 2\left|\sin \frac{\theta}{2}\right|$ to conclude the last equality. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Let $\alpha = \pi\frac{rb - c2^L}{2^L}$. Then

$$|\alpha| \leqslant \frac{\pi}{2^L}\cdot\frac{r}{2} < \frac{\pi}{2^{\frac{L}{2}+1}} \ll \frac{\pi}{2}.$$

Within this range, the function $\beta \mapsto \dfrac{\sin^2\beta(m_s+1)}{\sin^2\beta}$ cannot attain its minimum in the interior of the interval and we have

$$\frac{\sin^2\pi\frac{rb-c2^L}{2^L}(m_s+1)}{\sin^2\pi\frac{rb-c2^L}{2^L}} = \frac{\sin^2\alpha(m_s+1)}{\sin^2\alpha} \geqslant \frac{\sin^2\frac{\pi r}{2^{L+1}}(m_s+1)}{\sin^2\frac{\pi r}{2^{L+1}}}$$

for all $|\alpha| \leqslant \frac{\pi}{2^L}\cdot\frac{r}{2}$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Since $m \leqslant m_s + 1 \leqslant m + 1$ and $R = (2^L - 1) \bmod r$,

$$\frac{r(m_s+1)}{2^L} \geqslant \frac{mr}{2^L} = \frac{mr+R+1}{2^L} - \frac{R+1}{2^L} = 1 - \frac{R+1}{2^L} \geqslant 1 - \frac{r}{2^L}$$

and

$$\frac{r(m_s+1)}{2^L} \leqslant \frac{r(m+1)}{2^L} = \frac{mr+R+1}{2^L} + \frac{r-R-1}{2^L} \leqslant 1 + \frac{r}{2^L}\,.$$

Therefore, using $\sin^2 x \leqslant x^2$ and $\cos x \geqslant 1 - \frac{x^2}{2}$ $\forall\, x \in \mathbb{R}$,

$$\frac{\sin^2 \pi \frac{rb-c2^L}{2^L}(m_s+1)}{\sin^2 \pi \frac{rb-c2^L}{2^L}} \geqslant \frac{\sin^2 \frac{\pi r(m_s+1)}{2^{L+1}}}{\sin^2 \frac{\pi r}{2^{L+1}}} \geqslant \left(\frac{2^{L+1}}{\pi r}\right)^2 \sin^2 \frac{\pi r(m_s+1)}{2^{L+1}}$$

$$\geqslant \left(\frac{2^{L+1}}{\pi r}\right)^2 \sin^2\left[\frac{\pi}{2}\left(1-\frac{r}{2^L}\right)\right] \geqslant \left(\frac{2^{L+1}}{\pi r}\right)^2\left[1-\frac{1}{2}\left(\frac{\pi}{2}\frac{r}{2^L}\right)^2\right]^2$$

$$\geqslant \frac{2^{2L+2}}{\pi^2 r^2}\left[1-\left(\frac{\pi}{2}\frac{1}{\sqrt{2^L}}\right)^2\right] = \frac{2^{2L+2}}{\pi^2 r^2}\left(1-\frac{\pi^2}{2^{L+2}}\right).$$

□

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Since $m \leqslant m_s + 1 \leqslant m + 1$ and $R = (2^L - 1) \bmod r$,

$$\frac{r(m_s+1)}{2^L} \geqslant \frac{mr}{2^L} = \frac{mr+R+1}{2^L} - \frac{R+1}{2^L} = 1 - \frac{R+1}{2^L} \geqslant 1 - \frac{r}{2^L}$$

and

$$\frac{r(m_s+1)}{2^L} \leqslant \frac{r(m+1)}{2^L} = \frac{mr+R+1}{2^L} + \frac{r-R-1}{2^L} \leqslant 1 + \frac{r}{2^L}.$$

Therefore, using $\sin^2 x \leqslant x^2$ and $\cos x \geqslant 1 - \frac{x^2}{2}$ $\forall\, x \in \mathbb{R}$,

$$\frac{\sin^2 \pi \frac{rb-c2^L}{2^L}(m_s+1)}{\sin^2 \pi \frac{rb-c2^L}{2^L}} \geqslant \frac{\sin^2 \frac{\pi r(m_s+1)}{2^{L+1}}}{\sin^2 \frac{\pi r}{2^{L+1}}} \geqslant \left(\frac{2^{L+1}}{\pi r}\right)^2 \sin^2 \frac{\pi r(m_s+1)}{2^{L+1}}$$

$$\geqslant \left(\frac{2^{L+1}}{\pi r}\right)^2 \sin^2 \left[\frac{\pi}{2}\left(1 - \frac{r}{2^L}\right)\right] \geqslant \left(\frac{2^{L+1}}{\pi r}\right)^2 \left[1 - \frac{1}{2}\left(\frac{\pi}{2}\frac{r}{2^L}\right)^2\right]^2$$

$$\geqslant \frac{2^{2L+2}}{\pi^2 r^2}\left[1 - \left(\frac{\pi}{2}\frac{1}{\sqrt{2^L}}\right)^2\right] = \frac{2^{2L+2}}{\pi^2 r^2}\left(1 - \frac{\pi^2}{2^{L+2}}\right).$$

□

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Since $m \leqslant m_s + 1 \leqslant m + 1$ and $R = (2^L - 1) \bmod r$,

$$\frac{r(m_s+1)}{2^L} \geqslant \frac{mr}{2^L} = \frac{mr+R+1}{2^L} - \frac{R+1}{2^L} = 1 - \frac{R+1}{2^L} \geqslant 1 - \frac{r}{2^L}$$

and

$$\frac{r(m_s+1)}{2^L} \leqslant \frac{r(m+1)}{2^L} = \frac{mr+R+1}{2^L} + \frac{r-R-1}{2^L} \leqslant 1 + \frac{r}{2^L}.$$

Therefore, using $\sin^2 x \leqslant x^2$ and $\cos x \geqslant 1 - \frac{x^2}{2}$ $\forall\, x \in \mathbb{R}$,

$$\frac{\sin^2 \pi \frac{rb-c2^L}{2^L}(m_s+1)}{\sin^2 \pi \frac{rb-c2^L}{2^L}} \geqslant \frac{\sin^2 \frac{\pi r(m_s+1)}{2^{L+1}}}{\sin^2 \frac{\pi r}{2^{L+1}}} \geqslant \left(\frac{2^{L+1}}{\pi r}\right)^2 \sin^2 \frac{\pi r(m_s+1)}{2^{L+1}}$$

$$\geqslant \left(\frac{2^{L+1}}{\pi r}\right)^2 \sin^2\left[\frac{\pi}{2}\left(1-\frac{r}{2^L}\right)\right] \geqslant \left(\frac{2^{L+1}}{\pi r}\right)^2 \left[1-\frac{1}{2}\left(\frac{\pi}{2}\frac{r}{2^L}\right)^2\right]^2$$

$$\geqslant \frac{2^{2L+2}}{\pi^2 r^2}\left[1-\left(\frac{\pi}{2}\frac{1}{\sqrt{2^L}}\right)^2\right] = \frac{2^{2L+2}}{\pi^2 r^2}\left(1-\frac{\pi^2}{2^{L+2}}\right).$$

□

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Therefore,

$$P(b) \geqslant \frac{1}{2^{2L}} \sum_{s=0}^{r-1} \frac{2^{2L+2}}{\pi^2 r^2}\Big(1 - \frac{\pi^2}{2^{L+2}}\Big) = \frac{r}{2^{2L}} \frac{2^{2L+2}}{\pi^2 r^2}\Big(1 - \frac{\pi^2}{2^{L+2}}\Big)$$
$$= \frac{4}{\pi^2 r}\Big(1 - \frac{\pi^2}{2^{L+2}}\Big).$$

For $L \geqslant 4$, we have

$$\frac{4}{\pi^2 r}\Big(1 - \frac{\pi^2}{2^{L+2}}\Big) \leqslant \frac{1}{r}\Big(1 - \frac{1}{2^{L/2-1}}\Big);$$

thus

$$P_{\min} \equiv \frac{4}{\pi^2 r}\Big(1 - \frac{\pi^2}{2^{L+2}}\Big) \leqslant P(b) \quad \text{if } b \in B \text{ and } L \geqslant 4.$$

□

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Therefore,

$$P(b) \geqslant \frac{1}{2^{2L}} \sum_{s=0}^{r-1} \frac{2^{2L+2}}{\pi^2 r^2}\Big(1 - \frac{\pi^2}{2^{L+2}}\Big) = \frac{r}{2^{2L}} \frac{2^{2L+2}}{\pi^2 r^2}\Big(1 - \frac{\pi^2}{2^{L+2}}\Big)$$
$$= \frac{4}{\pi^2 r}\Big(1 - \frac{\pi^2}{2^{L+2}}\Big).$$

For $L \geqslant 4$, we have

$$\frac{4}{\pi^2 r}\Big(1 - \frac{\pi^2}{2^{L+2}}\Big) \leqslant \frac{1}{r}\Big(1 - \frac{1}{2^{L/2-1}}\Big);$$

thus

$$P_{\min} \equiv \frac{4}{\pi^2 r}\Big(1 - \frac{\pi^2}{2^{L+2}}\Big) \leqslant P(b) \quad \text{if } b \in B \text{ and } L \geqslant 4.$$

□

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Now we find a lower bound for $\mathbf{P}(E)$, the probability of measuring an element of E. By the definition of B for any $b \in B$ there exists $c \in \mathbb{Z}_r^*$ satisfying

$$\left|rb - c2^L\right| \leqslant \frac{r}{2}\,. \tag{3}$$

Moreover, if $c_1, c_2 \in \mathbb{Z}_r^*$ satisfy $\left|rb - c_1 2^L\right| \leqslant \frac{r}{2}$ and $\left|rb - c_2 2^L\right| \leqslant \frac{r}{2}$, then

$$\begin{aligned}\left|c_1 - c_2\right| &\leqslant \left|c_1 - \frac{rb}{2^L}\right| + \left|c_2 - \frac{rb}{2^L}\right| \leqslant \frac{1}{2^L}\Big(\left|rb - c_1 2^L\right| + \left|rb - c_2 2^L\right|\Big) \\ &\leqslant \frac{r}{2^L} < 1\,.\end{aligned}$$

Therefore, for any $b \in B$ there exists a unique $c = c_b \in \mathbb{Z}_r^*$ satisfying (3). □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

On the other hand, every $c \in \mathbb{Z}_r^*$ corresponds to a unique $b = b_c \in \{0, 1, \cdots, 2^L - 1\}$ such that (3) holds: if b_1 and b_2 both satisfy (3), then $|b_1 - b_2| = 1$ and

$$(b_1 + b_2)r = c2^{L+1}$$

which, by the fact that $b_1 + b_2$ is odd, implies that $2^{L+1}|r$, a contradiction to that $r < 2^{L/2}$. Therefore, there is a one-to-one correspondence between $\mathbb{Z}_r^*$ and B.

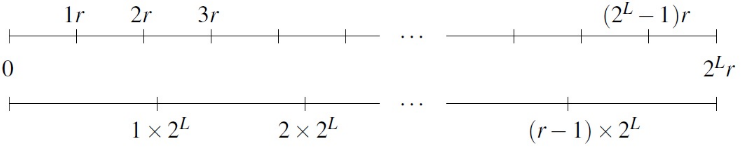

Figure 3: The distribution of br (↑) and $c2^L$ (↓) for various b and c. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

On the other hand, every $c \in \mathbb{Z}_r^*$ corresponds to a unique $b = b_c \in \{0, 1, \cdots, 2^L - 1\}$ such that (3) holds: if b_1 and b_2 both satisfy (3), then $|b_1 - b_2| = 1$ and

$$(b_1 + b_2)r = c2^{L+1}$$

which, by the fact that $b_1 + b_2$ is odd, implies that $2^{L+1}|r$, a contradiction to that $r < 2^{L/2}$. Therefore, there is a one-to-one correspondence between $\mathbb{Z}_r^*$ and B.

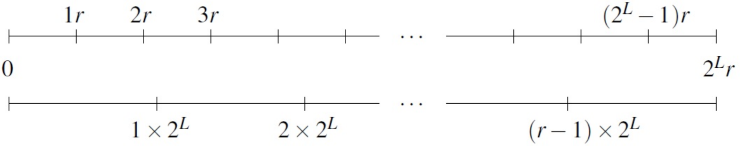

Figure 3: The distribution of br (↑) and $c2^L$ (↓) for various b and c. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

As a consequence, if $L \geqslant 4$,

$$\mathbf{P}(E) = \sum_{b \in E} P(b) \geqslant \sum_{b \in B} P(b) \geqslant \sum_{b \in B} \frac{4}{\pi^2 r}\Big(1 - \frac{\pi^2}{2^{L+2}}\Big)$$
$$= \frac{4\,\#B}{\pi^2 r}\Big(1 - \frac{\pi^2}{2^{L+2}}\Big) = \frac{4\,\varphi(r)}{\pi^2 r}\Big(1 - \frac{\pi^2}{2^{L+2}}\Big).$$

A famous result in number theory implies that

$$\frac{r}{\varphi(r)} < 4 \ln \ln r \qquad \forall\, r \geqslant 19\,;$$

thus if $r \geqslant 19$ (so that $L \geqslant 9$),

$$\mathbf{P}(E) \geqslant \frac{4}{\pi^2}\Big(1 - \frac{\pi^2}{2^{11}}\Big)\frac{1}{4 \ln \ln r} > \frac{1}{10 \ln L}\,.$$

□

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

As a consequence, if $L \geqslant 4$,

$$\mathbf{P}(E) = \sum_{b \in E} P(b) \geqslant \sum_{b \in B} P(b) \geqslant \sum_{b \in B} \frac{4}{\pi^2 r}\Big(1 - \frac{\pi^2}{2^{L+2}}\Big)$$
$$= \frac{4\,\#B}{\pi^2 r}\Big(1 - \frac{\pi^2}{2^{L+2}}\Big) = \frac{4\,\varphi(r)}{\pi^2 r}\Big(1 - \frac{\pi^2}{2^{L+2}}\Big).$$

A famous result in number theory implies that

$$\frac{r}{\varphi(r)} < 4 \ln \ln r \qquad \forall\, r \geqslant 19\,;$$

thus if $r \geqslant 19$ (so that $L \geqslant 9$),

$$\mathbf{P}(E) \geqslant \frac{4}{\pi^2}\Big(1 - \frac{\pi^2}{2^{11}}\Big)\frac{1}{4 \ln \ln r} > \frac{1}{10 \ln L}\,.$$

□

§6.5 Efficiency of Shor's Algorithm

§6.5.2 The period of $f(a) = x^a$ mod N is most likely even

In this sub-section we focus on proving the following

Theorem

Let $N \in \mathbb{N}$ be odd with prime factorization $N = \prod_{j=1}^{J} p_j^{\nu_j}$, where p_1, $\cdots$, p_J are distinct prime numbers. For a randomly chosen $b \in \mathbb{Z}_N^*$, the probability of that $r \equiv \min\left\{r \in \mathbb{N} \,\middle|\, b^r = 1 \text{ mod } N\right\}$ is even and $b^{r/2} + 1 \bmod N \neq 0$ is at least $1 - \frac{1}{2^{J-1}}$.

In the application of the factoring algorithm proposed in the previous sections, $J = 2$ so that the probability of that for a randomly chosen $b \in \mathbb{Z}_N^*$ the number $\gcd(b^{r/2} + 1, N)$ is a prime factor of N is at least $1/2$.

§6.5 Efficiency of Shor's Algorithm

§6.5.2 The period of $f(a) = x^a$ mod N is most likely even

In this sub-section we focus on proving the following

Theorem

Let $N \in \mathbb{N}$ be odd with prime factorization $N = \prod_{j=1}^{J} p_j^{\nu_j}$, where $p_1, \cdots, p_J$ are distinct prime numbers. For a randomly chosen $b \in \mathbb{Z}_N^*$, the probability of that $r \equiv \min\left\{r \in \mathbb{N} \,\middle|\, b^r = 1 \text{ mod } N\right\}$ is even and $b^{r/2} + 1 \bmod N \neq 0$ is at least $1 - \dfrac{1}{2^{J-1}}$.

In the application of the factoring algorithm proposed in the previous sections, $J = 2$ so that the probability of that for a randomly chosen $b \in \mathbb{Z}_N^*$ the number $\gcd(b^{r/2}+1, N)$ is a prime factor of N is at least $1/2$.

§6.5 Efficiency of Shor's Algorithm

Let $N \in \mathbb{N}$. Recall that $\mathbb{Z}_N^*$ consists of numbers from $\{1, 2, \cdots, N-1\}$ that is coprime to N; that is,

$$\mathbb{Z}_N^* = \left\{ n \in \mathbb{N} \,\middle|\, 1 \leqslant n \leqslant N-1 \text{ and } \gcd(n, N) = 1 \right\}.$$

The number of elements in $\mathbb{Z}_N^* = \varphi(N)$, where φ is the Euler function. Before proceeding, we introduce some terminologies.

Definition

Let $b, N \in \mathbb{N}$ with $\gcd(b, N) = 1$. The order of b in $\mathbb{Z}_N^*$, denoted by $\text{ord}_N(b)$, is the period of the function $f(x) = b^x - 1 \mod N$. In other words,

$$\text{ord}_N(b) = \min \left\{ r \in \mathbb{N} \,\middle|\, b^r = 1 \mod N \right\}.$$

If $\text{ord}_N(b) = \varphi(N)$, then b is called a **primitive root** modulo N.

§6.5 Efficiency of Shor's Algorithm

Let $N \in \mathbb{N}$. Recall that $\mathbb{Z}_N^*$ consists of numbers from $\{1, 2, \cdots, N-1\}$ that is coprime to N; that is,

$$\mathbb{Z}_N^* = \left\{n \in \mathbb{N} \,\middle|\, 1 \leqslant n \leqslant N-1 \text{ and } \gcd(n, N) = 1\right\}.$$

The number of elements in $\mathbb{Z}_N^* = \varphi(N)$, where φ is the Euler function. Before proceeding, we introduce some terminologies.

Definition

Let $b, N \in \mathbb{N}$ with $\gcd(b, N) = 1$. The order of b in $\mathbb{Z}_N^*$, denoted by $\text{ord}_N(b)$, is the period of the function $f(x) = b^x - 1 \bmod N$. In other words,

$$\text{ord}_N(b) = \min\left\{r \in \mathbb{N} \,\middle|\, b^r = 1 \bmod N\right\}.$$

If $\text{ord}_N(b) = \varphi(N)$, then b is called a **primitive root** modulo N.

§6.5 Efficiency of Shor's Algorithm

Theorem

Let $a, b, N \in \mathbb{N}$ with $\gcd(a, N) = 1 = \gcd(b, N)$. Then the following statements hold.

1. *For all $k \in \mathbb{N}$, $a^k = 1$ mod N if and only if $\mathrm{ord}_N(a) | k$.*
2. *$\mathrm{ord}_N(a) | \varphi(N)$; that is, $\mathrm{ord}_N(a)$ is a factor of $\varphi(N)$.*
3. *If $\gcd(\mathrm{ord}_N(a), \mathrm{ord}_N(b)) = 1$, then $\mathrm{ord}_N(ab) = \mathrm{ord}_N(a)\mathrm{ord}_N(b)$.*
4. *If a is a primitive root modulo N; that is, $\mathrm{ord}_N(a) = \varphi(N)$, then we also have*
   - ⓐ $\mathbb{Z}_N^* = \left\{ a^j \bmod N \,\middle|\, 1 \leqslant j \leqslant \varphi(N) \right\}$.
   - ⓑ *If $b = a^j$ mod N for some $j \in \mathbb{N}$, then*

$$\mathrm{ord}_N(b) = \mathrm{ord}_N(a^j) = \frac{\varphi(N)}{\gcd(j, \varphi(N))}. \tag{4}$$

§6.5 Efficiency of Shor's Algorithm

Proof.

Let $a, b, N \in \mathbb{N}$ with $\gcd(a, N) = 1 = \gcd(b, N)$.

1. ("$\Rightarrow$") Let $k \in \mathbb{N}$ satisfying $a^k = 1 \bmod N$. Then $k \geqslant \text{ord}_N(a)$. Let $c = k \bmod \text{ord}_N(a)$; that is, there exists $q \in \mathbb{N}$ such that $k = q \cdot \text{ord}_N(a) + c$ for some $c \in \{0, 1, \cdots, \text{ord}_N(a) - 1\}$. Then
$$1 = a^k \bmod N = a^{q \cdot \text{ord}_N(a) + c} \bmod N = a^c \bmod N;$$
thus by the definition of the order we must have $c = 0$. Therefore, $\text{ord}_N(a) | k$.

   ("$\Leftarrow$") Suppose that $\text{ord}_N(a) | k$. Then $k = q \cdot \text{ord}_N(a)$ for some $q \in \mathbb{N}$. Therefore,
$$a^k \bmod N = a^{q \cdot \text{ord}_N(a)} \bmod N = 1.$$

2. By one of previous theorems, we know that $a^{\varphi(N)} = 1 \bmod N$; thus ② follows from ①.

□

§6.5 Efficiency of Shor's Algorithm

Proof.

Let $a, b, N \in \mathbb{N}$ with $\gcd(a, N) = 1 = \gcd(b, N)$.

1. ("$\Rightarrow$") Let $k \in \mathbb{N}$ satisfying $a^k = 1 \bmod N$. Then $k \geqslant \mathrm{ord}_N(a)$. Let $c = k \bmod \mathrm{ord}_N(a)$; that is, there exists $q \in \mathbb{N}$ such that $k = q \cdot \mathrm{ord}_N(a) + c$ for some $c \in \{0, 1, \cdots, \mathrm{ord}_N(a) - 1\}$. Then
$$1 = a^k \bmod N = a^{q \cdot \mathrm{ord}_N(a) + c} \bmod N = a^c \bmod N;$$
thus by the definition of the order we must have $c = 0$. Therefore, $\mathrm{ord}_N(a) | k$.
("$\Leftarrow$") Suppose that $\mathrm{ord}_N(a) | k$. Then $k = q \cdot \mathrm{ord}_N(a)$ for some $q \in \mathbb{N}$. Therefore,
$$a^k \bmod N = a^{q \cdot \mathrm{ord}_N(a)} \bmod N = 1.$$
2. By one of previous theorems, we know that $a^{\varphi(N)} = 1 \bmod N$; thus ② follows from ①. □

§6.5 Efficiency of Shor's Algorithm

Proof.

Let $a, b, N \in \mathbb{N}$ with $\gcd(a, N) = 1 = \gcd(b, N)$.

1. ("⇒") Let $k \in \mathbb{N}$ satisfying $a^k = 1 \mod N$. Then $k \geqslant \mathrm{ord}_N(a)$. Let $c = k \mod \mathrm{ord}_N(a)$; that is, there exists $q \in \mathbb{N}$ such that $k = q \cdot \mathrm{ord}_N(a) + c$ for some $c \in \{0, 1, \cdots, \mathrm{ord}_N(a) - 1\}$. Then
$$1 = a^k \mod N = a^{q \cdot \mathrm{ord}_N(a) + c} \mod N = a^c \mod N;$$
thus by the definition of the order we must have $c = 0$. Therefore, $\mathrm{ord}_N(a) | k$.
("⇐") Suppose that $\mathrm{ord}_N(a) | k$. Then $k = q \cdot \mathrm{ord}_N(a)$ for some $q \in \mathbb{N}$. Therefore,
$$a^k \mod N = a^{q \cdot \mathrm{ord}_N(a)} \mod N = 1.$$
2. By one of previous theorems, we know that $a^{\varphi(N)} = 1 \mod N$; thus ② follows from ①.

□

§6.5 Efficiency of Shor's Algorithm

Proof.

Let $a, b, N \in \mathbb{N}$ with $\gcd(a, N) = 1 = \gcd(b, N)$.

1. ("$\Rightarrow$") Let $k \in \mathbb{N}$ satisfying $a^k = 1 \bmod N$. Then $k \geqslant \mathrm{ord}_N(a)$. Let $c = k \bmod \mathrm{ord}_N(a)$; that is, there exists $q \in \mathbb{N}$ such that $k = q \cdot \mathrm{ord}_N(a) + c$ for some $c \in \{0, 1, \cdots, \mathrm{ord}_N(a) - 1\}$. Then

   $$1 = a^k \bmod N = a^{q \cdot \mathrm{ord}_N(a) + c} \bmod N = a^c \bmod N;$$

   thus by the definition of the order we must have $c = 0$. Therefore, $\mathrm{ord}_N(a) | k$.

   ("$\Leftarrow$") Suppose that $\mathrm{ord}_N(a) | k$. Then $k = q \cdot \mathrm{ord}_N(a)$ for some $q \in \mathbb{N}$. Therefore,

   $$a^k \bmod N = a^{q \cdot \mathrm{ord}_N(a)} \bmod N = 1.$$

2. By one of previous theorems, we know that $a^{\varphi(N)} = 1 \bmod N$; thus ② follows from ①. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

3. By the rule of multiplication in $\mathbb{Z}_N^*$, we find that

$$(ab)^{\text{ord}_N(a)\text{ord}_N(b)} \bmod N = a^{\text{ord}_N(a)} b^{\text{ord}_N(b)} \bmod N = 1;$$

thus ① implies that

$$\text{ord}_N(ab) | \text{ord}_N(a)\text{ord}_N(b). \tag{5}$$

On the other hand, since $b^{\text{ord}_N(b)\text{ord}_N(ab)} = 1 \bmod N$,

$$\begin{aligned} a^{\text{ord}_N(b)\text{ord}_N(ab)} \bmod N &= a^{\text{ord}_N(b)\text{ord}_N(ab)} b^{\text{ord}_N(b)\text{ord}_N(ab)} \bmod N \\ &= (ab)^{\text{ord}_N(b)\text{ord}_N(ab)} \bmod N = 1. \end{aligned}$$

Therefore, ① shows that $\text{ord}_N(a)|\text{ord}_N(b)\text{ord}_N(ab)$. By the assumption that $\text{ord}_N(a)$ and $\text{ord}_N(b)$ are coprime, we must have $\text{ord}_N(a)|\text{ord}_N(ab)$. Similarly, we also have $\text{ord}_N(b)|\text{ord}_N(ab)$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

3. By the rule of multiplication in $\mathbb{Z}_N^*$, we find that

$$(ab)^{\mathrm{ord}_N(a)\mathrm{ord}_N(b)} \bmod N = a^{\mathrm{ord}_N(a)} b^{\mathrm{ord}_N(b)} \bmod N = 1;$$

thus ① implies that

$$\mathrm{ord}_N(ab) | \mathrm{ord}_N(a)\mathrm{ord}_N(b). \tag{5}$$

On the other hand, since $b^{\mathrm{ord}_N(b)\mathrm{ord}_N(ab)} = 1 \bmod N$,

$$\begin{aligned} a^{\mathrm{ord}_N(b)\mathrm{ord}_N(ab)} \bmod N &= a^{\mathrm{ord}_N(b)\mathrm{ord}_N(ab)} b^{\mathrm{ord}_N(b)\mathrm{ord}_N(ab)} \bmod N \\ &= (ab)^{\mathrm{ord}_N(b)\mathrm{ord}_N(ab)} \bmod N = 1. \end{aligned}$$

Therefore, ① shows that $\mathrm{ord}_N(a)|\mathrm{ord}_N(b)\mathrm{ord}_N(ab)$. By the assumption that $\mathrm{ord}_N(a)$ and $\mathrm{ord}_N(b)$ are coprime, we must have $\mathrm{ord}_N(a)|\mathrm{ord}_N(ab)$. Similarly, we also have $\mathrm{ord}_N(b)|\mathrm{ord}_N(ab)$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

3. By the rule of multiplication in $\mathbb{Z}_N^*$, we find that

$$(ab)^{\text{ord}_N(a)\text{ord}_N(b)} \mod N = a^{\text{ord}_N(a)} b^{\text{ord}_N(b)} \mod N = 1;$$

thus ① implies that

$$\text{ord}_N(ab) | \text{ord}_N(a)\text{ord}_N(b). \tag{5}$$

On the other hand, since $b^{\text{ord}_N(b)\text{ord}_N(ab)} = 1 \mod N$,

$$a^{\text{ord}_N(b)\text{ord}_N(ab)} \mod N = a^{\text{ord}_N(b)\text{ord}_N(ab)} b^{\text{ord}_N(b)\text{ord}_N(ab)} \mod N$$
$$= (ab)^{\text{ord}_N(b)\text{ord}_N(ab)} \mod N = 1.$$

Therefore, ① shows that $\text{ord}_N(a)|\text{ord}_N(b)\text{ord}_N(ab)$. By the assumption that $\text{ord}_N(a)$ and $\text{ord}_N(b)$ are coprime, we must have $\text{ord}_N(a)|\text{ord}_N(ab)$. Similarly, we also have $\text{ord}_N(b)|\text{ord}_N(ab)$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Therefore,

$$\mathrm{ord}_N(a)\mathrm{ord}_N(b)\,|\,\mathrm{ord}_N(ab)$$

which, together with (5), shows that $\mathrm{ord}_N(a)\mathrm{ord}_N(b) = \mathrm{ord}_N(ab)$.

4. Suppose that $\mathrm{ord}_N(a) = \varphi(N)$.
   - (a) First we note that the fact that $(\mathbb{Z}_N^*, \odot)$ is a group implies that $\{a^j \bmod N \,|\, 1 \leqslant j \leqslant \varphi(N)\} \subseteq \mathbb{Z}_N^*$. It then suffices to show that

$$\#\{a^j \bmod N \,|\, 1 \leqslant j \leqslant \varphi(N)\} = \varphi(N)\,. \tag{6}$$

Let $i, j \in \mathbb{N}$ with $1 \leqslant i \leqslant j \leqslant \varphi(N)$, and suppose that $a^i = a^j \bmod N$. Then $a^{j-i} = 1 \bmod N$. Therefore, ① shows that $\mathrm{ord}_N(a)\,|\,(j-i)$. Since $\mathrm{ord}_N(a) = \varphi(N)$ and $1 \leqslant i \leqslant j \leqslant \varphi(N)$, we must have $i = j$; thus (6) holds. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Therefore,

$$\text{ord}_N(a)\text{ord}_N(b)\,|\,\text{ord}_N(ab)$$

which, together with (5), shows that $\text{ord}_N(a)\text{ord}_N(b) = \text{ord}_N(ab)$.

4. Suppose that $\text{ord}_N(a) = \varphi(N)$.

   ⓐ First we note that the fact that $(\mathbb{Z}_N^*, \odot)$ is a group implies that $\{a^j \bmod N \,|\, 1 \leqslant j \leqslant \varphi(N)\} \subseteq \mathbb{Z}_N^*$. It then suffices to show that

   $$\#\{a^j \bmod N \,|\, 1 \leqslant j \leqslant \varphi(N)\} = \varphi(N)\,. \tag{6}$$

   Let $i, j \in \mathbb{N}$ with $1 \leqslant i \leqslant j \leqslant \varphi(N)$, and suppose that $a^i = a^j \bmod N$. Then $a^{j-i} = 1 \bmod N$. Therefore, ① shows that $\text{ord}_N(a)\,|\,(j-i)$. Since $\text{ord}_N(a) = \varphi(N)$ and $1 \leqslant i \leqslant j \leqslant \varphi(N)$, we must have $i = j$; thus (6) holds. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Therefore,

$$\mathrm{ord}_N(a)\mathrm{ord}_N(b)\,\big|\,\mathrm{ord}_N(ab)$$

which, together with (5), shows that $\mathrm{ord}_N(a)\mathrm{ord}_N(b)=\mathrm{ord}_N(ab)$.

4. Suppose that $\mathrm{ord}_N(a)=\varphi(N)$.
   - ⓐ First we note that the fact that $(\mathbb{Z}_N^*,\odot)$ is a group implies that $\big\{a^j \bmod N \,\big|\, 1\leqslant j\leqslant\varphi(N)\big\}\subseteq\mathbb{Z}_N^*$. It then suffices to show that

     $$\#\big\{a^j \bmod N \,\big|\, 1\leqslant j\leqslant\varphi(N)\big\}=\varphi(N)\,. \tag{6}$$

     Let $i,j\in\mathbb{N}$ with $1\leqslant i\leqslant j\leqslant\varphi(N)$, and suppose that $a^i=a^j \bmod N$. Then $a^{j-i}=1 \bmod N$. Therefore, ① shows that $\mathrm{ord}_N(a)\,\big|\,(j-i)$. Since $\mathrm{ord}_N(a)=\varphi(N)$ and $1\leqslant i\leqslant j\leqslant\varphi(N)$, we must have $i=j$; thus (6) holds. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

ⓑ **Goal**: $\mathrm{ord}_N(b) = \mathrm{ord}_N(a^j) = \dfrac{\varphi(N)}{\gcd(j, \varphi(N))}$ if $\begin{cases} \mathrm{ord}_N(a) = \varphi(N) \\ b = a^j \bmod N \end{cases}$ (4)

We first establish the first "=" of (4); that is, if $b = a^j \bmod N$, then $\mathrm{ord}_N(b) = \mathrm{ord}_N(a^j)$. To see that, we note that the identity

$$1 = b^{\mathrm{ord}_N(b)} \bmod N = (a^j \bmod N)^{\mathrm{ord}_N(b)} \bmod N$$
$$= (a^j)^{\mathrm{ord}_N(b)} \bmod N$$

shows that $\mathrm{ord}_N(a^j) \leqslant \mathrm{ord}_N(b)$, while the identity

$$1 = (a^j)^{\mathrm{ord}_N(a^j)} \bmod N = (a^j \bmod N)^{\mathrm{ord}_N(a^j)} \bmod N$$
$$= b^{\mathrm{ord}_N(a^j)} \bmod N$$

shows that $\mathrm{ord}_N(b) \leqslant \mathrm{ord}_N(a^j)$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

ⓑ **Goal**: $\text{ord}_N(b) = \text{ord}_N(a^j) = \dfrac{\varphi(N)}{\gcd(j, \varphi(N))}$ if $\begin{cases} \text{ord}_N(a) = \varphi(N) \\ b = a^j \bmod N \end{cases}$ (4)

We first establish the first "$=$" of (4); that is, if $b = a^j \bmod N$, then $\text{ord}_N(b) = \text{ord}_N(a^j)$. To see that, we note that the identity

$$\begin{aligned} 1 &= b^{\text{ord}_N(b)} \bmod N = (a^j \bmod N)^{\text{ord}_N(b)} \bmod N \\ &= (a^j)^{\text{ord}_N(b)} \bmod N \end{aligned}$$

shows that $\text{ord}_N(a^j) \leqslant \text{ord}_N(b)$, while the identity

$$\begin{aligned} 1 &= (a^j)^{\text{ord}_N(a^j)} \bmod N = (a^j \bmod N)^{\text{ord}_N(a^j)} \bmod N \\ &= b^{\text{ord}_N(a^j)} \bmod N \end{aligned}$$

shows that $\text{ord}_N(b) \leqslant \text{ord}_N(a^j)$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

ⓑ **Goal**: $\mathrm{ord}_N(b) = \mathrm{ord}_N(a^j) = \dfrac{\varphi(N)}{\gcd(j, \varphi(N))}$ if $\begin{cases} \mathrm{ord}_N(a) = \varphi(N) \\ b = a^j \bmod N \end{cases}$ (4)

We first establish the first "$=$" of (4); that is, if $b = a^j \bmod N$, then $\mathrm{ord}_N(b) = \mathrm{ord}_N(a^j)$. To see that, we note that the identity

$$\begin{aligned} 1 &= b^{\mathrm{ord}_N(b)} \bmod N = (a^j \bmod N)^{\mathrm{ord}_N(b)} \bmod N \\ &= (a^j)^{\mathrm{ord}_N(b)} \bmod N \end{aligned}$$

shows that $\mathrm{ord}_N(a^j) \leqslant \mathrm{ord}_N(b)$, while the identity

$$\begin{aligned} 1 &= (a^j)^{\mathrm{ord}_N(a^j)} \bmod N = (a^j \bmod N)^{\mathrm{ord}_N(a^j)} \bmod N \\ &= b^{\mathrm{ord}_N(a^j)} \bmod N \end{aligned}$$

shows that $\mathrm{ord}_N(b) \leqslant \mathrm{ord}_N(a^j)$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

ⓑ **Goal**: $\text{ord}_N(b) = \text{ord}_N(a^j) = \dfrac{\varphi(N)}{\gcd(j, \varphi(N))}$ if $\begin{cases} \text{ord}_N(a) = \varphi(N) \\ b = a^j \bmod N \end{cases}$ (4)

We first establish the first "=" of (4); that is, if $b = a^j \bmod N$, then $\text{ord}_N(b) = \text{ord}_N(a^j)$. To see that, we note that the identity

$$1 = b^{\text{ord}_N(b)} \bmod N = (a^j \bmod N)^{\text{ord}_N(b)} \bmod N$$
$$= (a^j)^{\text{ord}_N(b)} \bmod N$$

shows that $\text{ord}_N(a^j) \leqslant \text{ord}_N(b)$, while the identity

$$1 = (a^j)^{\text{ord}_N(a^j)} \bmod N = (a^j \bmod N)^{\text{ord}_N(a^j)} \bmod N$$
$$= b^{\text{ord}_N(a^j)} \bmod N$$

shows that $\text{ord}_N(b) \leqslant \text{ord}_N(a^j)$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

ⓑ **Goal**: $\text{ord}_N(b) = \text{ord}_N(a^j) = \dfrac{\varphi(N)}{\gcd(j, \varphi(N))}$ if $\begin{cases} \text{ord}_N(a) = \varphi(N) \\ b = a^j \bmod N \end{cases}$ (4)

We next establish the second "=" of (4). We note that ② implies that there exists $m_1 \in \mathbb{N}$ such that $m_1 \cdot \text{ord}_N(a^j) = \varphi(N)$; thus it suffices to show that $m_1 = \gcd(j, \varphi(N))$.

We remark that m_1 must satisfy $m_1 | \varphi(N)$. Moreover, since

$$1 = (a^j)^{\text{ord}_N(a^j)} \bmod N = a^{j \cdot \text{ord}_N(a^j)} \bmod N,$$

we have $\text{ord}_N(a) | j \cdot \text{ord}_N(a^j)$. By the assumption that $\text{ord}_N(a) = \varphi(N)$, there is $m_2 \in \mathbb{N}$ such that $m_2 \cdot \varphi(N) = j \cdot \text{ord}_N(a^j)$. Therefore, $j = m_1 m_2$. In particular, $m_1 | j$; thus $m_1 | \gcd(j, \varphi(N))$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

ⓑ **Goal**: $\text{ord}_N(b) = \text{ord}_N(a^j) = \dfrac{\varphi(N)}{\gcd(j, \varphi(N))}$ if $\begin{cases} \text{ord}_N(a) = \varphi(N) \\ b = a^j \bmod N \end{cases}$ (4)

We next establish the second "$=$" of (4). We note that ② implies that there exists $m_1 \in \mathbb{N}$ such that $m_1 \cdot \text{ord}_N(a^j) = \varphi(N)$; thus it suffices to show that $m_1 = \gcd(j, \varphi(N))$.

We remark that m_1 must satisfy $m_1 | \varphi(N)$. Moreover, since

$$1 = (a^j)^{\text{ord}_N(a^j)} \bmod N = a^{j \cdot \text{ord}_N(a^j)} \bmod N,$$

we have $\text{ord}_N(a) \,|\, j \cdot \text{ord}_N(a^j)$. By the assumption that $\text{ord}_N(a) = \varphi(N)$, there is $m_2 \in \mathbb{N}$ such that $m_2 \cdot \varphi(N) = j \cdot \text{ord}_N(a^j)$. Therefore, $j = m_1 m_2$. In particular, $m_1 \,|\, j$; thus $m_1 | \gcd(j, \varphi(N))$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

ⓑ **Goal**: $\text{ord}_N(b) = \text{ord}_N(a^j) = \dfrac{\varphi(N)}{\gcd(j, \varphi(N))}$ if $\begin{cases} \text{ord}_N(a) = \varphi(N) \\ b = a^j \bmod N \end{cases}$ (4)

We next establish the second "$=$" of (4). We note that ② implies that there exists $m_1 \in \mathbb{N}$ such that $m_1 \cdot \text{ord}_N(a^j) = \varphi(N)$; thus it suffices to show that $m_1 = \gcd(j, \varphi(N))$.

We remark that m_1 must satisfy $m_1 | \varphi(N)$. Moreover, since

$$1 = (a^j)^{\text{ord}_N(a^j)} \bmod N = a^{j \cdot \text{ord}_N(a^j)} \bmod N,$$

we have $\text{ord}_N(a) | j \cdot \text{ord}_N(a^j)$. By the assumption that $\text{ord}_N(a) = \varphi(N)$, there is $m_2 \in \mathbb{N}$ such that $m_2 \cdot \varphi(N) = j \cdot \text{ord}_N(a^j)$. Therefore, $j = m_1 m_2$. In particular, $m_1 | j$; thus $m_1 | \gcd(j, \varphi(N))$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

ⓑ **Goal**: $\mathrm{ord}_N(b) = \mathrm{ord}_N(a^j) = \dfrac{\varphi(N)}{\gcd(j,\varphi(N))}$ if $\begin{cases} \mathrm{ord}_N(a) = \varphi(N) \\ b = a^j \bmod N \end{cases}$ (4)

Suppose the contrary that $m_1 < \widehat{m} \equiv \gcd(j, \varphi(N))$. Then

$$\widehat{r} \equiv \frac{\varphi(N)}{\widehat{m}} < \frac{\varphi(N)}{m_1} = \mathrm{ord}_N(a^j)\,. \tag{7}$$

On the other hand, the fact that $\widehat{m} \,|\, j$ shows that

$$(a^j)^{\widehat{r}} \bmod N = (a^j)^{\varphi(N)/\widehat{m}} \bmod N = \left(a^{\varphi(N)}\right)^{j/\widehat{m}} \bmod N$$
$$= \left(a^{\varphi(N)} \bmod N\right)^{j/\widehat{m}} \bmod N = 1\,.$$

Thus, we conclude from ① that $\mathrm{ord}_N(a^j) \,|\, \widehat{r}$, a contradiction to (7). □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

ⓑ **Goal**: $\text{ord}_N(b) = \text{ord}_N(a^j) = \dfrac{\varphi(N)}{\gcd(j, \varphi(N))}$ if $\begin{cases} \text{ord}_N(a) = \varphi(N) \\ b = a^j \bmod N \end{cases}$ (4)

Suppose the contrary that $m_1 < \widehat{m} \equiv \gcd(j, \varphi(N))$. Then

$$\widehat{r} \equiv \frac{\varphi(N)}{\widehat{m}} < \frac{\varphi(N)}{m_1} = \text{ord}_N(a^j)\,. \tag{7}$$

On the other hand, the fact that $\widehat{m} \mid j$ shows that

$$(a^j)^{\widehat{r}} \bmod N = (a^j)^{\varphi(N)/\widehat{m}} \bmod N = \left(a^{\varphi(N)}\right)^{j/\widehat{m}} \bmod N$$
$$= \left(a^{\varphi(N)} \bmod N\right)^{j/\widehat{m}} \bmod N = 1\,.$$

Thus, we conclude from ① that $\text{ord}_N(a^j) \mid \widehat{r}$, a contradiction to (7). □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

ⓑ **Goal**: $\text{ord}_N(b) = \text{ord}_N(a^j) = \dfrac{\varphi(N)}{\gcd(j,\varphi(N))}$ if $\begin{cases} \text{ord}_N(a) = \varphi(N) \\ b = a^j \bmod N \end{cases}$ (4)

Suppose the contrary that $m_1 < \widehat{m} \equiv \gcd(j,\varphi(N))$. Then

$$\widehat{r} \equiv \frac{\varphi(N)}{\widehat{m}} < \frac{\varphi(N)}{m_1} = \text{ord}_N(a^j)\,. \tag{7}$$

On the other hand, the fact that $\widehat{m} \,|\, j$ shows that

$$\begin{aligned}(a^j)^{\widehat{r}} \bmod N &= (a^j)^{\varphi(N)/\widehat{m}} \bmod N = \big(a^{\varphi(N)}\big)^{j/\widehat{m}} \bmod N \\ &= \big(a^{\varphi(N)} \bmod N\big)^{j/\widehat{m}} \bmod N = 1\,.\end{aligned}$$

Thus, we conclude from ① that $\text{ord}_N(a^j)\,|\,\widehat{r}$, a contradiction to (7). □

§6.5 Efficiency of Shor's Algorithm

Lemma

Let p be a prime, $k \in \mathbb{N} \cup \{0\}$, and $f_0, f_1, \cdots, f_k$ be integers such that $p \nmid f_k$. If f is a polynomial given by $f(x) = \sum_{j=0}^{k} f_j x^j$, then either

1. $\#\{x \in \mathbb{Z}_p^* \,|\, f(x) = 0 \text{ mod } p\} \leqslant k$

or

2. $f(x) = 0$ *mod p for all* $x \in \mathbb{Z}$ *(or* $\mathbb{Z}_p^*$*).*

Proof.

We show this by induction in the degree of the polynomial, which we start at $k = 0$: if $f(x) = f_0 \neq 0$ such that $p \nmid f_0$, then it follows that $f_0 \neq 0$ mod p, and there is no $x \in \mathbb{Z}$ with $f(x) = 0$ mod p. If $f_0 = 0$, then f is the zero-polynomial. □

§6.5 Efficiency of Shor's Algorithm

Lemma

Let p be a prime, $k \in \mathbb{N} \cup \{0\}$, and $f_0, f_1, \cdots, f_k$ be integers such that $p \nmid f_k$. If f is a polynomial given by $f(x) = \sum_{j=0}^{k} f_j x^j$, then either

1. $\#\{x \in \mathbb{Z}_p^* \mid f(x) = 0 \text{ mod } p\} \leqslant k$

or

2. $f(x) = 0$ *mod p for all $x \in \mathbb{Z}$ (or $\mathbb{Z}_p^*$).*

Proof.

We show this by induction in the degree of the polynomial, which we start at $k = 0$: if $f(x) = f_0 \neq 0$ such that $p \nmid f_0$, then it follows that $f_0 \neq 0$ mod p, and there is no $x \in \mathbb{Z}$ with $f(x) = 0$ mod p. If $f_0 = 0$, then f is the zero-polynomial. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Suppose then the claim holds for all polynomials of degree up to $k-1$ and f is a polynomial of degree k. If f has fewer than k zeros modulo p in $\mathbb{Z}_p^*$, the claim holds already. Suppose that f has at least k zeros modulo p, and $n_1, n_2, \cdots, n_k \in \mathbb{Z}_p^*$ are distinct zeros of f modulo p (there may be more zeros of f modulo p, but we randomly pick k distinct zeros). Then

$$g(x) \equiv f(x) - f_k \prod_{j=1}^{k} (x - n_j) = \sum_{\ell=0}^{k-1} g_\ell x^\ell$$

is a polynomial of degree not exceeding $k-1$. Set

$$m = \max\{\ell \in \{0, 1, \cdots, k-1\} \mid p \nmid g_\ell\},$$

and define $\widetilde{g}(x) = \sum_{\ell=0}^{m} g_\ell x^\ell$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Suppose then the claim holds for all polynomials of degree up to $k-1$ and f is a polynomial of degree k. If f has fewer than k zeros modulo p in $\mathbb{Z}_p^*$, the claim holds already. Suppose that f has at least k zeros modulo p, and $n_1, n_2, \cdots, n_k \in \mathbb{Z}_p^*$ are distinct zeros of f modulo p (there may be more zeros of f modulo p, but we randomly pick k distinct zeros). Then

$$g(x) = f(x) - f_k \prod_{j=1}^{k} (x - n_j) = \sum_{\ell=0}^{k-1} g_\ell x^\ell$$

is a polynomial of degree not exceeding $k-1$. Set

$$m = \max\left\{\ell \in \{0, 1, \cdots, k-1\} \,\middle|\, p \nmid g_\ell\right\},$$

and define $\widetilde{g}(x) = \sum_{\ell=0}^{m} g_\ell x^\ell$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Suppose then the claim holds for all polynomials of degree up to $k-1$ and f is a polynomial of degree k. If f has fewer than k zeros modulo p in $\mathbb{Z}_p^*$, the claim holds already. Suppose that f has at least k zeros modulo p, and $n_1, n_2, \cdots, n_k \in \mathbb{Z}_p^*$ are distinct zeros of f modulo p (there may be more zeros of f modulo p, but we randomly pick k distinct zeros). Then

$$g(x) \equiv f(x) - f_k \prod_{j=1}^{k} (x - n_j) = \sum_{\ell=0}^{k-1} g_\ell x^\ell$$

is a polynomial of degree not exceeding $k-1$. Set

$$m = \max\left\{\ell \in \{0, 1, \cdots, k-1\} \,\middle|\, p \nmid g_\ell\right\},$$

and define $\widetilde{g}(x) = \sum_{\ell=0}^{m} g_\ell x^\ell$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Suppose then the claim holds for all polynomials of degree up to $k-1$ and f is a polynomial of degree k. If f has fewer than k zeros modulo p in $\mathbb{Z}_p^*$, the claim holds already. Suppose that f has at least k zeros modulo p, and $n_1, n_2, \cdots, n_k \in \mathbb{Z}_p^*$ are distinct zeros of f modulo p (there may be more zeros of f modulo p, but we randomly pick k distinct zeros). Then

$$g(x) \equiv f(x) - f_k \prod_{j=1}^{k} (x - n_j) = \sum_{\ell=0}^{k-1} g_\ell x^\ell$$

is a polynomial of degree not exceeding $k-1$. Set

$$m = \max\left\{\ell \in \{0, 1, \cdots, k-1\} \,\middle|\, p \nmid g_\ell\right\},$$

and define $\widetilde{g}(x) = \sum_{\ell=0}^{m} g_\ell x^\ell$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Then for $x \in \mathbb{Z}$,

$$\tilde{g}(x) \bmod p = g(x) \bmod p\,.$$

Moreover, for $1 \leqslant j \leqslant k$ we have $g(n_j) = f(n_j) = 0 \bmod p$. Therefore, $\tilde{g}$ has at least k zeros modulo p; thus by the induction assumption $\tilde{g}$ must be the zero polynomial. This shows that g is also the zero polynomial. By the definition of g,

$$f(x) = f_k \prod_{j=1}^{k} (x - n_j) \bmod p \quad \forall\, x \in \mathbb{Z}.$$

Suppose that z is a zero of f modulo p. Then by the fact that $p \nmid f_k$, the cancellation law for $\mathbb{Z}_p$ implies that $z - n_j = 0 \bmod p$ for some $1 \leqslant j \leqslant k$. This implies that f has k distinct zeros in $\mathbb{Z}_p^*$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Then for $x \in \mathbb{Z}$,

$$\tilde{g}(x) \bmod p = g(x) \bmod p\,.$$

Moreover, for $1 \leqslant j \leqslant k$ we have $g(n_j) = f(n_j) = 0 \bmod p$. Therefore, $\tilde{g}$ has at least k zeros modulo p; thus by the induction assumption $\tilde{g}$ must be the zero polynomial. This shows that g is also the zero polynomial. By the definition of g,

$$f(x) = f_k \prod_{j=1}^{k} (x - n_j) \bmod p \quad \forall\, x \in \mathbb{Z}.$$

Suppose that z is a zero of f modulo p. Then by the fact that $p \nmid f_k$, the cancellation law for $\mathbb{Z}_p$ implies that $z - n_j = 0 \bmod p$ for some $1 \leqslant j \leqslant k$. This implies that f has k distinct zeros in $\mathbb{Z}_p^*$. □

§6.5 Efficiency of Shor's Algorithm

Lemma

*Let p be prime, d a natural number satisfying $d|(p-1)$ and let h be the polynomial $h(x) = x^d - 1$. Then there exist **exactly** d distinct numbers $n_1, n_2, \cdots, n_d$ in $\mathbb{Z}_p^*$ satisfying $h(n_j) = 0 \bmod p$.*

Proof.

Let $k \in \mathbb{N}$ be such that $p-1 = dk$. Define $f(x) = \sum_{\ell=0}^{k-1} x^{d\ell}$ and $g = hf$. Then

$$g(x) = (x^d - 1) \sum_{\ell=0}^{k-1} x^{d\ell} = x^{kd} - 1 = x^{p-1} - 1.$$

Therefore, $g(x) = 0 \bmod p$ for all $x \in \mathbb{Z}_p^*$. The cancellation law in $\mathbb{Z}_p$ further implies that

$$\text{for all } x \in \mathbb{Z}_p^*, \text{ either } h(x) = 0 \bmod p \text{ or } f(x) = 0 \bmod p. \qquad \square$$

§6.5 Efficiency of Shor's Algorithm

Lemma

*Let p be prime, d a natural number satisfying $d|(p-1)$ and let h be the polynomial $h(x) = x^d - 1$. Then there exist **exactly** d distinct numbers $n_1, n_2, \cdots, n_d$ in $\mathbb{Z}_p^*$ satisfying $h(n_j) = 0$ mod p.*

Proof.

Let $k \in \mathbb{N}$ be such that $p-1 = dk$. Define $f(x) = \sum_{\ell=0}^{k-1} x^{d\ell}$ and $g = hf$. Then

$$g(x) = (x^d - 1) \sum_{\ell=0}^{k-1} x^{d\ell} = x^{kd} - 1 = x^{p-1} - 1 .$$

Therefore, $g(x) = 0$ mod p for all $x \in \mathbb{Z}_p^*$. The cancellation law in $\mathbb{Z}_p$ further implies that

$$\text{for all } x \in \mathbb{Z}_p^*, \text{ either } h(x) = 0 \text{ mod } p \text{ or } f(x) = 0 \text{ mod } p.$$ □

§6.5 Efficiency of Shor's Algorithm

Lemma

*Let p be prime, d a natural number satisfying $d|(p-1)$ and let h be the polynomial $h(x) = x^d - 1$. Then there exist **exactly** d distinct numbers $n_1, n_2, \cdots, n_d$ in $\mathbb{Z}_p^*$ satisfying $h(n_j) = 0$ mod p.*

Proof.

Let $k \in \mathbb{N}$ be such that $p-1 = dk$. Define $f(x) = \sum_{\ell=0}^{k-1} x^{d\ell}$ and $g = hf$. Then

$$g(x) = (x^d - 1)\sum_{\ell=0}^{k-1} x^{d\ell} = x^{kd} - 1 = x^{p-1} - 1\,.$$

Therefore, $g(x) = 0$ mod p for all $x \in \mathbb{Z}_p^*$. The cancellation law in $\mathbb{Z}_p$ further implies that

for all $x \in \mathbb{Z}_p^*$, either $h(x) = 0$ mod p or $f(x) = 0$ mod p. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Since $h(p-1) = p-2 \bmod p$ and $f(1) = k$, h and f are not zero polynomials. By the fact that the leading coefficient of f and h are both 1 (and $p \nmid 1$), the previous lemma implies that the polynomial h has at most d and the polynomial f has at most $d(k-1)$ zeros modulo p in $\mathbb{Z}_p^*$. Denoting the number of zeros modulo p in $\{1, \cdots p-1\}$ of the polynomials g, h and f by N_g, N_h and N_f, we have

$$dk = N_g \leqslant N_h + N_f \leqslant d + d(k-1) = dk.$$

Therefore, exactly $d(k-1)$ elements in $\mathbb{Z}_p^*$ are zeros of f modulo p, and exactly d elements in $\mathbb{Z}_p^*$ are zeros of h modulo p. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Since $h(p-1) = p-2$ mod p and $f(1) = k$, h and f are not zero polynomials. By the fact that the leading coefficient of f and h are both 1 (and $p \nmid 1$), the previous lemma implies that the polynomial h has at most d and the polynomial f has at most $d(k-1)$ zeros modulo p in $\mathbb{Z}_p^*$. Denoting the number of zeros modulo p in $\{1, \cdots p-1\}$ of the polynomials g, h and f by N_g, N_h and N_f, we have

$$dk = N_g \leqslant N_h + N_f \leqslant d + d(k-1) = dk.$$

Therefore, exactly $d(k-1)$ elements in $\mathbb{Z}_p^*$ are zeros of f modulo p, and exactly d elements in $\mathbb{Z}_p^*$ are zeros of h modulo p. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Since $h(p-1) = p-2 \bmod p$ and $f(1) = k$, h and f are not zero polynomials. By the fact that the leading coefficient of f and h are both 1 (and $p \nmid 1$), the previous lemma implies that the polynomial h has at most d and the polynomial f has at most $d(k-1)$ zeros modulo p in $\mathbb{Z}_p^*$. Denoting the number of zeros modulo p in $\{1, \cdots p-1\}$ of the polynomials g, h and f by N_g, N_h and N_f, we have

$$dk = N_g \leqslant N_h + N_f \leqslant d + d(k-1) = dk.$$

Therefore, exactly $d(k-1)$ elements in $\mathbb{Z}_p^*$ are zeros of f modulo p, and exactly d elements in $\mathbb{Z}_p^*$ are zeros of h modulo p. □

§6.5 Efficiency of Shor's Algorithm

Theorem

For every odd prime p there exists at least one primitive root modulo p; that is, there exists $a \in \mathbb{N}$ such that $\operatorname{ord}_p(a) = \varphi(p) = p - 1$.

Proof.

For a prime factor q of $p-1$, let k_q be the unique number satisfying

$$q^{k_q} | (p-1) , \qquad q^{k_q+1} \nmid (p-1) .$$

We first prove that for each prime factor q of $p-1$ there exists $a = a_q \in \mathbb{Z}_p^*$ such that $\operatorname{ord}_p(a_q) = q^{k_q}$.

Let q be a prime factor of $p-1$. By the previous lemma the polynomial $h(x) \equiv x^{q^{k_q}} - 1$ has exactly q^{k_q} zeros modulo p in $\mathbb{Z}_p^*$. Let a_q be one of these zeros, then $a_q^{q^{k_q}} = 1 \bmod p$ so it follows that $\operatorname{ord}_p(a_q) | q^{k_q}$. □

§6.5 Efficiency of Shor's Algorithm

Theorem

For every odd prime p there exists at least one primitive root modulo p; that is, there exists $a \in \mathbb{N}$ such that $\mathrm{ord}_p(a) = \varphi(p) = p-1$.

Proof.

For a prime factor q of $p-1$, let k_q be the unique number satisfying

$$q^{k_q} | (p-1) , \qquad q^{k_q+1} \nmid (p-1) .$$

We first prove that for each prime factor q of $p-1$ there exists $a = a_q \in \mathbb{Z}_p^*$ such that $\mathrm{ord}_p(a_q) = q^{k_q}$.

Let q be a prime factor of $p-1$. By the previous lemma the polynomial $h(x) \equiv x^{q^{k_q}} - 1$ has exactly q^{k_q} zeros modulo p in $\mathbb{Z}_p^*$. Let a_q be one of these zeros, then $a_q^{q^{k_q}} = 1 \bmod p$ so it follows that $\mathrm{ord}_p(a_q) | q^{k_q}$. □

§6.5 Efficiency of Shor's Algorithm

Theorem

For every odd prime p there exists at least one primitive root modulo p; that is, there exists $a \in \mathbb{N}$ such that $\mathrm{ord}_p(a) = \varphi(p) = p - 1$.

Proof.

For a prime factor q of $p-1$, let k_q be the unique number satisfying

$$q^{k_q} | (p-1)\,, \qquad q^{k_q+1} \nmid (p-1)\,.$$

We first prove that for each prime factor q of $p-1$ there exists $a = a_q \in \mathbb{Z}_p^*$ such that $\mathrm{ord}_p(a_q) = q^{k_q}$.

Let q be a prime factor of $p-1$. By the previous lemma the polynomial $h(x) \equiv x^{q^{k_q}} - 1$ has exactly q^{k_q} zeros modulo p in $\mathbb{Z}_p^*$. Let a_q be one of these zeros, then $a_q^{q^{k_q}} = 1 \bmod p$ so it follows that $\mathrm{ord}_p(a_q) | q^{k_q}$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

If this zero a_q of h has the additional property $\text{ord}_p(a_q)|q^j$ for some $j \in \mathbb{N}$ with $j < k_q$, then $\text{ord}_p(a_q)|q^{k_q-1}$ holds. Then

$$a_q^{q^{k_q-1}} = 1 \bmod p.$$

Hence, $a_q \in \mathbb{Z}_p^*$ is a zero modulo p of the polynomial $f(x) \equiv x^{q^{k_q-1}} - 1$. By the previous lemma, there are exactly q^{k_q-1} of these. This means that of the q^{k_q} zeros a_q of h at most q^{k_q-1} such a_q satisfy in addition $\text{ord}_p(a_q)|q^j$ with $j < k_q$. Therefore, there remain $q^{k_q} - q^{k_q-1}$ zeros $a_q \in \{1, \cdots, p-1\}$ that satisfy

$$\text{ord}_p(a_q)|q^{k_q} \quad \text{and} \quad \text{ord}_p(a_q) \nmid q^j \quad \forall j < k_q. \tag{8}$$

Since q is assumed prime, we conclude that there are $q^{k_q} - q^{k_q-1}$ numbers $a_q \in \{1, 2, \cdots, p-1\}$ satisfying $q^{k_q} = \text{ord}_p(a_q)$. This establishes the first statement. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

If this zero a_q of h has the additional property $\mathrm{ord}_p(a_q)|q^j$ for some $j \in \mathbb{N}$ with $j < k_q$, then $\mathrm{ord}_p(a_q)|q^{k_q-1}$ holds. Then

$$a_q^{q^{k_q-1}} = 1 \bmod p.$$

Hence, $a_q \in \mathbb{Z}_p^*$ is a zero modulo p of the polynomial $f(x) \equiv x^{q^{k_q-1}}-1$. By the previous lemma, there are exactly q^{k_q-1} of these. This means that of the q^{k_q} zeros a_q of h at most q^{k_q-1} such a_q satisfy in addition $\mathrm{ord}_p(a_q)|q^j$ with $j < k_q$. Therefore, there remain $q^{k_q}-q^{k_q-1}$ zeros $a_q \in \{1, \cdots, p-1\}$ that satisfy

$$\mathrm{ord}_p(a_q)|q^{k_q} \quad \text{and} \quad \mathrm{ord}_p(a_q) \nmid q^j \quad \forall j < k_q. \tag{8}$$

Since q is assumed prime, we conclude that there are $q^{k_q}-q^{k_q-1}$ numbers $a_q \in \{1, 2, \cdots, p-1\}$ satisfying $q^{k_q} = \mathrm{ord}_p(a_q)$. This establishes the first statement. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

If this zero a_q of h has the additional property $\mathrm{ord}_p(a_q)|q^j$ for some $j \in \mathbb{N}$ with $j < k_q$, then $\mathrm{ord}_p(a_q)|q^{k_q-1}$ holds. Then

$$a_q^{q^{k_q-1}} = 1 \bmod p\,.$$

Hence, $a_q \in \mathbb{Z}_p^*$ is a zero modulo p of the polynomial $f(x) \equiv x^{q^{k_q-1}}-1$. By the previous lemma, there are exactly q^{k_q-1} of these. This means that of the q^{k_q} zeros a_q of h at most q^{k_q-1} such a_q satisfy in addition $\mathrm{ord}_p(a_q)|q^j$ with $j < k_q$. Therefore, there remain $q^{k_q}-q^{k_q-1}$ zeros $a_q \in \{1, \cdots, p-1\}$ that satisfy

$$\mathrm{ord}_p(a_q)|q^{k_q} \qquad \text{and} \qquad \mathrm{ord}_p(a_q) \nmid q^j \quad \forall\, j < k_q\,. \tag{8}$$

Since q is assumed prime, we conclude that there are $q^{k_q}-q^{k_q-1}$ numbers $a_q \in \{1, 2, \cdots, p-1\}$ satisfying $q^{k_q} = \mathrm{ord}_p(a_q)$. This establishes the first statement. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

For each prime factor q of $p-1$, let a_q be **one particular** number in $\{1, 2, \cdots, p-1\}$ satisfying $\text{ord}_p(a_q) = q^{k_q}$. Define

$$a = \prod_{\text{q: prime factor of } p-1} a_q \,.$$

Then a is a primitive root modulo p since

$$\text{ord}_p(a) = \prod_{\text{q: prime factor of } p-1} \text{ord}_p(a_q) \,.$$

which can be shown inductively using ③ of the first theorem in this sub-section. □

§6.5 Efficiency of Shor's Algorithm

Lemma

Let p be an odd prime and a be a primitive root modulo p satisfying

$$a^{\varphi(p)} \bmod p^2 \neq 1 .$$

Then for all $k \in \mathbb{N}$, $a^{\varphi(p^k)} \bmod p^{k+1} \neq 1$.

Proof.

We first note that if $k \in \mathbb{N}$, by the fact that $\gcd(a, p^k) = 1$ the Euler Theorem implies that

$$a^{\varphi(p^k)} \bmod p^k = 1 ;$$

thus there exists $n_k \in \mathbb{N}$ such that

$$a^{\varphi(p^k)} = 1 + n_k p^k .$$

Let $D = \left\{k \in \mathbb{N} \,\middle|\, a^{\varphi(p^k)} \bmod p^{k+1} \neq 1\right\}$. By assumption, $1 \in D$. □

§6.5 Efficiency of Shor's Algorithm

Lemma

Let p be an odd prime and a be a primitive root modulo p satisfying

$$a^{\varphi(p)} \bmod p^2 \neq 1 .$$

Then for all $k \in \mathbb{N}$, $a^{\varphi(p^k)} \bmod p^{k+1} \neq 1$.

Proof.

We first note that if $k \in \mathbb{N}$, by the fact that $\gcd(a, p^k) = 1$ the Euler Theorem implies that

$$a^{\varphi(p^k)} \bmod p^k = 1 ;$$

thus there exists $n_k \in \mathbb{N}$ such that

$$a^{\varphi(p^k)} = 1 + n_k p^k .$$

Let $D = \{k \in \mathbb{N} \,|\, a^{\varphi(p^k)} \bmod p^{k+1} \neq 1\}$. By assumption, $1 \in D$. □

§6.5 Efficiency of Shor's Algorithm

Lemma

Let p be an odd prime and a be a primitive root modulo p satisfying

$$a^{\varphi(p)} \bmod p^2 \neq 1\,.$$

Then for all $k \in \mathbb{N}$, $a^{\varphi(p^k)} \mod p^{k+1} \neq 1$.

Proof.

We first note that if $k \in \mathbb{N}$, by the fact that $\gcd(a, p^k) = 1$ the Euler Theorem implies that

$$a^{\varphi(p^k)} \text{ mod } p^k = 1\,;$$

thus there exists $n_k \in \mathbb{N}$ such that

$$a^{\varphi(p^k)} = 1 + n_k p^k\,.$$

Let $D = \left\{k \in \mathbb{N} \,\middle|\, a^{\varphi(p^k)} \text{ mod} p^{k+1} \neq 1\right\}$. By assumption, $1 \in D$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Suppose that $k \in D$. Then $a^{\varphi(p^k)} \neq 1 + mp^{k+1}$ for all $m \in \mathbb{N}$. Therefore, $p \nmid n_k$. Using the formula for the Euler function,

$$\varphi(p^{k+1}) = p^k(p-1) = p\varphi(p^k);$$

thus

$$\begin{aligned} a^{\varphi(p^{k+1})} &= a^{p\varphi(p^k)} = \left(a^{\varphi(p^k)}\right)^p = (1 + n_k p^k)^p \\ &= 1 + n_k p^{k+1} + \sum_{\ell=2}^{p} C^p_\ell n_k^\ell p^{k\ell}. \end{aligned}$$

Therefore, by the fact that $p \nmid n_k$ and $p^{k+2} | p^{k\ell}$ for all $\ell \geqslant 2$ and $k \in \mathbb{N}$, we find that

$$a^{\varphi(p^{k+1})} \bmod p^{k+2} = (1 + n_k p^{k+1}) \bmod p^{k+2} \neq 1.$$

This shows that $k+1 \in D$. By induction we conclude the lemma. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Suppose that $k \in D$. Then $a^{\varphi(p^k)} \neq 1 + mp^{k+1}$ for all $m \in \mathbb{N}$. Therefore, $p \nmid n_k$. Using the formula for the Euler function,

$$\varphi(p^{k+1}) = p^k(p-1) = p\varphi(p^k);$$

thus

$$a^{\varphi(p^{k+1})} = a^{p\varphi(p^k)} = \left(a^{\varphi(p^k)}\right)^p = (1 + n_k p^k)^p$$
$$= 1 + n_k p^{k+1} + \sum_{\ell=2}^{p} C^p_\ell n_k^\ell p^{k\ell}.$$

Therefore, by the fact that $p \nmid n_k$ and $p^{k+2} | p^{k\ell}$ for all $\ell \geqslant 2$ and $k \in \mathbb{N}$, we find that

$$a^{\varphi(p^{k+1})} \bmod p^{k+2} = (1 + n_k p^{k+1}) \bmod p^{k+2} \neq 1.$$

This shows that $k+1 \in D$. By induction we conclude the lemma. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Suppose that $k \in D$. Then $a^{\varphi(p^k)} \neq 1 + mp^{k+1}$ for all $m \in \mathbb{N}$. Therefore, $p \nmid n_k$. Using the formula for the Euler function,

$$\varphi(p^{k+1}) = p^k(p-1) = p\varphi(p^k)\,;$$

thus

$$a^{\varphi(p^{k+1})} = a^{p\varphi(p^k)} = \left(a^{\varphi(p^k)}\right)^p = (1 + n_k p^k)^p$$
$$= 1 + n_k p^{k+1} + \sum_{\ell=2}^{p} C^p_\ell n_k^\ell p^{k\ell}\,.$$

Therefore, by the fact that $p \nmid n_k$ and $p^{k+2} | p^{k\ell}$ for all $\ell \geqslant 2$ and $k \in \mathbb{N}$, we find that

$$a^{\varphi(p^{k+1})} \bmod p^{k+2} = (1 + n_k p^{k+1}) \bmod p^{k+2} \neq 1\,.$$

This shows that $k+1 \in D$. By induction we conclude the lemma. □

§6.5 Efficiency of Shor's Algorithm

Theorem

Let p be an odd prime and let a be a primitive root modulo p. Then for all $k \in \mathbb{N}$ either $\operatorname{ord}_{p^k}(a) = \varphi(p^k)$ or $\operatorname{ord}_{p^k}(a+p) = \varphi(p^k)$; that is, either a or $a+p$ is a primitive root modulo p^k.

Proof.

Let a be a primitive root modulo p.

Case 1 - $a^{\varphi(p)} \bmod p^2 \neq 1$: Let $D = \{k \in \mathbb{N} \mid \operatorname{ord}_{p^k}(a) = \varphi(p^k)\}$.

Since a is a primitive root modulo p, $1 \in D$. Suppose that $k \in D$. By the definition of the order, there exists $n \in \mathbb{N}$ such that

$$a^{\operatorname{ord}_{p^{k+1}}(a)} = 1 + np^{k+1} = 1 + npp^k .$$

Therefore, $a^{\operatorname{ord}_{p^{k+1}}(a)} \equiv 1 \bmod p^k$ and the first theorem in this sub-section implies that $\operatorname{ord}_{p^k}(a) | \operatorname{ord}_{p^{k+1}}(a)$. □

§6.5 Efficiency of Shor's Algorithm

Theorem

Let p be an odd prime and let a be a primitive root modulo p. Then for all $k \in \mathbb{N}$ either $\operatorname{ord}_{p^k}(a) = \varphi(p^k)$ or $\operatorname{ord}_{p^k}(a+p) = \varphi(p^k)$; that is, either a or $a+p$ is a primitive root modulo p^k.

Proof.

Let a be a primitive root modulo p.

Case 1 - $a^{\varphi(p)} \bmod p^2 \neq 1$: Let $D = \left\{k \in \mathbb{N} \,\middle|\, \operatorname{ord}_{p^k}(a) = \varphi(p^k)\right\}$.

Since a is a primitive root modulo p, $1 \in D$. Suppose that $k \in D$. By the definition of the order, there exists $n \in \mathbb{N}$ such that

$$a^{\operatorname{ord}_{p^{k+1}}(a)} = 1 + np^{k+1} = 1 + npp^k .$$

Therefore, $a^{\operatorname{ord}_{p^{k+1}}(a)} \equiv 1 \bmod p^k$ and the first theorem in this sub-section implies that $\operatorname{ord}_{p^k}(a) \big| \operatorname{ord}_{p^{k+1}}(a)$. □

§6.5 Efficiency of Shor's Algorithm

Theorem

Let p be an odd prime and let a be a primitive root modulo p. Then for all $k \in \mathbb{N}$ either $\operatorname{ord}_{p^k}(a) = \varphi(p^k)$ or $\operatorname{ord}_{p^k}(a+p) = \varphi(p^k)$; that is, either a or $a+p$ is a primitive root modulo p^k.

Proof.

Let a be a primitive root modulo p.

Case 1 - $a^{\varphi(p)} \bmod p^2 \neq 1$: Let $D = \left\{k \in \mathbb{N} \,\middle|\, \operatorname{ord}_{p^k}(a) = \varphi(p^k)\right\}$.

Since a is a primitive root modulo p, $1 \in D$. Suppose that $k \in D$. By the definition of the order, there exists $n \in \mathbb{N}$ such that

$$a^{\operatorname{ord}_{p^{k+1}}(a)} = 1 + np^{k+1} = 1 + npp^k \,.$$

Therefore, $a^{\operatorname{ord}_{p^{k+1}}(a)} \equiv 1 \bmod p^k$ and the first theorem in this sub-section implies that $\operatorname{ord}_{p^k}(a) \,|\, \operatorname{ord}_{p^{k+1}}(a)$. □

§6.5 Efficiency of Shor's Algorithm

Theorem

Let p be an odd prime and let a be a primitive root modulo p. Then for all $k \in \mathbb{N}$ either $\operatorname{ord}_{p^k}(a) = \varphi(p^k)$ or $\operatorname{ord}_{p^k}(a+p) = \varphi(p^k)$; that is, either a or $a+p$ is a primitive root modulo p^k.

Proof.

Let a be a primitive root modulo p.

Case 1 - $a^{\varphi(p)} \bmod p^2 \neq 1$: Let $D = \left\{k \in \mathbb{N} \,\middle|\, \operatorname{ord}_{p^k}(a) = \varphi(p^k)\right\}$.

Since a is a primitive root modulo p, $1 \in D$. Suppose that $k \in D$. By the definition of the order, there exists $n \in \mathbb{N}$ such that

$$a^{\operatorname{ord}_{p^{k+1}}(a)} = 1 + np^{k+1} = 1 + npp^k .$$

Therefore, $a^{\operatorname{ord}_{p^{k+1}}(a)} \equiv 1 \bmod p^k$ and the first theorem in this sub-section implies that $\operatorname{ord}_{p^k}(a) | \operatorname{ord}_{p^{k+1}}(a)$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

By the assumption that $k \in D$, $\text{ord}_{p^k}(a) = \varphi(p^k) = p^{k-1}(p-1)$; thus $p^{k-1}(p-1) | \text{ord}_{p^{k+1}}(a)$. This implies that there exists $n_1 \in \mathbb{N}$ such that

$$\text{ord}_{p^{k+1}}(a) = n_1 p^{k-1}(p-1)\,.$$

On the other hand, the first theorem in this sub-section implies that

$$\text{ord}_{p^{k+1}}(a) | \varphi(p^{k+1})\,;$$

thus there exists $n_2 \in \mathbb{N}$ such that

$$n_2 \cdot \text{ord}_{p^{k+1}}(a) = \varphi(p^{k+1}) = p^k(p-1)\,.$$

Therefore, $n_1 n_2 = p$ which, by the fact that p is prime, shows that $(n_1, n_2) = (1, p)$ or $(n_1, n_2) = (p, 1)$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

By the assumption that $k \in D$, $\text{ord}_{p^k}(a) = \varphi(p^k) = p^{k-1}(p-1)$; thus $p^{k-1}(p-1)|\text{ord}_{p^{k+1}}(a)$. This implies that there exists $n_1 \in \mathbb{N}$ such that

$$\text{ord}_{p^{k+1}}(a) = n_1 p^{k-1}(p-1)\,.$$

On the other hand, the first theorem in this sub-section implies that

$$\text{ord}_{p^{k+1}}(a)|\varphi(p^{k+1})\,;$$

thus there exists $n_2 \in \mathbb{N}$ such that

$$n_2 \cdot \text{ord}_{p^{k+1}}(a) = \varphi(p^{k+1}) = p^k(p-1)\,.$$

Therefore, $n_1 n_2 = p$ which, by the fact that p is prime, shows that $(n_1, n_2) = (1, p)$ or $(n_1, n_2) = (p, 1)$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

By the assumption that $k \in D$, $\text{ord}_{p^k}(a) = \varphi(p^k) = p^{k-1}(p-1)$; thus $p^{k-1}(p-1) | \text{ord}_{p^{k+1}}(a)$. This implies that there exists $n_1 \in \mathbb{N}$ such that

$$\text{ord}_{p^{k+1}}(a) = n_1 p^{k-1}(p-1)\,.$$

On the other hand, the first theorem in this sub-section implies that

$$\text{ord}_{p^{k+1}}(a) | \varphi(p^{k+1})\,;$$

thus there exists $n_2 \in \mathbb{N}$ such that

$$n_2 \cdot \text{ord}_{p^{k+1}}(a) = \varphi(p^{k+1}) = p^k(p-1)\,.$$

Therefore, $n_1 n_2 = p$ which, by the fact that p is prime, shows that $(n_1, n_2) = (1, p)$ or $(n_1, n_2) = (p, 1)$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

If $(n_1, n_2) = (1, p)$, then $\text{ord}_{p^{k+1}}(a) = p^{k-1}(p-1) = \varphi(p^k)$ which further shows that

$$a^{\varphi(p^k)} \bmod p^{k+1} = 1\,,$$

a contradiction to the previous lemma. Therefore, $(n_1, n_2) = (p, 1)$ and we then have

$$\text{ord}_{p^{k+1}}(a) = p^k(p-1) = \varphi(p^{k+1})\,.$$

This concludes that $k+1 \in D$. By induction, $D = \mathbb{N}$.

Case 2 - $a^{\varphi(p)} \bmod p^2 = 1$: First we note that in this case there exists $n_3 \in \mathbb{N}$ such that $a^{p-1} = 1 + n_3 p^2$. Let $r = \text{ord}_p(a+p)$. Then $r | \varphi(p)$ and

$$(a+p)^r \bmod p = 1\,.$$

□

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

If $(n_1, n_2) = (1, p)$, then $\text{ord}_{p^{k+1}}(a) = p^{k-1}(p-1) = \varphi(p^k)$ which further shows that

$$a^{\varphi(p^k)} \bmod p^{k+1} = 1\,,$$

a contradiction to the previous lemma. Therefore, $(n_1, n_2) = (p, 1)$ and we then have

$$\text{ord}_{p^{k+1}}(a) = p^k(p-1) = \varphi(p^{k+1})\,.$$

This concludes that $k+1 \in D$. By induction, $D = \mathbb{N}$.

Case 2 - $a^{\varphi(p)} \bmod p^2 = 1$: First we note that in this case there exists $n_3 \in \mathbb{N}$ such that $a^{p-1} = 1 + n_3 p^2$. Let $r = \text{ord}_p(a+p)$. Then $r | \varphi(p)$ and

$$(a+p)^r \bmod p = 1\,.$$

□

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

If $(n_1, n_2) = (1, p)$, then $\text{ord}_{p^{k+1}}(a) = p^{k-1}(p-1) = \varphi(p^k)$ which further shows that

$$a^{\varphi(p^k)} \bmod p^{k+1} = 1\,,$$

a contradiction to the previous lemma. Therefore, $(n_1, n_2) = (p, 1)$ and we then have

$$\text{ord}_{p^{k+1}}(a) = p^k(p-1) = \varphi(p^{k+1})\,.$$

This concludes that $k+1 \in D$. By induction, $D = \mathbb{N}$.

Case 2 - $a^{\varphi(p)} \bmod p^2 = 1$: First we note that in this case there exists $n_3 \in \mathbb{N}$ such that $a^{p-1} = 1 + n_3 p^2$. Let $r = \text{ord}_p(a+p)$. Then $r \,|\, \varphi(p)$ and

$$(a+p)^r \bmod p = 1\,.$$

□

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

By binomial expansion, $a^r \bmod p = 1$ which further implies that $\varphi(p) | r$. Therefore, $r = \varphi(p)$; thus $a + p$ is also a primitive root modulo p. Next we show that $(a+p)^{\varphi(p)} \bmod p^2 \neq 1$. To see this, by binomial expansion we have

$$\begin{aligned}(a+p)^{p-1} &= a^{p-1} + (p-1)a^{p-2}p + \sum_{\ell=2}^{p-1} C_\ell^{p-1} a^{p-\ell-1} p^\ell \\ &= 1 + n_3 p^2 - pa^{p-2} + p^2 a^{p-2} + p^2 \sum_{\ell=2}^{p-1} C_\ell^{p-1} a^{p-\ell-1} p^{\ell-2} \\ &= 1 + n_4 p^2 - pa^{p-2}.\end{aligned}$$

Since (by Fermat little theorem) $a^{p-1} \bmod p = 1$, $p \nmid a^{p-2}$; thus $(a+p)^{\varphi(p)} \bmod p^2 \neq 1$. Therefore, Case 1 shows that $\text{ord}_{p^{k+1}}(a+p) = \varphi(p^{k+1})$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

By binomial expansion, $a^r \bmod p = 1$ which further implies that $\varphi(p)|r$. Therefore, $r = \varphi(p)$; thus $a+p$ is also a primitive root modulo p. Next we show that $(a+p)^{\varphi(p)} \bmod p^2 \neq 1$. To see this, by binomial expansion we have

$$\begin{aligned}(a+p)^{p-1} &= a^{p-1} + (p-1)a^{p-2}p + \sum_{\ell=2}^{p-1} C_\ell^{p-1} a^{p-\ell-1} p^\ell \\ &= 1 + n_3 p^2 - pa^{p-2} + p^2 a^{p-2} + p^2 \sum_{\ell=2}^{p-1} C_\ell^{p-1} a^{p-\ell-1} p^{\ell-2} \\ &= 1 + n_4 p^2 - pa^{p-2}\,.\end{aligned}$$

Since (by Fermat little theorem) $a^{p-1} \bmod p = 1$, $p \nmid a^{p-2}$; thus $(a+p)^{\varphi(p)} \bmod p^2 \neq 1$. Therefore, Case 1 shows that $\text{ord}_{p^{k+1}}(a+p) = \varphi(p^{k+1})$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

By binomial expansion, $a^r \bmod p = 1$ which further implies that $\varphi(p) | r$. Therefore, $r = \varphi(p)$; thus $a+p$ is also a primitive root modulo p. Next we show that $(a+p)^{\varphi(p)} \bmod p^2 \neq 1$. To see this, by binomial expansion we have

$$\begin{aligned}(a+p)^{p-1} &= a^{p-1} + (p-1)a^{p-2}p + \sum_{\ell=2}^{p-1} C_\ell^{p-1} a^{p-\ell-1} p^\ell \\ &= 1 + n_3 p^2 - pa^{p-2} + p^2 a^{p-2} + p^2 \sum_{\ell=2}^{p-1} C_\ell^{p-1} a^{p-\ell-1} p^{\ell-2} \\ &= 1 + n_4 p^2 - pa^{p-2} .\end{aligned}$$

Since (by Fermat little theorem) $a^{p-1} \bmod p = 1$, $p \nmid a^{p-2}$; thus $(a+p)^{\varphi(p)} \bmod p^2 \neq 1$. Therefore, Case 1 shows that $\text{ord}_{p^{k+1}}(a+p) = \varphi(p^{k+1})$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

By binomial expansion, $a^r \bmod p = 1$ which further implies that $\varphi(p)|r$. Therefore, $r = \varphi(p)$; thus $a + p$ is also a primitive root modulo p. Next we show that $(a+p)^{\varphi(p)} \bmod p^2 \neq 1$. To see this, by binomial expansion we have

$$\begin{aligned}(a+p)^{p-1} &= a^{p-1} + (p-1)a^{p-2}p + \sum_{\ell=2}^{p-1} C_\ell^{p-1} a^{p-\ell-1} p^\ell \\ &= 1 + n_3 p^2 - pa^{p-2} + p^2 a^{p-2} + p^2 \sum_{\ell=2}^{p-1} C_\ell^{p-1} a^{p-\ell-1} p^{\ell-2} \\ &= 1 + n_4 p^2 - pa^{p-2}\,.\end{aligned}$$

Since (by Fermat little theorem) $a^{p-1} \bmod p = 1$, $p \nmid a^{p-2}$; thus $(a+p)^{\varphi(p)} \bmod p^2 \neq 1$. Therefore, Case 1 shows that $\mathrm{ord}_{p^{k+1}}(a+p) = \varphi(p^{k+1})$. □

§6.5 Efficiency of Shor's Algorithm

Theorem

Let $N = \prod_{j=1}^{J} n_j$ *with* $n_j \in \mathbb{N}$ *and* $\gcd(n_i, n_j) = 1$ *if* $i \neq j$. *Then* $g : \mathbb{Z}_N^* \to \mathbb{Z}_{n_1}^* \times \mathbb{Z}_{n_2}^* \times \cdots \times \mathbb{Z}_{n_J}^*$ *defined by*

$$g(a) = \big(a \bmod n_1, a \bmod n_2, \cdots, a \bmod n_J\big)$$

is a bijection.

Proof.

We first show that $g(\mathbb{Z}_N^*) \subseteq \mathbb{Z}_{n_1}^* \times \mathbb{Z}_{n_2}^* \times \cdots \times \mathbb{Z}_{n_J}^*$. For each $1 \leqslant j \leqslant J$, let $g_j(a) = a \bmod n_j$. Then $g = (g_1, \cdots, g_J)$, and $g_j(a) \in \mathbb{Z}_{n_j}^*$ for all $a \in \mathbb{Z}_N^*$. Let $a \in \mathbb{Z}_N^*$ and $j \in \{1, 2, \cdots, J\}$ be given, and $\gamma = \gcd(g_j(a), n_j)$. Then there exist $\ell, k \in \mathbb{N}$ such that $g_j(a) = \gamma\ell$ and $n_j = \gamma k$. □

§6.5 Efficiency of Shor's Algorithm

Theorem

Let $N = \prod_{j=1}^{J} n_j$ *with* $n_j \in \mathbb{N}$ *and* $\gcd(n_i, n_j) = 1$ *if* $i \neq j$. *Then* $g : \mathbb{Z}_N^* \to \mathbb{Z}_{n_1}^* \times \mathbb{Z}_{n_2}^* \times \cdots \times \mathbb{Z}_{n_J}^*$ *defined by*

$$g(a) = \big(a \bmod n_1, a \bmod n_2, \cdots, a \bmod n_J\big)$$

is a bijection.

Proof.

We first show that $g(\mathbb{Z}_N^*) \subseteq \mathbb{Z}_{n_1}^* \times \mathbb{Z}_{n_2}^* \times \cdots \times \mathbb{Z}_{n_J}^*$. For each $1 \leqslant j \leqslant J$, let $g_j(a) = a \bmod n_j$. Then $g = (g_1, \cdots, g_J)$, and $g_j(a) \in \mathbb{Z}_{n_j}^*$ for all $a \in \mathbb{Z}_N^*$. Let $a \in \mathbb{Z}_N^*$ and $j \in \{1, 2, \cdots, J\}$ be given, and $\gamma = \gcd(g_j(a), n_j)$. Then there exist $\ell, k \in \mathbb{N}$ such that $g_j(a) = \gamma\ell$ and $n_j = \gamma k$. □

§6.5 Efficiency of Shor's Algorithm

Theorem

Let $N = \prod_{j=1}^{J} n_j$ *with* $n_j \in \mathbb{N}$ *and* $\gcd(n_i, n_j) = 1$ *if* $i \neq j$. *Then* $g : \mathbb{Z}_N^* \to \mathbb{Z}_{n_1}^* \times \mathbb{Z}_{n_2}^* \times \cdots \times \mathbb{Z}_{n_J}^*$ *defined by*

$$g(a) = \big(a \bmod n_1, a \bmod n_2, \cdots, a \bmod n_J\big)$$

is a bijection.

Proof.

We first show that $g(\mathbb{Z}_N^*) \subseteq \mathbb{Z}_{n_1}^* \times \mathbb{Z}_{n_2}^* \times \cdots \times \mathbb{Z}_{n_J}^*$. For each $1 \leqslant j \leqslant J$, let $g_j(a) = a \bmod n_j$. Then $g = (g_1, \cdots, g_J)$, and $g_j(a) \in \mathbb{Z}_{n_j}^*$ for all $a \in \mathbb{Z}_N^*$. Let $a \in \mathbb{Z}_N^*$ and $j \in \{1, 2, \cdots, J\}$ be given, and $\gamma = \gcd(g_j(a), n_j)$. Then there exist $\ell, k \in \mathbb{N}$ such that $g_j(a) = \gamma\ell$ and $n_j = \gamma k$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Therefore,

$$\gamma\ell = g_j(a) = a - \left[\frac{a}{n_j}\right] n_j = a - \left[\frac{a}{n_j}\right]\gamma k\,,$$

we find that

$$\frac{a}{\gamma} = \ell + \left[\frac{a}{n_j}\right] k\,.$$

The identity above shows that $\gamma \mid a$. Moreover, $n_j \mid N$, we must have $\gamma \mid N$ as well; thus the fact that $\gcd(a, N) = 1$ implies that $\gamma = 1$. In other words, $\gcd(g_j(a), n_j) = 1$ for all $1 \leqslant j \leqslant J$, and this shows that $g_j(a) \in \mathbb{Z}_{n_j}^*$ for all $1 \leqslant j \leqslant J$; thus $g(\mathbb{Z}_N^*) \subseteq \mathbb{Z}_{n_1}^* \times \mathbb{Z}_{n_2}^* \times \cdots \times \mathbb{Z}_{n_J}^*$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Therefore,

$$\gamma\ell = g_j(a) = a - \left[\frac{a}{n_j}\right] n_j = a - \left[\frac{a}{n_j}\right]\gamma k,$$

we find that

$$\frac{a}{\gamma} = \ell + \left[\frac{a}{n_j}\right] k.$$

The identity above shows that $\gamma \,|\, a$. Moreover, $n_j \,|\, N$, we must have $\gamma \,|\, N$ as well; thus the fact that $\gcd(a, N) = 1$ implies that $\gamma = 1$. In other words, $\gcd(g_j(a), n_j) = 1$ for all $1 \leqslant j \leqslant J$, and this shows that $g_j(a) \in \mathbb{Z}^*_{n_j}$ for all $1 \leqslant j \leqslant J$; thus $g(\mathbb{Z}^*_N) \subseteq \mathbb{Z}^*_{n_1} \times \mathbb{Z}^*_{n_2} \times \cdots \times \mathbb{Z}^*_{n_J}$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Therefore,

$$\gamma\ell = g_j(a) = a - \left[\frac{a}{n_j}\right] n_j = a - \left[\frac{a}{n_j}\right]\gamma k\,,$$

we find that

$$\frac{a}{\gamma} = \ell + \left[\frac{a}{n_j}\right] k\,.$$

The identity above shows that $\gamma \,|\, a$. Moreover, $n_j \,|\, N$, we must have $\gamma \,|\, N$ as well; thus the fact that $\gcd(a, N) = 1$ implies that $\gamma = 1$. In other words, $\gcd(g_j(a), n_j) = 1$ for all $1 \leqslant j \leqslant J$, and this shows that $g_j(a) \in \mathbb{Z}_{n_j}^*$ for all $1 \leqslant j \leqslant J$; thus $g(\mathbb{Z}_N^*) \subseteq \mathbb{Z}_{n_1}^* \times \mathbb{Z}_{n_2}^* \times \cdots \times \mathbb{Z}_{n_J}^*$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Next we show that g is injective. Suppose the contrary that there exist $a_1, a_2 \in \mathbb{Z}_N^*$, $a_1 \neq a_2$, such that $g(a_1) = g(a_2)$. W.L.O.G. we assume that $a_1 > a_2$. Then for all $1 \leqslant j \leqslant J$, $g_j(a_1) = g_j(a_2)$; thus

$$a_1 - a_2 = \left(\left[\frac{a_1}{n_j}\right] - \left[\frac{a_2}{n_j}\right]\right) n_j \qquad \forall\, 1 \leqslant j \leqslant J\,.$$

Therefore, $n_j \big| (a_1 - a_2)$ for all $1 \leqslant j \leqslant J$. Since $\gcd(n_i, n_j) = 1$ if $i \neq j$ and $N = \prod\limits_{j=1}^{J} n_j$, we find that $N \big| (a_1 - a_2)$, a contradiction. This establishes that g is injective. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Next we show that g is injective. Suppose the contrary that there exist $a_1, a_2 \in \mathbb{Z}_N^*$, $a_1 \neq a_2$, such that $g(a_1) = g(a_2)$. W.L.O.G. we assume that $a_1 > a_2$. Then for all $1 \leqslant j \leqslant J$, $g_j(a_1) = g_j(a_2)$; thus

$$a_1 - a_2 = \left(\left[\frac{a_1}{n_j}\right] - \left[\frac{a_2}{n_j}\right]\right) n_j \qquad \forall\, 1 \leqslant j \leqslant J\,.$$

Therefore, $n_j \big| (a_1 - a_2)$ for all $1 \leqslant j \leqslant J$. Since $\gcd(n_i, n_j) = 1$ if $i \neq j$ and $N = \prod_{j=1}^{J} n_j$, we find that $N \big| (a_1 - a_2)$, a contradiction. This establishes that g is injective. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Next we show that g is injective. Suppose the contrary that there exist $a_1, a_2 \in \mathbb{Z}_N^*$, $a_1 \neq a_2$, such that $g(a_1) = g(a_2)$. W.L.O.G. we assume that $a_1 > a_2$. Then for all $1 \leqslant j \leqslant J$, $g_j(a_1) = g_j(a_2)$; thus

$$a_1 - a_2 = \left(\left[\frac{a_1}{n_j}\right] - \left[\frac{a_2}{n_j}\right]\right) n_j \qquad \forall\, 1 \leqslant j \leqslant J\,.$$

Therefore, $n_j | (a_1 - a_2)$ for all $1 \leqslant j \leqslant J$. Since $\gcd(n_i, n_j) = 1$ if $i \neq j$ and $N = \prod_{j=1}^{J} n_j$, we find that $N | (a_1 - a_2)$, a contradiction. This establishes that g is injective. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Finally, we prove that g is surjective. Let $m_j = N/n_j$. Then $\gcd(m_j, n_j) = 1$; thus there exist $x_j, y_j \in \mathbb{Z}$ such that $m_j x_j + n_j y_j = 1$. For $\boldsymbol{b} = (b_1, \cdots, b_J) \in \mathbb{Z}^*_{n_1} \times \mathbb{Z}^*_{n_2} \times \cdots \times \mathbb{Z}^*_{n_J}$, define

$$h(\boldsymbol{b}) = \Big(\sum_{j=1}^{J} m_j x_j b_j \Big) \bmod N. \tag{9}$$

Such h is well-defined: if $\widetilde{x}_j$ and $\widetilde{y}_j$ also validate $m_j \widetilde{x}_j + n_j \widetilde{y}_j = 1$, then for all $1 \leqslant k \leqslant J$, by the fact that $m_j/n_k \in \mathbb{N}$ if $j \neq k$,

$$\begin{aligned} \frac{1}{n_k} \sum_{j=1}^{J} m_j (x_j - \widetilde{x}_j) b_j &= \sum_{j \neq k} \frac{m_j}{n_k} (x_j - \widetilde{x}_j) b_j + \frac{m_k}{n_k} (x_k - \widetilde{x}_k) b_k \\ &= \sum_{j \neq k} \frac{m_j}{n_k} (x_j - \widetilde{x}_j) b_j + \frac{m_k x_k - m_k \widetilde{x}_k}{n_k} b_k \end{aligned}$$

□

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Finally, we prove that g is surjective. Let $m_j = N/n_j$. Then $\gcd(m_j, n_j) = 1$; thus there exist $x_j, y_j \in \mathbb{Z}$ such that $m_j x_j + n_j y_j = 1$. For $\boldsymbol{b} = (b_1, \cdots, b_J) \in \mathbb{Z}^*_{n_1} \times \mathbb{Z}^*_{n_2} \times \cdots \times \mathbb{Z}^*_{n_J}$, define

$$h(\boldsymbol{b}) = \Big(\sum_{j=1}^{J} m_j x_j b_j\Big) \bmod N. \tag{9}$$

Such h is well-defined: if $\widetilde{x}_j$ and $\widetilde{y}_j$ also validate $m_j\widetilde{x}_j + n_j\widetilde{y}_j = 1$, then for all $1 \leqslant k \leqslant J$, by the fact that $m_j/n_k \in \mathbb{N}$ if $j \neq k$,

$$\frac{1}{n_k}\sum_{j=1}^{J} m_j(x_j - \widetilde{x}_j)b_j = \sum_{j\neq k}\frac{m_j}{n_k}(x_j - \widetilde{x}_j)b_j + \frac{m_k}{n_k}(x_k - \widetilde{x}_k)b_k$$

$$= \sum_{j\neq k}\frac{m_j}{n_k}(x_j - \widetilde{x}_j)b_j + \frac{m_k x_k - m_k\widetilde{x}_k}{n_k}b_k$$

□

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Finally, we prove that g is surjective. Let $m_j = N/n_j$. Then $\gcd(m_j, n_j) = 1$; thus there exist $x_j, y_j \in \mathbb{Z}$ such that $m_j x_j + n_j y_j = 1$. For $\boldsymbol{b} = (b_1, \cdots, b_J) \in \mathbb{Z}_{n_1}^* \times \mathbb{Z}_{n_2}^* \times \cdots \times \mathbb{Z}_{n_J}^*$, define

$$h(\boldsymbol{b}) = \Big(\sum_{j=1}^{J} m_j x_j b_j\Big) \bmod N. \tag{9}$$

Such h is well-defined: if $\widetilde{x}_j$ and $\widetilde{y}_j$ also validate $m_j\widetilde{x}_j + n_j\widetilde{y}_j = 1$, then for all $1 \leqslant k \leqslant J$, by the fact that $m_j/n_k \in \mathbb{N}$ if $j \neq k$,

$$\begin{aligned}\frac{1}{n_k}\sum_{j=1}^{J} m_j(x_j - \widetilde{x}_j)b_j &= \sum_{j\neq k}\frac{m_j}{n_k}(x_j - \widetilde{x}_j)b_j + \frac{m_k}{n_k}(x_k - \widetilde{x}_k)b_k \\ &= \sum_{j\neq k}\frac{m_j}{n_k}(x_j - \widetilde{x}_j)b_j + \frac{m_k x_k - m_k\widetilde{x}_k}{n_k}b_k\end{aligned}$$

□

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Finally, we prove that g is surjective. Let $m_j = N/n_j$. Then $\gcd(m_j, n_j) = 1$; thus there exist $x_j, y_j \in \mathbb{Z}$ such that $m_j x_j + n_j y_j = 1$. For $\boldsymbol{b} = (b_1, \cdots, b_J) \in \mathbb{Z}_{n_1}^* \times \mathbb{Z}_{n_2}^* \times \cdots \times \mathbb{Z}_{n_J}^*$, define

$$h(\boldsymbol{b}) = \Big(\sum_{j=1}^{J} m_j x_j b_j\Big) \bmod N. \tag{9}$$

Such h is well-defined: if $\tilde{x}_j$ and $\tilde{y}_j$ also validate $m_j \tilde{x}_j + n_j \tilde{y}_j = 1$, then for all $1 \leqslant k \leqslant J$, by the fact that $m_j/n_k \in \mathbb{N}$ if $j \neq k$,

$$\begin{aligned}\frac{1}{n_k}\sum_{j=1}^{J} m_j(x_j - \tilde{x}_j)b_j &= \sum_{j\neq k}\frac{m_j}{n_k}(x_j - \tilde{x}_j)b_j + \frac{(1-n_k y_k) - (1-n_k\tilde{y}_k)}{n_k}b_k \\ &= \sum_{j\neq k}\frac{m_j}{n_k}(x_j - \tilde{x}_j)b_j + \frac{m_k x_k - m_k\tilde{x}_k}{n_k}b_k \qquad \square\end{aligned}$$

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Finally, we prove that g is surjective. Let $m_j = N/n_j$. Then $\gcd(m_j, n_j) = 1$; thus there exist $x_j, y_j \in \mathbb{Z}$ such that $m_j x_j + n_j y_j = 1$. For $\boldsymbol{b} = (b_1, \cdots, b_J) \in \mathbb{Z}^*_{n_1} \times \mathbb{Z}^*_{n_2} \times \cdots \times \mathbb{Z}^*_{n_J}$, define

$$h(\boldsymbol{b}) = \Big(\sum_{j=1}^{J} m_j x_j b_j\Big) \bmod N. \tag{9}$$

Such h is well-defined: if $\tilde{x}_j$ and $\tilde{y}_j$ also validate $m_j\tilde{x}_j + n_j\tilde{y}_j = 1$, then for all $1 \leqslant k \leqslant J$, by the fact that $m_j/n_k \in \mathbb{N}$ if $j \neq k$,

$$\begin{aligned}\frac{1}{n_k}\sum_{j=1}^{J} m_j(x_j - \tilde{x}_j)b_j &= \sum_{j\neq k}\frac{m_j}{n_k}(x_j - \tilde{x}_j)b_j + \frac{(1 - n_k y_k) - (1 - n_k\tilde{y}_k)}{n_k}b_k \\ &= \sum_{j\neq k}\frac{m_j}{n_k}(x_j - \tilde{x}_j)b_j - (y_k - \tilde{y}_k)b_k \in \mathbb{Z}.\end{aligned}$$

□

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Finally, we prove that g is surjective. Let $m_j = N/n_j$. Then $\gcd(m_j, n_j) = 1$; thus there exist $x_j, y_j \in \mathbb{Z}$ such that $m_j x_j + n_j y_j = 1$. For $\boldsymbol{b} = (b_1, \cdots, b_J) \in \mathbb{Z}^*_{n_1} \times \mathbb{Z}^*_{n_2} \times \cdots \times \mathbb{Z}^*_{n_J}$, define

$$h(\boldsymbol{b}) = \Big(\sum_{j=1}^{J} m_j x_j b_j \Big) \bmod N. \tag{9}$$

Such h is well-defined: if $\widetilde{x}_j$ and $\widetilde{y}_j$ also validate $m_j\widetilde{x}_j + n_j\widetilde{y}_j = 1$, then for all $1 \leqslant k \leqslant J$, by the fact that $m_j/n_k \in \mathbb{N}$ if $j \neq k$,

$$\begin{aligned}\frac{1}{n_k} \sum_{j=1}^{J} m_j (x_j - \widetilde{x}_j) b_j &= \sum_{j \neq k} \frac{m_j}{n_k} (x_j - \widetilde{x}_j) b_j + \frac{(1 - n_k y_k) - (1 - n_k \widetilde{y}_k)}{n_k} b_k \\ &= \sum_{j \neq k} \frac{m_j}{n_k} (x_j - \widetilde{x}_j) b_j - (y_k - \widetilde{y}_k) b_k \in \mathbb{Z}\,.\end{aligned}$$

□

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

This shows that n_k is a factor of $\sum_{j=1}^{J} m_j(x_j - \widetilde{x}_j)b_j$ for all $1 \leqslant k \leqslant J$. Since $\gcd(n_i, n_j) = 1$ if $i \neq j$, we also have N is a factor of $\sum_{j=1}^{J} m_j(x_j - \widetilde{x}_j)b_j$. Therefore,

$$\Big(\sum_{j=1}^{J} m_j x_j b_j\Big) \bmod N = \Big(\sum_{j=1}^{J} m_j \widetilde{x}_j b_j\Big) \bmod N;$$

thus h given by (9) is well-defined.

Now we show that g is surjective by showing that $h(\boldsymbol{b}) \in \mathbb{Z}_N^*$ and $g(h(\boldsymbol{b})) = \boldsymbol{b}$ for all $\boldsymbol{b} \in \mathbb{Z}_{n_1}^* \times \mathbb{Z}_{n_2}^* \times \cdots \times \mathbb{Z}_{n_J}^*$. Let $\boldsymbol{b} \in \mathbb{Z}_{n_1}^* \times \mathbb{Z}_{n_2}^* \times \cdots \times \mathbb{Z}_{n_J}^*$ be given. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

This shows that n_k is a factor of $\sum_{j=1}^{J} m_j(x_j - \widetilde{x}_j)b_j$ for all $1 \leqslant k \leqslant J$. Since $\gcd(n_i, n_j) = 1$ if $i \neq j$, we also have N is a factor of $\sum_{j=1}^{J} m_j(x_j - \widetilde{x}_j)b_j$. Therefore,

$$\left(\sum_{j=1}^{J} m_j x_j b_j\right) \bmod N = \left(\sum_{j=1}^{J} m_j \widetilde{x}_j b_j\right) \bmod N;$$

thus h given by (9) is well-defined.

Now we show that g is surjective by showing that $h(\boldsymbol{b}) \in \mathbb{Z}_N^*$ and $g(h(\boldsymbol{b})) = \boldsymbol{b}$ for all $\boldsymbol{b} \in \mathbb{Z}_{n_1}^* \times \mathbb{Z}_{n_2}^* \times \cdots \times \mathbb{Z}_{n_J}^*$. Let $\boldsymbol{b} \in \mathbb{Z}_{n_1}^* \times \mathbb{Z}_{n_2}^* \times \cdots \times \mathbb{Z}_{n_J}^*$ be given. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

This shows that n_k is a factor of $\sum_{j=1}^{J} m_j(x_j - \widetilde{x}_j)b_j$ for all $1 \leqslant k \leqslant J$. Since $\gcd(n_i, n_j) = 1$ if $i \neq j$, we also have N is a factor of $\sum_{j=1}^{J} m_j(x_j - \widetilde{x}_j)b_j$. Therefore,

$$\Big(\sum_{j=1}^{J} m_j x_j b_j\Big) \bmod N = \Big(\sum_{j=1}^{J} m_j \widetilde{x}_j b_j\Big) \bmod N;$$

thus h given by (9) is well-defined.

Now we show that g is surjective by showing that $h(\boldsymbol{b}) \in \mathbb{Z}_N^*$ and $g(h(\boldsymbol{b})) = \boldsymbol{b}$ for all $\boldsymbol{b} \in \mathbb{Z}_{n_1}^* \times \mathbb{Z}_{n_2}^* \times \cdots \times \mathbb{Z}_{n_J}^*$. Let $\boldsymbol{b} \in \mathbb{Z}_{n_1}^* \times \mathbb{Z}_{n_2}^* \times \cdots \times \mathbb{Z}_{n_J}^*$ be given. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

For a fixed $k \in \{1, 2, \cdots, J\}$,

$$\frac{1}{n_k}(h(\boldsymbol{b}) - b_k) = \frac{1}{n_k}\left[\left(\sum_{j=1}^{J} m_j x_j b_j\right) \bmod N - b_k\right]$$

$$= \frac{1}{n_k}\left\{\sum_{j=1}^{J} m_j x_j b_j - \left[\frac{\sum_{j=1}^{J} m_j x_j b_j}{N}\right] N - b_k\right\}$$

$$= \sum_{j \neq k} \frac{m_j}{n_k} x_j b_j + \frac{m_k x_k - 1}{n_k} b_k - \left[\frac{\sum_{j=1}^{J} m_j x_j b_j}{N}\right] \frac{N}{n_k} \in \mathbb{Z}.$$

Therefore, for each $1 \leqslant k \leqslant J$ there exists $z_k \in \mathbb{Z}$ such that

$$h(\boldsymbol{b}) = b_k + z_k n_k. \tag{10}$$

To show that g is surjective it then suffices to show that $h(\boldsymbol{b}) \in \mathbb{Z}_N^*$ since then $g_k(h(\boldsymbol{b})) = b_k$ which establishes that $g(h(\boldsymbol{b})) = \boldsymbol{b}$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

For a fixed $k \in \{1, 2, \cdots, J\}$,

$$\frac{1}{n_k}(h(\boldsymbol{b}) - b_k) = \frac{1}{n_k}\Big[\Big(\sum_{j=1}^{J} m_j x_j b_j\Big) \bmod N - b_k\Big]$$

$$= \frac{1}{n_k}\Big\{\sum_{j=1}^{J} m_j x_j b_j - \Big\lfloor \frac{\sum_{j=1}^{J} m_j x_j b_j}{N} \Big\rfloor N - b_k\Big\}$$

$$= \sum_{j \neq k} \frac{m_j}{n_k} x_j b_j + \frac{m_k x_k - 1}{n_k} b_k - \Big\lfloor \frac{\sum_{j=1}^{J} m_j x_j b_j}{N} \Big\rfloor \frac{N}{n_k} \in \mathbb{Z}\,.$$

Therefore, for each $1 \leqslant k \leqslant J$ there exists $z_k \in \mathbb{Z}$ such that

$$h(\boldsymbol{b}) = b_k + z_k n_k\,. \tag{10}$$

To show that g is surjective it then suffices to show that $h(\boldsymbol{b}) \in \mathbb{Z}_N^*$ since then $g_k(h(\boldsymbol{b})) = b_k$ which establishes that $g(h(\boldsymbol{b})) = \boldsymbol{b}$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

For a fixed $k \in \{1, 2, \cdots, J\}$,

$$\begin{aligned}\frac{1}{n_k}(h(\boldsymbol{b}) - b_k) &= \frac{1}{n_k}\Big[\Big(\sum_{j=1}^{J} m_j x_j b_j\Big) \bmod N - b_k\Big] \\ &= \frac{1}{n_k}\Big\{\sum_{j=1}^{J} m_j x_j b_j - \Big[\frac{\sum_{j=1}^{J} m_j x_j b_j}{N}\Big] N - b_k\Big\} \\ &= \sum_{j \neq k} \frac{m_j}{n_k} x_j b_j + \frac{m_k x_k - 1}{n_k} b_k - \Big[\frac{\sum_{j=1}^{J} m_j x_j b_j}{N}\Big]\frac{N}{n_k} \in \mathbb{Z}\,.\end{aligned}$$

Therefore, for each $1 \leqslant k \leqslant J$ there exists $z_k \in \mathbb{Z}$ such that

$$h(\boldsymbol{b}) = b_k + z_k n_k\,. \tag{10}$$

To show that g is surjective it then suffices to show that $h(\boldsymbol{b}) \in \mathbb{Z}_N^*$ since then $g_k(h(\boldsymbol{b})) = b_k$ which establishes that $g(h(\boldsymbol{b})) = \boldsymbol{b}$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Nevertheless, that for each $1 \leqslant k \leqslant J$ there exists $z_k \in \mathbb{Z}$ such that

$$h(\boldsymbol{b}) = b_k + z_k n_k \tag{10}$$

implies that

$$\gcd(h(\boldsymbol{b}), n_k) = \gcd(b_k, n_k) = 1 \qquad \forall\, 1 \leqslant k \leqslant J.$$

The fact that $\gcd(n_i, n_j) = 1$ if $i \neq j$ further shows that $\gcd(h(\boldsymbol{b}), N) = 1$; thus $h(\boldsymbol{b}) \in \mathbb{Z}_N^*$ and we conclude that g is surjective. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Nevertheless, that for each $1 \leqslant k \leqslant J$ there exists $z_k \in \mathbb{Z}$ such that

$$h(\boldsymbol{b}) = b_k + z_k n_k \tag{10}$$

implies that

$$\gcd(h(\boldsymbol{b}), n_k) = \gcd(b_k, n_k) = 1 \qquad \forall\, 1 \leqslant k \leqslant J\,.$$

The fact that $\gcd(n_i, n_j) = 1$ if $i \neq j$ further shows that $\gcd(h(\boldsymbol{b}), N) = 1$; thus $h(\boldsymbol{b}) \in \mathbb{Z}_N^*$ and we conclude that g is surjective. □

§6.5 Efficiency of Shor's Algorithm

Lemma

Let p be an odd prime, $k \in \mathbb{N}$, and $s \in \mathbb{N} \cup \{0\}$. For a randomly chosen b from $\mathbb{Z}^*_{p^k}$ with equally distributed probability $1/\varphi(p^k)$, the probability of that $\operatorname{ord}_{p^k}(b)/2^s$ is an odd number is not greater than $1/2$. In other words,

$$(\forall\, p, k, s)\Big(\#\big\{b \in \mathbb{Z}^*_{p^k} \,\big|\, \operatorname{ord}_{p^k}(b) = 2^s t \text{ with an odd } t\big\} \leqslant \frac{1}{2}\varphi(p^k)\Big).$$

Proof.

Let p, k and s be given. Then $\#\mathbb{Z}^*_{p^k} = \varphi(p^k)$ and there exist uniquely determined $\mu, \nu \in \mathbb{N}$ with ν odd such that $\varphi(p^k) = p^k(p-1) = 2^\mu \nu$. It follows from previous theorems that there exists a primitive root $a \in \mathbb{N}$ for p^k and

$$\mathbb{Z}^*_{p^k} = \left\{a^j \bmod p^k \,\middle|\, j \in \{1, 2, \cdots, \varphi(p^k)\}\right\}.$$

□

§6.5 Efficiency of Shor's Algorithm

Lemma

Let p be an odd prime, $k \in \mathbb{N}$, and $s \in \mathbb{N} \cup \{0\}$. For a randomly chosen b from $\mathbb{Z}^*_{p^k}$ with equally distributed probability $1/\varphi(p^k)$, the probability of that $\mathrm{ord}_{p^k}(b)/2^s$ is an odd number is not greater than $1/2$. In other words,

$$(\forall\, p, k, s)\Big(\#\big\{b \in \mathbb{Z}^*_{p^k} \,\big|\, \mathrm{ord}_{p^k}(b) = 2^s t \text{ with an odd } t\big\} \leqslant \frac{1}{2}\varphi(p^k)\Big).$$

Proof.

Let p, k and s be given. Then $\#\mathbb{Z}^*_{p^k} = \varphi(p^k)$ and there exist uniquely determined $\mu, \nu \in \mathbb{N}$ with ν odd such that $\varphi(p^k) = p^k(p-1) = 2^\mu \nu$. It follows from previous theorems that there exists a primitive root $a \in \mathbb{N}$ for p^k and

$$\mathbb{Z}^*_{p^k} = \big\{a^j \bmod p^k \,\big|\, j \in \{1, 2, \cdots, \varphi(p^k)\}\big\}. \qquad \square$$

§6.5 Efficiency of Shor's Algorithm

Lemma

Let p be an odd prime, $k \in \mathbb{N}$, and $s \in \mathbb{N} \cup \{0\}$. For a randomly chosen b from $\mathbb{Z}_{p^k}^*$ with equally distributed probability $1/\varphi(p^k)$, the probability of that $\mathrm{ord}_{p^k}(b)/2^s$ is an odd number is not greater than $1/2$. In other words,

$$(\forall\, p,k,s)\Big(\#\big\{b \in \mathbb{Z}_{p^k}^* \,\big|\, \mathrm{ord}_{p^k}(b) = 2^s t \text{ with an odd } t\big\} \leqslant \frac{1}{2}\varphi(p^k)\Big).$$

Proof.

Let p, k and s be given. Then $\#\mathbb{Z}_{p^k}^* = \varphi(p^k)$ and there exist uniquely determined $\mu, \nu \in \mathbb{N}$ with ν odd such that $\varphi(p^k) = p^k(p-1) = 2^\mu \nu$.
It follows from previous theorems that there exists a primitive root $a \in \mathbb{N}$ for p^k and

$$\mathbb{Z}_{p^k}^* = \big\{a^j \bmod p^k \,\big|\, j \in \{1, 2, \cdots, \varphi(p^k)\}\big\}. \qquad \square$$

§6.5 Efficiency of Shor's Algorithm

Lemma

Let p be an odd prime, $k \in \mathbb{N}$, and $s \in \mathbb{N} \cup \{0\}$. For a randomly chosen b from $\mathbb{Z}_{p^k}^*$ with equally distributed probability $1/\varphi(p^k)$, the probability of that $\operatorname{ord}_{p^k}(b)/2^s$ is an odd number is not greater than $1/2$. In other words,

$$(\forall\, p, k, s)\left(\#\left\{b \in \mathbb{Z}_{p^k}^* \,\middle|\, \operatorname{ord}_{p^k}(b) = 2^s t \text{ with an odd } t\right\} \leqslant \frac{1}{2}\varphi(p^k)\right).$$

Proof.

Let p, k and s be given. Then $\#\mathbb{Z}_{p^k}^* = \varphi(p^k)$ and there exist uniquely determined $\mu, \nu \in \mathbb{N}$ with ν odd such that $\varphi(p^k) = p^k(p-1) = 2^\mu \nu$. It follows from previous theorems that there exists a primitive root $a \in \mathbb{N}$ for p^k and

$$\mathbb{Z}_{p^k}^* = \left\{a^j \bmod p^k \,\middle|\, j \in \{1, 2, \cdots, \varphi(p^k)\}\right\}.$$

□

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Hence, via the identification $b = a^j \bmod p^k$, the random selection of one of the equally distributed b in $\mathbb{Z}^*_{p^k}$ is the same as the random selection of an equally distributed $j \in \{1, \cdots, \varphi(p^k)\}$. Moreover,

$$\mathrm{ord}_{p^k}(b) = \frac{\varphi(p^k)}{\gcd(j, \varphi(p^k))}$$

which shows that $\mathrm{ord}_{p^k}(b) = 2^s t$ if and only if

$$2^s t = \frac{2^\mu \nu}{\gcd(j, 2^\mu \nu)}. \qquad (11)$$

By (11) we can deduce that the case $s > \mu$ cannot occur because in that case we would have $2|\nu$, a contradiction to the assumption of ν is odd. Therefore, for the event "$\mathrm{ord}_{p^k}(b)/2^s$ is odd" to happen, we must have $s \leqslant \mu$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Hence, via the identification $b = a^j \bmod p^k$, the random selection of one of the equally distributed b in $\mathbb{Z}^*_{p^k}$ is the same as the random selection of an equally distributed $j \in \{1, \cdots, \varphi(p^k)\}$. Moreover,

$$\operatorname{ord}_{p^k}(b) = \frac{\varphi(p^k)}{\gcd(j, \varphi(p^k))}$$

which shows that $\operatorname{ord}_{p^k}(b) = 2^s t$ if and only if

$$2^s t = \frac{2^\mu \nu}{\gcd(j, 2^\mu \nu)}. \tag{11}$$

By (11) we can deduce that the case $s > \mu$ cannot occur because in that case we would have $2|\nu$, a contradiction to the assumption of ν is odd. Therefore, for the event "$\operatorname{ord}_{p^k}(b)/2^s$ is odd" to happen, we must have $s \leqslant \mu$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Hence, via the identification $b = a^j \bmod p^k$, the random selection of one of the equally distributed b in $\mathbb{Z}^*_{p^k}$ is the same as the random selection of an equally distributed $j \in \{1, \cdots, \varphi(p^k)\}$. Moreover,

$$\text{ord}_{p^k}(b) = \frac{\varphi(p^k)}{\gcd(j, \varphi(p^k))}$$

which shows that $\text{ord}_{p^k}(b) = 2^s t$ if and only if

$$2^s t = \frac{2^\mu \nu}{\gcd(j, 2^\mu \nu)}. \tag{11}$$

By (11) we can deduce that the case $s > \mu$ cannot occur because in that case we would have $2|\nu$, a contradiction to the assumption of ν is odd. Therefore, for the event "$\text{ord}_{p^k}(b)/2^s$ is odd" to happen, we must have $s \leqslant \mu$.

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Hence, via the identification $b = a^j \bmod p^k$, the random selection of one of the equally distributed b in $\mathbb{Z}^*_{p^k}$ is the same as the random selection of an equally distributed $j \in \{1, \cdots, \varphi(p^k)\}$. Moreover,

$$\mathrm{ord}_{p^k}(b) = \frac{\varphi(p^k)}{\gcd(j, \varphi(p^k))}$$

which shows that $\mathrm{ord}_{p^k}(b) = 2^s t$ if and only if

$$2^s t = \frac{2^\mu \nu}{\gcd(j, 2^\mu \nu)}. \tag{11}$$

By (11) we can deduce that the case $s > \mu$ cannot occur because in that case we would have $2 | \nu$, a contradiction to the assumption of ν is odd. Therefore, for the event "$\mathrm{ord}_{p^k}(b)/2^s$ is odd" to happen, we must have $s \leqslant \mu$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Now consider the case $s \leqslant \mu$ (so that the event "$\text{ord}_{p^k}(b)/2^s$ is odd" could happen). Suppose that $j = 2^\omega x$ for some odd x (in the identification $b = a^j \bmod p^k$). Then

$$\gcd(j, 2^\mu \nu) = 2^{\min\{\omega,\mu\}} \prod_{p:\ \text{odd primes}} p^{\kappa_p} \tag{12}$$

with some $\kappa_p \in \mathbb{N} \cup \{0\}$. In order to have $\text{ord}_{p^k}(b) = 2^s t$, using (11) we obtain that

$$\gcd(j, 2^\mu \nu) = 2^{\mu - s}\nu/t\,. \tag{13}$$

Since ν and t are assumed odd, it follows that then ν/t has to be odd as well. It then follows from (12) and (13) that $\min\{\omega, \mu\} = \mu - s$ which shows $\omega = \mu - s$; thus j takes the form $j = 2^{\mu-s}x$ with an odd x and belong to $\{1, \cdots, \varphi(p^k)\}$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Now consider the case $s \leqslant \mu$ (so that the event "$\text{ord}_{p^k}(b)/2^s$ is odd" could happen). Suppose that $j = 2^\omega x$ for some odd x (in the identification $b = a^j \bmod p^k$). Then

$$\gcd(j, 2^\mu \nu) = 2^{\min\{\omega,\mu\}} \prod_{p:\ \text{odd primes}} p^{\kappa_p} \tag{12}$$

with some $\kappa_p \in \mathbb{N} \cup \{0\}$. In order to have $\text{ord}_{p^k}(b) = 2^s t$, using (11) we obtain that

$$\gcd(j, 2^\mu \nu) = 2^{\mu-s}\nu/t\,. \tag{13}$$

Since ν and t are assumed odd, it follows that then ν/t has to be odd as well. It then follows from (12) and (13) that $\min\{\omega,\mu\} = \mu-s$ which shows $\omega = \mu-s$; thus j takes the form $j = 2^{\mu-s}x$ with an odd x and belong to $\{1, \cdots, \varphi(p^k)\}$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Now consider the case $s \leqslant \mu$ (so that the event "$\text{ord}_{p^k}(b)/2^s$ is odd" could happen). Suppose that $j = 2^\omega x$ for some odd x (in the identification $b = a^j \bmod p^k$). Then

$$\gcd(j, 2^\mu \nu) = 2^{\min\{\omega,\mu\}} \prod_{p:\ \text{odd primes}} p^{\kappa_p} \tag{12}$$

with some $\kappa_p \in \mathbb{N} \cup \{0\}$. In order to have $\text{ord}_{p^k}(b) = 2^s t$, using (11) we obtain that

$$\gcd(j, 2^\mu \nu) = 2^{\mu - s} \nu / t\,. \tag{13}$$

Since ν and t are assumed odd, it follows that then ν/t has to be odd as well. It then follows from (12) and (13) that $\min\{\omega, \mu\} = \mu - s$ which shows $\omega = \mu - s$; thus j takes the form $j = 2^{\mu-s} x$ with an odd x and belong to $\{1, \cdots, \varphi(p^k)\}$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Since $\varphi(p^k) = 2^\mu \nu$, in the set $\{1, \cdots, \varphi(p^k)\}$ there exist $2^s\nu$ multiples of $2^{\mu-s}$, namely

$$\{2^{\mu-s} \times 1, 2^{\mu-s} \times 2, \cdots, 2^{\mu-s} \times 2^s\nu\}.$$

Of these $2^s\nu$ multiples of $2^{\mu-s}$ only half are of the form $j = 2^{\mu-s}x$ with an odd x. Therefore, when $s \leqslant u$ the fact that all j are chosen with the same probability implies that the probability of that $\mathrm{ord}_{p^k}(b)/2^s$ is an odd number is given by

$$\frac{\text{Number of possible } j \text{ of the form } j = 2^{\mu-s}x \text{ with } x \text{ odd}}{\text{Number of possible } j}$$

which, using that $s \leqslant \mu$, is not greater than $1/2$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Since $\varphi(p^k) = 2^\mu \nu$, in the set $\{1, \cdots, \varphi(p^k)\}$ there exist $2^s \nu$ multiples of $2^{\mu-s}$, namely

$$\{2^{\mu-s} \times 1, 2^{\mu-s} \times 2, \cdots, 2^{\mu-s} \times 2^s \nu\}.$$

Of these $2^s\nu$ multiples of $2^{\mu-s}$ only half are of the form $j = 2^{\mu-s}x$ with an odd x. Therefore, when $s \leqslant u$ the fact that all j are chosen with the same probability implies that the probability of that $\mathrm{ord}_{p^k}(b)/2^s$ is an odd number is given by

$$\frac{\text{Number of possible } j \text{ of the form } j = 2^{\mu-s}x \text{ with } x \text{ odd}}{\text{Number of possible } j}$$

which, using that $s \leqslant \mu$, is not greater than $1/2$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Since $\varphi(p^k) = 2^\mu \nu$, in the set $\{1, \cdots, \varphi(p^k)\}$ there exist $2^s \nu$ multiples of $2^{\mu - s}$, namely

$$\{2^{\mu-s} \times 1, 2^{\mu-s} \times 2, \cdots, 2^{\mu-s} \times 2^s \nu\}.$$

Of these $2^s \nu$ multiples of $2^{\mu-s}$ only half are of the form $j = 2^{\mu-s} x$ with an odd x. Therefore, when $s \leqslant u$ the fact that all j are chosen with the same probability implies that the probability of that $\mathrm{ord}_{p^k}(b)/2^s$ is an odd number is given by

$$\frac{\text{Number of possible } j \text{ of the form } j = 2^{\mu-s}x \text{ with } x \text{ odd}}{\#\{1, 2, \cdots, \varphi(p^k)\}}$$

which, using that $s \leqslant \mu$, is not greater than $1/2$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Since $\varphi(p^k) = 2^\mu \nu$, in the set $\{1, \cdots, \varphi(p^k)\}$ there exist $2^s\nu$ multiples of $2^{\mu-s}$, namely

$$\{2^{\mu-s} \times 1, 2^{\mu-s} \times 2, \cdots, 2^{\mu-s} \times 2^s\nu\}.$$

Of these $2^s\nu$ multiples of $2^{\mu-s}$ only half are of the form $j = 2^{\mu-s}x$ with an odd x. Therefore, when $s \leqslant u$ the fact that all j are chosen with the same probability implies that the probability of that $\mathrm{ord}_{p^k}(b)/2^s$ is an odd number is given by

$$\frac{\text{Number of possible } j \text{ of the form } j = 2^{\mu-s}x \text{ with } x \text{ odd}}{\varphi(p^k)}$$

which, using that $s \leqslant \mu$, is not greater than $1/2$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Since $\varphi(p^k) = 2^\mu \nu$, in the set $\{1, \cdots, \varphi(p^k)\}$ there exist $2^s \nu$ multiples of $2^{\mu-s}$, namely

$$\{2^{\mu-s} \times 1, 2^{\mu-s} \times 2, \cdots, 2^{\mu-s} \times 2^s \nu\}.$$

Of these $2^s \nu$ multiples of $2^{\mu-s}$ only half are of the form $j = 2^{\mu-s} x$ with an odd x. Therefore, when $s \leqslant u$ the fact that all j are chosen with the same probability implies that the probability of that $\mathrm{ord}_{p^k}(b)/2^s$ is an odd number is given by

$$\frac{\text{Number of possible } j \text{ of the form } j = 2^{\mu-s} x \text{ with } x \text{ odd}}{2^\mu \nu}$$

which, using that $s \leqslant \mu$, is not greater than $1/2$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Since $\varphi(p^k) = 2^\mu \nu$, in the set $\{1, \cdots, \varphi(p^k)\}$ there exist $2^s\nu$ multiples of $2^{\mu-s}$, namely

$$\{2^{\mu-s} \times 1, 2^{\mu-s} \times 2, \cdots, 2^{\mu-s} \times 2^s \nu\}.$$

Of these $2^s\nu$ multiples of $2^{\mu-s}$ only half are of the form $j = 2^{\mu-s}x$ with an odd x. Therefore, when $s \leqslant u$ the fact that all j are chosen with the same probability implies that the probability of that $\mathrm{ord}_{p^k}(b)/2^s$ is an odd number is given by

$$\frac{2^{s-1}\nu}{2^\mu \nu}$$

which, using that $s \leqslant \mu$, is not greater than $1/2$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Since $\varphi(p^k) = 2^\mu \nu$, in the set $\{1, \cdots, \varphi(p^k)\}$ there exist $2^s\nu$ multiples of $2^{\mu-s}$, namely

$$\{2^{\mu-s} \times 1, 2^{\mu-s} \times 2, \cdots, 2^{\mu-s} \times 2^s\nu\}.$$

Of these $2^s\nu$ multiples of $2^{\mu-s}$ only half are of the form $j = 2^{\mu-s}x$ with an odd x. Therefore, when $s \leqslant u$ the fact that all j are chosen with the same probability implies that the probability of that $\mathrm{ord}_{p^k}(b)/2^s$ is an odd number is given by

$$\frac{2^{s-1}\nu}{2^\mu \nu}$$

which, using that $s \leqslant \mu$, is not greater than $1/2$. □

§6.5 Efficiency of Shor's Algorithm

Finally, we restate and prove the main theorem in this sub-section.

Theorem

Let $N \in \mathbb{N}$ be odd with prime factorization $N = \prod_{j=1}^{J} p_j^{\nu_j}$, where $p_1, \cdots, p_J$ are distinct prime numbers. For a randomly chosen $b \in \mathbb{Z}_N^*$, the probability of that $r \equiv \mathrm{ord}_N(b)$ is even and $b^{r/2} + 1 \bmod N \neq 0$ is at least $1 - 1/2^{J-1}$.

Proof.

Since by assumption N is odd, all its prime factors $p_1, \cdots, p_J$ have to be odd as well, and we can apply the previous lemma for their powers $p_j^{\nu_j}$. We establish the theorem by showing that the probability of that "r is odd" **or** "r is even but $b^{r/2} + 1 = 0 \bmod N$" is not greater than $1/2^{J-1}$. □

§6.5 Efficiency of Shor's Algorithm

Finally, we restate and prove the main theorem in this sub-section.

Theorem

Let $N \in \mathbb{N}$ be odd with prime factorization $N = \prod_{j=1}^{J} p_j^{\nu_j}$, where $p_1, \cdots, p_J$ are distinct prime numbers. For a randomly chosen $b \in \mathbb{Z}_N^*$, the probability of that $r \equiv \mathrm{ord}_N(b)$ is even and $b^{r/2} + 1 \bmod N \neq 0$ is at least $1 - 1/2^{J-1}$.

Proof.

Since by assumption N is odd, all its prime factors $p_1, \cdots, p_J$ have to be odd as well, and we can apply the previous lemma for their powers $p_j^{\nu_j}$. We establish the theorem by showing that the probability of that "r is odd" **or** "r is even but $b^{r/2} + 1 = 0 \bmod N$" is not greater than $1/2^{J-1}$. □

§6.5 Efficiency of Shor's Algorithm

Finally, we restate and prove the main theorem in this sub-section.

Theorem

Let $N \in \mathbb{N}$ be odd with prime factorization $N = \prod_{j=1}^{J} p_j^{\nu_j}$, where p_1, $\cdots$, p_J are distinct prime numbers. For a randomly chosen $b \in \mathbb{Z}_N^*$, the probability of that $r \equiv \mathrm{ord}_N(b)$ is even and $b^{r/2} + 1 \bmod N \neq 0$ is at least $1 - 1/2^{J-1}$.

Proof.

Since by assumption N is odd, all its prime factors p_1, $\cdots$, p_J have to be odd as well, and we can apply the previous lemma for their powers $p_j^{\nu_j}$. We establish the theorem by showing that the probability of that "r is odd" **or** "r is even but $b^{r/2} + 1 = 0 \bmod N$" is not greater than $1/2^{J-1}$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

By one of the previous theorem, every $b \in \mathbb{Z}_N^*$ corresponds uniquely to a set of $b_j \in \mathbb{Z}_{n_j}^*$ with $1 \leqslant j \leqslant J$ and vice versa, where $n_j = p_j^{\nu_j}$ and $b_j \equiv b \bmod n_j$. An arbitrary selection of b is thus equivalent to an arbitrary selection of the tuple $(b_1, \cdots, b_J) \in \mathbb{Z}_{n_1}^* \times \cdots \times \mathbb{Z}_{n_J}^*$.

Suppose that $r = \text{ord}_N(b)$, $r_j = \text{ord}_{n_j}(b_j)$ and write $r = 2^s t$, $r_j = 2^{s_j} t_j$ for some odd numbers t and t_j. We first show that

$$r = \text{lcm}(r_1, r_2, \cdots, r_J), \tag{14}$$

where $\text{lcm}(r_1, r_2, \cdots, r_J)$ denotes the least common multiple of r_1, r_2, $\cdots$, r_J. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

By one of the previous theorem, every $b \in \mathbb{Z}_N^*$ corresponds uniquely to a set of $b_j \in \mathbb{Z}_{n_j}^*$ with $1 \leqslant j \leqslant J$ and vice versa, where $n_j = p_j^{\nu_j}$ and $b_j \equiv b \bmod n_j$. An arbitrary selection of b is thus equivalent to an arbitrary selection of the tuple $(b_1, \cdots, b_J) \in \mathbb{Z}_{n_1}^* \times \cdots \times \mathbb{Z}_{n_J}^*$.

Suppose that $r = \text{ord}_N(b)$, $r_j = \text{ord}_{n_j}(b_j)$ and write $r = 2^s t$, $r_j = 2^{s_j} t_j$ for some odd numbers t and t_j. We first show that

$$r = \text{lcm}(r_1, r_2, \cdots, r_J)\,, \tag{14}$$

where $\text{lcm}(r_1, r_2, \cdots, r_J)$ denotes the least common multiple of r_1, r_2, $\cdots$, r_J. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

To see this, note that for any $k \in \mathbb{N}$,

$$b_j^k \bmod p_j^{\nu_j} = \left(b \bmod p_j^{\nu_j}\right)^k \bmod p_j^{\nu_j} = b^k \bmod p_j^{\nu_j};$$

thus r_j is also the smallest natural number satisfying

$$b^{r_j} \equiv 1 \bmod p_j^{\nu_j}. \tag{15}$$

In other words, $r_j = \text{ord}_{n_j}(b)$. By the definition of r there exists $z \in \mathbb{N}$ such that

$$b^r = 1 + zN = 1 + z\prod_{j=1}^{J} p_j^{\nu_j},$$

thus $b^r \equiv 1 \bmod p_j^{\nu_j}$ for all $1 \leqslant j \leqslant J$. The first theorem in this sub-section then shows that $r_j | r$ for all $1 \leqslant j \leqslant J$ so that we have

$$\text{lcm}(r_1, r_2, \cdots, r_J) | r. \tag{16}$$

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

To see this, note that for any $k \in \mathbb{N}$,

$$b_j^k \bmod p_j^{\nu_j} = \left(b \bmod p_j^{\nu_j}\right)^k \bmod p_j^{\nu_j} = b^k \bmod p_j^{\nu_j};$$

thus r_j is also the smallest natural number satisfying

$$b^{r_j} \equiv 1 \bmod p_j^{\nu_j}. \tag{15}$$

In other words, $r_j = \mathrm{ord}_{n_j}(b)$. By the definition of r there exists $z \in \mathbb{N}$ such that

$$b^r = 1 + zN = 1 + z\prod_{j=1}^{J} p_j^{\nu_j},$$

thus $b^r \equiv 1 \bmod p_j^{\nu_j}$ for all $1 \leqslant j \leqslant J$. The first theorem in this sub-section then shows that $r_j \big| r$ for all $1 \leqslant j \leqslant J$ so that we have

$$\mathrm{lcm}(r_1, r_2, \cdots, r_J)\big| r. \tag{16}$$

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

To see this, note that for any $k \in \mathbb{N}$,

$$b_j^k \bmod p_j^{\nu_j} = \left(b \bmod p_j^{\nu_j}\right)^k \bmod p_j^{\nu_j} = b^k \bmod p_j^{\nu_j};$$

thus r_j is also the smallest natural number satisfying

$$b^{r_j} \equiv 1 \bmod p_j^{\nu_j}. \tag{15}$$

In other words, $r_j = \text{ord}_{n_j}(b)$. By the definition of r there exists $z \in \mathbb{N}$ such that

$$b^r = 1 + zN = 1 + z\prod_{j=1}^{J} p_j^{\nu_j},$$

thus $b^r \equiv 1 \bmod p_j^{\nu_j}$ for all $1 \leqslant j \leqslant J$. The first theorem in this sub-section then shows that $r_j | r$ for all $1 \leqslant j \leqslant J$ so that we have

$$\text{lcm}(r_1, r_2, \cdots, r_J) | r. \tag{16}$$

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

To see this, note that for any $k \in \mathbb{N}$,

$$b_j^k \bmod p_j^{\nu_j} = \left(b \bmod p_j^{\nu_j}\right)^k \bmod p_j^{\nu_j} = b^k \bmod p_j^{\nu_j};$$

thus r_j is also the smallest natural number satisfying

$$b^{r_j} \equiv 1 \bmod p_j^{\nu_j}. \tag{15}$$

In other words, $r_j = \mathrm{ord}_{n_j}(b)$. By the definition of r there exists $z \in \mathbb{N}$ such that

$$b^r = 1 + zN = 1 + z\prod_{j=1}^{J} p_j^{\nu_j},$$

thus $b^r \equiv 1 \bmod p_j^{\nu_j}$ for all $1 \leqslant j \leqslant J$. The first theorem in this sub-section then shows that $r_j | r$ for all $1 \leqslant j \leqslant J$ so that we have

$$\mathrm{lcm}(r_1, r_2, \cdots, r_J) | r. \tag{16}$$

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Let $L \equiv \text{lcm}(r_1, r_2, \cdots, r_J)$ and $1 \leqslant j \leqslant J$. By the first Theorem in this sub-section again L satisfies $b^L \equiv 1 \mod p_j^{\nu_j}$; thus $p_j^{\nu_j}$ is a factor of $b^L - 1$. Since $p_1, \cdots, p_J$ are distinct primes, we find that the product of all $p_j^{\nu_j}$ is also a factor of $b^L - 1$. Therefore, $b^L \equiv 1 \mod N$ which further implies that $r \,|\, L$. Together with (16), we conclude

$$r = \text{lcm}(r_1, r_2, \cdots, r_J)\,. \tag{14}$$

Next we show that

> the event "r is odd" $\vee$ "$2 \,|\, r$ but $b^{r/2} + 1 \equiv 0 \mod N$"
> corresponds to a **subset** of the set
> $\big\{(s_1, \cdots, s_J) \,\big|\, (\exists\, s \in \mathbb{N} \cup \{0\})(\forall\, 1 \leqslant j \leqslant J)(s_j = s)\big\}$.

□

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Let $L \equiv \text{lcm}(r_1, r_2, \cdots, r_J)$ and $1 \leqslant j \leqslant J$. By the first Theorem in this sub-section again L satisfies $b^L \equiv 1 \bmod p_j^{\nu_j}$; thus $p_j^{\nu_j}$ is a factor of $b^L - 1$. Since $p_1, \cdots, p_J$ are distinct primes, we find that the product of all $p_j^{\nu_j}$ is also a factor of $b^L - 1$. Therefore, $b^L \equiv 1 \bmod N$ which further implies that $r \,|\, L$. Together with (16), we conclude

$$r = \text{lcm}(r_1, r_2, \cdots, r_J)\,. \tag{14}$$

Next we show that

> the event "r is odd" $\vee$ "$2 \,|\, r$ but $b^{r/2} + 1 \equiv 0 \bmod N$" corresponds to a **subset** of the set
> $\{(s_1, \cdots, s_J) \,|\, (\exists\, s \in \mathbb{N} \cup \{0\})(\forall\, 1 \leqslant j \leqslant J)(s_j = s)\}$.

□

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Let $L \equiv \text{lcm}(r_1, r_2, \cdots, r_J)$ and $1 \leqslant j \leqslant J$. By the first Theorem in this sub-section again L satisfies $b^L \equiv 1 \bmod p_j^{\nu_j}$; thus $p_j^{\nu_j}$ is a factor of $b^L - 1$. Since $p_1, \cdots, p_J$ are distinct primes, we find that the product of all $p_j^{\nu_j}$ is also a factor of $b^L - 1$. Therefore, $b^L \equiv 1 \bmod N$ which further implies that $r \,|\, L$. Together with (16), we conclude

$$r = \text{lcm}(r_1, r_2, \cdots, r_J)\,. \tag{14}$$

Next we show that

the event "r is odd" $\vee$ "$2 \,|\, r$ but $b^{r/2} + 1 \equiv 0 \bmod N$" corresponds to a **subset** of the set

$$\big\{(s_1, \cdots, s_J) \,\big|\, (\exists\, s \in \mathbb{N} \cup \{0\})(\forall\, 1 \leqslant j \leqslant J)(s_j = s)\big\}.$$

□

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Using (14), we find that r is odd if and only if $2 \nmid r_j$ for all $1 \leqslant j \leqslant J$. Therefore,

$$r \text{ is odd if and only if } s_j = 0 \text{ for all } 1 \leqslant j \leqslant J. \tag{17}$$

Now we consider the case that r is even but $b^{r/2} + 1 \equiv 0 \mod N$. Then there exists $\ell \in \mathbb{N}$ such that $b^{r/2} + 1 = \ell N$. Letting $\ell_j = \ell N / p_j^{\nu_j}$, we have $b^{r/2} + 1 = \ell_j p_j^{\nu_j}$ for all $1 \leqslant j \leqslant J$; thus

$$b^{r/2} + 1 \equiv 0 \mod p_j^{\nu_j}. \tag{18}$$

On the other hand, note that (14) implies that $s_j \leqslant s$ for all $1 \leqslant j \leqslant J$. Suppose that $s_j < s$ for some $1 \leqslant j \leqslant J$. Then the fact that

$$2^s t = r = k_j r_j = k_j 2^{s_j} t_j$$

shows that $k_j = 2^{s-s_j} t / t_j$ is even. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Using (14), we find that r is odd if and only if $2 \nmid r_j$ for all $1 \leqslant j \leqslant J$. Therefore,

$$r \text{ is odd if and only if } s_j = 0 \text{ for all } 1 \leqslant j \leqslant J. \tag{17}$$

Now we consider the case that r is even but $b^{r/2} + 1 \equiv 0 \mod N$. Then there exists $\ell \in \mathbb{N}$ such that $b^{r/2} + 1 = \ell N$. Letting $\ell_j = \ell N / p_j^{\nu_j}$, we have $b^{r/2} + 1 = \ell_j p_j^{\nu_j}$ for all $1 \leqslant j \leqslant J$; thus

$$b^{r/2} + 1 \equiv 0 \mod p_j^{\nu_j}. \tag{18}$$

On the other hand, note that (14) implies that $s_j \leqslant s$ for all $1 \leqslant j \leqslant J$. Suppose that $s_j < s$ for some $1 \leqslant j \leqslant J$. Then the fact that

$$2^s t = r = k_j r_j = k_j 2^{s_j} t_j$$

shows that $k_j = 2^{s - s_j} t / t_j$ is even. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Using (14), we find that r is odd if and only if $2 \nmid r_j$ for all $1 \leqslant j \leqslant J$. Therefore,

$$r \text{ is odd if and only if } s_j = 0 \text{ for all } 1 \leqslant j \leqslant J. \tag{17}$$

Now we consider the case that r is even but $b^{r/2} + 1 \equiv 0 \mod N$. Then there exists $\ell \in \mathbb{N}$ such that $b^{r/2} + 1 = \ell N$. Letting $\ell_j = \ell N / p_j^{\nu_j}$, we have $b^{r/2} + 1 = \ell_j p_j^{\nu_j}$ for all $1 \leqslant j \leqslant J$; thus

$$b^{r/2} + 1 \equiv 0 \mod p_j^{\nu_j}. \tag{18}$$

On the other hand, note that (14) implies that $s_j \leqslant s$ for all $1 \leqslant j \leqslant J$. Suppose that $s_j < s$ for some $1 \leqslant j \leqslant J$. Then the fact that

$$2^s t = r = k_j r_j = k_j 2^{s_j} t_j$$

shows that $k_j = 2^{s - s_j} t / t_j$ is even. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Let $z_j = k_j/2$. Then $r/2 = z_j r_j$ with $z_j \in \mathbb{N}$; thus using (15) we find that

$$b^{r/2} \bmod p_j^{\nu_j} = b^{z_j r_j} \bmod p_j^{\nu_j} = \left(b^{r_j} \bmod p_j^{\nu_j}\right)^{z_j} \bmod p_j^{\nu_j}$$
$$= 1 \bmod p_j^{\nu_j} = 1\,,$$

a contradiction to (18). Therefore, we must have $s_j = s$ for all $1 \leqslant j \leqslant J$ if r is even but $b^r + 1 \equiv 0 \bmod N$. Together with (17), we conclude that

the event "r is odd" $\vee$ "$2 | r$ but $b^{r/2} + 1 \equiv 0 \bmod N$" corresponds to a **subset** of the set
$\left\{(s_1, \cdots, s_J) \,\middle|\, (\exists\, s \in \mathbb{N} \cup \{0\})(\forall\, 1 \leqslant j \leqslant J)(s_j = s)\right\}$.

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Let $z_j = k_j/2$. Then $r/2 = z_j r_j$ with $z_j \in \mathbb{N}$; thus using (15) we find that

$$b^{r/2} \bmod p_j^{\nu_j} = b^{z_j r_j} \bmod p_j^{\nu_j} = \left(b^{r_j} \bmod p_j^{\nu_j}\right)^{z_j} \bmod p_j^{\nu_j} = 1 \bmod p_j^{\nu_j} = 1\,,$$

a contradiction to (18). Therefore, we must have $s_j = s$ for all $1 \leqslant j \leqslant J$ if r is even but $b^r + 1 \equiv 0 \bmod N$. Together with (17), we conclude that

the event "r is odd" $\vee$ "$2 | r$ but $b^{r/2} + 1 \equiv 0 \bmod N$" corresponds to a **subset** of the set $\left\{(s_1, \cdots, s_J) \,\middle|\, (\exists\, s \in \mathbb{N} \cup \{0\})(\forall\, 1 \leqslant j \leqslant J)(s_j = s)\right\}$.

□

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Since all s_j's are chosen **independently**, the previous lemma that

$$\begin{aligned}
&\mathbf{P}(\{r \text{ is odd}\} \vee \{b^{r/2}+1 \equiv 0 \bmod N\}) \\
&\leqslant \sum_{s=0}^{\infty} \mathbf{P}(\{s_j = s \text{ for all } 1 \leqslant j \leqslant J\}) \\
&= \sum_{s=0}^{\infty} \prod_{j=1}^{J} \mathbf{P}(\{s_j = s\}) = \sum_{s=0}^{\infty} \mathbf{P}(\{s_1 = s\}) \prod_{j=2}^{J} \mathbf{P}(\{s_j = s\}) \\
&= \sum_{s=0}^{\infty} \mathbf{P}(\{s_1 = s\}) \prod_{j=2}^{J} \mathbf{P}(\{r_j = 2^s t \text{ with an odd } t\}) \\
&\leqslant \sum_{s=0}^{\infty} \mathbf{P}(\{s_1 = s\}) \frac{1}{2^{J-1}} = \frac{1}{2^{J-1}}.
\end{aligned}$$

Therefore, the probability of that $r \equiv \mathrm{ord}_N(b)$ is even and $b^{r/2}+1 \bmod N \neq 0$ is at least $1 - 1/2^{J-1}$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Since all s_j's are chosen **independently**, the previous lemma that

$$
\begin{aligned}
&\mathbf{P}(\{r \text{ is odd}\} \vee \{b^{r/2} + 1 \equiv 0 \text{ mod } N\}) \\
&\leqslant \sum_{s=0}^{\infty} \mathbf{P}(\{s_j = s \text{ for all } 1 \leqslant j \leqslant J\}) \\
&= \sum_{s=0}^{\infty} \prod_{j=1}^{J} \mathbf{P}(\{s_j = s\}) = \sum_{s=0}^{\infty} \mathbf{P}(\{s_1 = s\}) \prod_{j=2}^{J} \mathbf{P}(\{s_j = s\}) \\
&= \sum_{s=0}^{\infty} \mathbf{P}(\{s_1 = s\}) \prod_{j=2}^{J} \mathbf{P}(\{r_j = 2^s t \text{ with an odd } t\}) \\
&\leqslant \sum_{s=0}^{\infty} \mathbf{P}(\{s_1 = s\}) \frac{1}{2^{J-1}} = \frac{1}{2^{J-1}}.
\end{aligned}
$$

Therefore, the probability of that $r \equiv \text{ord}_N(b)$ is even and $b^{r/2} + 1 \text{ mod } N \neq 0$ is at least $1 - 1/2^{J-1}$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Since all s_j's are chosen **independently**, the previous lemma that

$$
\begin{aligned}
&\mathbf{P}(\{r \text{ is odd}\} \vee \{b^{r/2}+1 \equiv 0 \bmod N\}) \\
&\leqslant \sum_{s=0}^{\infty} \mathbf{P}(\{s_j = s \text{ for all } 1 \leqslant j \leqslant J\}) \\
&= \sum_{s=0}^{\infty} \prod_{j=1}^{J} \mathbf{P}(\{s_j = s\}) = \sum_{s=0}^{\infty} \mathbf{P}(\{s_1 = s\}) \prod_{j=2}^{J} \mathbf{P}(\{s_j = s\}) \\
&= \sum_{s=0}^{\infty} \mathbf{P}(\{s_1 = s\}) \prod_{j=2}^{J} \mathbf{P}(\{r_j = 2^s t \text{ with an odd } t\}) \\
&\leqslant \sum_{s=0}^{\infty} \mathbf{P}(\{s_1 = s\}) \frac{1}{2^{J-1}} = \frac{1}{2^{J-1}}.
\end{aligned}
$$

Therefore, the probability of that $r \equiv \mathrm{ord}_N(b)$ is even and $b^{r/2}+1 \bmod N \neq 0$ is at least $1 - 1/2^{J-1}$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Since all s_j's are chosen **independently**, the previous lemma that

$$\begin{aligned}
&\mathbf{P}(\{r \text{ is odd}\} \vee \{b^{r/2}+1 \equiv 0 \bmod N\}) \\
&\leqslant \sum_{s=0}^{\infty} \mathbf{P}(\{s_j = s \text{ for all } 1 \leqslant j \leqslant J\}) \\
&= \sum_{s=0}^{\infty} \prod_{j=1}^{J} \mathbf{P}(\{s_j = s\}) = \sum_{s=0}^{\infty} \mathbf{P}(\{s_1 = s\}) \prod_{j=2}^{J} \mathbf{P}(\{s_j = s\}) \\
&= \sum_{s=0}^{\infty} \mathbf{P}(\{s_1 = s\}) \prod_{j=2}^{J} \mathbf{P}\big(\{r_j = 2^s t \text{ with an odd } t\}\big) \\
&\leqslant \sum_{s=0}^{\infty} \mathbf{P}(\{s_1 = s\}) \frac{1}{2^{J-1}} = \frac{1}{2^{J-1}}.
\end{aligned}$$

Therefore, the probability of that $r \equiv \mathrm{ord}_N(b)$ is even and $b^{r/2}+1 \bmod N \neq 0$ is at least $1 - 1/2^{J-1}$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Since all s_j's are chosen **independently**, the previous lemma that

$$\begin{aligned}
&\mathbf{P}(\{r \text{ is odd}\} \vee \{b^{r/2}+1 \equiv 0 \text{ mod } N\}) \\
&\leqslant \sum_{s=0}^{\infty} \mathbf{P}(\{s_j = s \text{ for all } 1 \leqslant j \leqslant J\}) \\
&= \sum_{s=0}^{\infty} \prod_{j=1}^{J} \mathbf{P}(\{s_j = s\}) = \sum_{s=0}^{\infty} \mathbf{P}(\{s_1 = s\}) \prod_{j=2}^{J} \mathbf{P}(\{s_j = s\}) \\
&= \sum_{s=0}^{\infty} \mathbf{P}(\{s_1 = s\}) \prod_{j=2}^{J} \mathbf{P}\big(\{r_j = 2^s t \text{ with an odd } t\}\big) \\
&\leqslant \sum_{s=0}^{\infty} \mathbf{P}(\{s_1 = s\}) \frac{1}{2^{J-1}} = \frac{1}{2^{J-1}}\,.
\end{aligned}$$

Therefore, the probability of that $r \equiv \text{ord}_N(b)$ is even and $b^{r/2}+1 \text{ mod } N \neq 0$ is at least $1 - 1/2^{J-1}$. □

§6.5 Efficiency of Shor's Algorithm

Proof (cont'd).

Since all s_j's are chosen **independently**, the previous lemma that

$$\begin{aligned}
&\mathbf{P}(\{r \text{ is odd}\} \vee \{b^{r/2} + 1 \equiv 0 \bmod N\}) \\
&\leqslant \sum_{s=0}^{\infty} \mathbf{P}(\{s_j = s \text{ for all } 1 \leqslant j \leqslant J\}) \\
&= \sum_{s=0}^{\infty} \prod_{j=1}^{J} \mathbf{P}(\{s_j = s\}) = \sum_{s=0}^{\infty} \mathbf{P}(\{s_1 = s\}) \prod_{j=2}^{J} \mathbf{P}(\{s_j = s\}) \\
&= \sum_{s=0}^{\infty} \mathbf{P}(\{s_1 = s\}) \prod_{j=2}^{J} \mathbf{P}\big(\{r_j = 2^s t \text{ with an odd } t\}\big) \\
&\leqslant \sum_{s=0}^{\infty} \mathbf{P}(\{s_1 = s\}) \frac{1}{2^{J-1}} = \frac{1}{2^{J-1}}\,.
\end{aligned}$$

Therefore, the probability of that $r \equiv \mathrm{ord}_N(b)$ is even and $b^{r/2} + 1 \bmod N \neq 0$ is at least $1 - 1/2^{J-1}$. □