# QUANTUM CIRCUIS and SIMPLE QUANTUM ALGORITHMS

prof. RNDr. Jozef Gruska, DrSc.

Faculty of Informatics
Masaryk University
Brno, Czech Republic

2016

# Part I

# Seminal quantum algorithms

# SEMINAL QUANTUM ALGORITHMS

Jozef Gruska
Faculty of Informatics, Masaryk Univers.
Brno, Czech Republic

ISCQI School, IOF Bhubaneswar
February 6, 2016

# INTEGER FACTORIZATION and ITS IMPORTANCE

# INTEGER FACTORIZATION and ITS IMPORTANCE

- A **prime** is an integer that has no divisors expect 1 and itself.

# INTEGER FACTORIZATION and ITS IMPORTANCE

- A **prime** is an integer that has no divisors expect 1 and itself.
- **Integer factorization theorem:** each integer $n$ can be uniquelly expressed as a product of the powers of primes

$$n - \prod_{i=1}^{k} p_i^{e_i}.$$

where $p_1 < p_2 < \ldots < p_k$ are primes

# INTEGER FACTORIZATION and ITS IMPORTANCE

- A **prime** is an integer that has no divisors expect 1 and itself.
- **Integer factorization theorem:** each integer $n$ can be uniquelly expressed as a product of the powers of primes

$$n - \prod_{i=1}^{k} p_i^{e_i}.$$

where $p_1 < p_2 < \ldots < p_k$ are primes
- No efficient classical algorithm is known to find, in general factors $p_1, p_2, \ldots p_k$

# INTEGER FACTORIZATION and ITS IMPORTANCE

- A **prime** is an integer that has no divisors expect 1 and itself.
- **Integer factorization theorem:** each integer $n$ can be uniquelly expressed as a product of the powers of primes

$$n - \prod_{i=1}^{k} p_i^{e_i}.$$

where $p_1 < p_2 < \ldots < p_k$ are primes

- No efficient classical algorithm is known to find, in general factors $p_1, p_2, \ldots p_k$
- The fastest classical algorithm to factor an $m$-bit integer that is product of two primes has exponential complexity $\mathcal{O}(e^{cm^{1/3}(\lg m)^{2/3}})$, where $e$ is the basis of natural logarithms.

# INTEGER FACTORIZATION and ITS IMPORTANCE

- A **prime** is an integer that has no divisors expect 1 and itself.
- **Integer factorization theorem:** each integer $n$ can be uniquelly expressed as a product of the powers of primes

$$n - \prod_{i=1}^{k} p_i^{e_i}.$$

where $p_1 < p_2 < \ldots < p_k$ are primes

- No efficient classical algorithm is known to find, in general factors $p_1, p_2, \ldots p_k$
- The fastest classical algorithm to factor an $m$-bit integer that is product of two primes has exponential complexity $\mathcal{O}(e^{cm^{1/3}(\lg m)^{2/3}})$, where $e$ is the basis of natural logarithms.
- The fact that there is not known classical algorithm to factor integers is playing very important role in cryptography - for making secure encryptions and secure digital signatures.

## LARGEST PRIME

On February 3, 2016 C. Cooper from university Missouri announced a new (Mersene) prime

$$2^{74207181}$$

that has 5 millions more digits as previously known largest prime.

# DESIGN and USE of RSA CRYPTOSYSTEM

# DESIGN and USE of RSA CRYPTOSYSTEM

Invented in 1978 by Rivest, Shamir, Adleman

# DESIGN and USE of RSA CRYPTOSYSTEM

Invented in 1978 by Rivest, Shamir, Adleman

**Basic idea:** prime multiplication is very easy, integer factorization unfeasible.

# DESIGN and USE of RSA CRYPTOSYSTEM

Invented in 1978 by Rivest, Shamir, Adleman

**Basic idea:** prime multiplication is very easy, integer factorization unfeasible.

## Design of RSA cryptosystems

# DESIGN and USE of RSA CRYPTOSYSTEM

Invented in 1978 by Rivest, Shamir, Adleman

**Basic idea:** prime multiplication is very easy, integer factorization unfeasible.

## Design of RSA cryptosystems

1. Choose randomly two large about s-bit primes p,q, where $s \in [512, 1024]$, and denote

$$n = pq, \phi(n) = (p-1)(q-1)$$

# DESIGN and USE of RSA CRYPTOSYSTEM

Invented in 1978 by Rivest, Shamir, Adleman

**Basic idea:** prime multiplication is very easy, integer factorization unfeasible.

## Design of RSA cryptosystems

1. Choose randomly two large about s-bit primes p,q, where $s \in [512, 1024]$, and denote

$$n = pq, \phi(n) = (p-1)(q-1)$$

2. Choose a large d such that

$$\gcd(d, \phi(n)) = 1$$

# DESIGN and USE of RSA CRYPTOSYSTEM

Invented in 1978 by Rivest, Shamir, Adleman

**Basic idea:** prime multiplication is very easy, integer factorization unfeasible.

## Design of RSA cryptosystems

1. Choose randomly two large about s-bit primes p,q, where $s \in [512, 1024]$, and denote

$$n = pq, \phi(n) = (p-1)(q-1)$$

2. Choose a large d such that

$$\gcd(d, \phi(n)) = 1$$

and compute

$$e = d^{-1} (\text{mod } \phi(n))$$

# DESIGN and USE of RSA CRYPTOSYSTEM

Invented in 1978 by Rivest, Shamir, Adleman

**Basic idea:** prime multiplication is very easy, integer factorization unfeasible.

## Design of RSA cryptosystems

1. Choose randomly two large about s-bit primes p,q, where $s \in [512, 1024]$, and denote

$$n = pq, \phi(n) = (p-1)(q-1)$$

2. Choose a large d such that

$$\gcd(d, \phi(n)) = 1$$

and compute

$$e = d^{-1} (\text{mod } \phi(n))$$

Public key: n (modulus), e (encryption exponent)
Trapdoor information: $p, q, d$ (decryption exponent)

# DESIGN and USE of RSA CRYPTOSYSTEM

Invented in 1978 by Rivest, Shamir, Adleman

**Basic idea:** prime multiplication is very easy, integer factorization unfeasible.

## Design of RSA cryptosystems

1. Choose randomly two large about s-bit primes p,q, where $s \in [512, 1024]$, and denote

$$n = pq, \phi(n) = (p-1)(q-1)$$

2. Choose a large d such that

$$\gcd(d, \phi(n)) = 1$$

and compute

$$e = d^{-1} (\text{mod } \phi(n))$$

Public key: n (modulus), e (encryption exponent)
Trapdoor information: $p, q, d$ (decryption exponent)

**Plaintext** $w$
**Encryption:** cryptotext $c = w^e \bmod n$
**Decryption:** plaintext $w = c^d \bmod n$

# RSA CHALLENGE

The first public description of the RSA cryptosystem was in the paper.

Martin Gardner: A newkind of cipher that would take million years to break, Scientific American, 1977

# RSA CHALLENGE

The first public description of the RSA cryptosystem was in the paper.

Martin Gardner: A newkind of cipher that would take million years to break, Scientific American, 1977

and in this paper the RSA inventors presented the following challenge.

# RSA CHALLENGE

The first public description of the RSA cryptosystem was in the paper.

Martin Gardner: A newkind of cipher that would take million years to break, Scientific American, 1977

and in this paper the RSA inventors presented the following challenge.

**Decrypt the cryptotext:**

9686 9613 7546 2206 1477 1409 2225 4355 8829 0575 9991 1245 7431 9874 6951 2093 0816 2982 2514 5708 3569 3147 6622 8839 8962 8013 3919 9055 1829 9451 5781 5154

# RSA CHALLENGE

The first public description of the RSA cryptosystem was in the paper.

Martin Gardner: A newkind of cipher that would take million years to break, Scientific American, 1977

and in this paper the RSA inventors presented the following challenge.

**Decrypt the cryptotext:**

9686 9613 7546 2206 1477 1409 2225 4355 8829 0575 9991 1245 7431 9874 6951 2093 0816 2982 2514 5708 3569 3147 6622 8839 8962 8013 3919 9055 1829 9451 5781 5154

**encrypted using the RSA cryptosystem with 129 digit number, called also RSA129**

n: 114 381 625 757 888 867 669 235 779 976 146 612 010 218 296 721 242 362 562 561 842 935 706 935 245 733 897 830 597 123 513 958 705 058 989 075 147 599 290 026 879 543 541.

and with $e = 9007$.

# RSA CHALLENGE

The first public description of the RSA cryptosystem was in the paper.

Martin Gardner: A newkind of cipher that would take million years to break, Scientific American, 1977

and in this paper the RSA inventors presented the following challenge.

**Decrypt the cryptotext:**

9686 9613 7546 2206 1477 1409 2225 4355 8829 0575 9991 1245 7431 9874 6951 2093 0816 2982 2514 5708 3569 3147 6622 8839 8962 8013 3919 9055 1829 9451 5781 5154

**encrypted using the RSA cryptosystem with 129 digit number, called also RSA129**

n: 114 381 625 757 888 867 669 235 779 976 146 612 010 218 296 721 242 362 562 561 842 935 706 935 245 733 897 830 597 123 513 958 705 058 989 075 147 599 290 026 879 543 541.

and with $e = 9007$.

The problem was solved in 1994 by first factorizing n into one 64-bit prime and one 65-bit prime, and then computing the plaintext

THE MAGIC WORDS ARE SQUEMISH OSSIFRAGE

# FROM RSA CRYPTOSYSTEM to RSA SIGNATURES

The idea of RSA cryptosystem is simple.
Public key: modulus $n = pq$ and encryption exponent $e$.
Secret key: decryption exponent $d$ and primes $p, q$

Encryption of a message $w$: $c = w^e$
Decryption of the cryptotext $c$: $w = c^d$.

# FROM RSA CRYPTOSYSTEM to RSA SIGNATURES

The idea of RSA cryptosystem is simple.
Public key: modulus $n = pq$ and encryption exponent $e$.
Secret key: decryption exponent $d$ and primes $p, q$

Encryption of a message $w$: $c = w^e$
Decryption of the cryptotext $c$: $w = c^d$.

Does it has a sense to change the order of these two operations: To do first $c = w^d$ and then $c^e$?.

# FROM RSA CRYPTOSYSTEM to RSA SIGNATURES

The idea of RSA cryptosystem is simple.
Public key: modulus $n = pq$ and encryption exponent $e$.
Secret key: decryption exponent $d$ and primes $p, q$

Encryption of a message $w$: $c = w^e$
Decryption of the cryptotext $c$: $w = c^d$.

Does it has a sense to change the order of these two operations: To do first $c = w^d$ and then $c^e$?. Is this a crazy idea?

# FROM RSA CRYPTOSYSTEM to RSA SIGNATURES

The idea of RSA cryptosystem is simple.
Public key: modulus $n = pq$ and encryption exponent $e$.
Secret key: decryption exponent $d$ and primes $p, q$

Encryption of a message $w$: $c = w^e$
Decryption of the cryptotext $c$: $w = c^d$.

Does it has a sense to change the order of these two operations: To do first $c = w^d$ and then $c^e$?. Is this a crazy idea?

No, we just ned to interpret outcomes of these operations differently. Indeed, $s = w^d$ should be interpreted as the signature of the message $w$ - to be sent together with $w$

# FROM RSA CRYPTOSYSTEM to RSA SIGNATURES

The idea of RSA cryptosystem is simple.
Public key: modulus $n = pq$ and encryption exponent $e$.
Secret key: decryption exponent $d$ and primes $p, q$

Encryption of a message $w$: $c = w^e$
Decryption of the cryptotext $c$: $w = c^d$.

Does it has a sense to change the order of these two operations: To do first $c = w^d$ and then $c^e$?. Is this a crazy idea?

No, we just ned to interpret outcomes of these operations differently. Indeed, $s = w^d$ should be interpreted as the signature of the message $w$ - to be sent together with $w$

and $w = s^e$ can then be verification of such a signature.

# WHY ARE RSA ENCRYPTIONS and SIGNATURES SECURE?

RSA encryptions and signatures are considered as secure because there is not known a methods that could be able factorize in RSA used moduly on current and in near future forseeable classical supercomputers.

# REDUCTIONS of FACTORIZATION PROBLEM

**REDUCTIONS of the FACTORIZATION PROBLEM**

# FIRST REDUCTION

# FIRST REDUCTION

Lemma If there is a polynomial time deterministic (randomized) [quantum] algorithm to find a nontrivial solution of the modular quadratic equations

$$a^2 \equiv 1 \pmod{n},$$

then there is a polynomial time deterministic (randomized) [quantum] algorithm to factorize integers.

# FIRST REDUCTION

Lemma If there is a polynomial time deterministic (randomized) [quantum] algorithm to find a nontrivial solution of the modular quadratic equations

$$a^2 \equiv 1 \pmod{n},$$

then there is a polynomial time deterministic (randomized) [quantum] algorithm to factorize integers.

**Proof.** Let $a \neq \pm 1$ be such that $a^2 \equiv 1 \pmod{n}$.

# FIRST REDUCTION

Lemma If there is a polynomial time deterministic (randomized) [quantum] algorithm to find a nontrivial solution of the modular quadratic equations

$$a^2 \equiv 1 \pmod{n},$$

then there is a polynomial time deterministic (randomized) [quantum] algorithm to factorize integers.

**Proof.** Let $a \neq \pm 1$ be such that $a^2 \equiv 1 \pmod{n}$. Since

$$a^2 - 1 = (a+1)(a-1),$$

if $n$ is not prime, then a prime factor of $n$ has to be a prime factor of either $a+1$ or $a-1$.

# FIRST REDUCTION

Lemma If there is a polynomial time deterministic (randomized) [quantum] algorithm to find a nontrivial solution of the modular quadratic equations

$$a^2 \equiv 1 \pmod{n},$$

then there is a polynomial time deterministic (randomized) [quantum] algorithm to factorize integers.

**Proof.** Let $a \neq \pm 1$ be such that $a^2 \equiv 1 \pmod{n}$. Since

$$a^2 - 1 = (a+1)(a-1),$$

if $n$ is not prime, then a prime factor of $n$ has to be a prime factor of either $a+1$ or $a-1$.

By using Euclid's algorithm to compute

$$gcd(a+1, n) \text{ and } gcd(a-1, n)$$

we can find, in $O(\lg n)$ steps, a prime factor of $n$.

# SECOND REDUCTION

# SECOND REDUCTION

The second key concept is that of the **period** of functions

$$f_{n,x}(k) = x^k \bmod n.$$

# SECOND REDUCTION

The second key concept is that of the **period** of functions

$$f_{n,x}(k) = x^k \bmod n.$$

Period is the smallest integer $r$ such that

$$f_{n,x}(k+r) = f_{n,x}(k)$$

for any $k$, i.e. the smallest $r$ such that

$$x^r \equiv 1 \pmod{n}.$$

# SECOND REDUCTION

The second key concept is that of the **period** of functions

$$f_{n,x}(k) = x^k \bmod n.$$

Period is the smallest integer $r$ such that

$$f_{n,x}(k + r) = f_{n,x}(k)$$

for any $k$, i.e. the smallest $r$ such that

$$x^r \equiv 1 \pmod{n}.$$

**AN ALGORITHM TO SOLVE EQUATION $x^2 \equiv 1 \pmod{n}$.**

# SECOND REDUCTION

The second key concept is that of the **period** of functions

$$f_{n,x}(k) = x^k \bmod n.$$

Period is the smallest integer $r$ such that

$$f_{n,x}(k+r) = f_{n,x}(k)$$

for any $k$, i.e. the smallest $r$ such that

$$x^r \equiv 1 \pmod{n}.$$

**AN ALGORITHM TO SOLVE EQUATION $x^2 \equiv 1 \pmod{n}$.**

1. *Choose randomly* $1 < a < n$.

# SECOND REDUCTION

The second key concept is that of the **period** of functions

$$f_{n,x}(k) = x^k \bmod n.$$

Period is the smallest integer $r$ such that

$$f_{n,x}(k+r) = f_{n,x}(k)$$

for any $k$, i.e. the smallest $r$ such that

$$x^r \equiv 1 \pmod{n}.$$

### AN ALGORITHM TO SOLVE EQUATION $x^2 \equiv 1 \pmod{n}$.

1. *Choose randomly* $1 < a < n$.
2. *Compute* $gcd(a,n)$. *If* $gcd(a,n) \neq 1$ *we have a factor.*

# SECOND REDUCTION

The second key concept is that of the **period** of functions

$$f_{n,x}(k) = x^k \bmod n.$$

Period is the smallest integer $r$ such that

$$f_{n,x}(k+r) = f_{n,x}(k)$$

for any $k$, i.e. the smallest $r$ such that

$$x^r \equiv 1 \pmod{n}.$$

**AN ALGORITHM TO SOLVE EQUATION $x^2 \equiv 1 \pmod{n}$.**

1. *Choose randomly* $1 < a < n$.
2. *Compute* $gcd(a, n)$. *If* $gcd(a, n) \neq 1$ *we have a factor.*
3. *Find period* $r$ *of function* $a^k \bmod n$.

# SECOND REDUCTION

The second key concept is that of the **period** of functions

$$f_{n,x}(k) = x^k \bmod n.$$

Period is the smallest integer $r$ such that

$$f_{n,x}(k+r) = f_{n,x}(k)$$

for any $k$, i.e. the smallest $r$ such that

$$x^r \equiv 1 \pmod{n}.$$

**AN ALGORITHM TO SOLVE EQUATION $x^2 \equiv 1 \pmod{n}$.**

1. *Choose randomly $1 < a < n$.*
2. *Compute $gcd(a, n)$. If $gcd(a, n) \neq 1$ we have a factor.*
3. *Find period $r$ of function $a^k \bmod n$.*
4. *If $r$ is odd or $a^{r/2} \equiv \pm 1 \pmod{n}$,then go to step 1; otherwise stop.*

# SECOND REDUCTION

The second key concept is that of the **period** of functions

$$f_{n,x}(k) = x^k \bmod n.$$

Period is the smallest integer $r$ such that

$$f_{n,x}(k+r) = f_{n,x}(k)$$

for any $k$, i.e. the smallest $r$ such that

$$x^r \equiv 1 \pmod{n}.$$

## AN ALGORITHM TO SOLVE EQUATION $x^2 \equiv 1 \pmod{n}$.

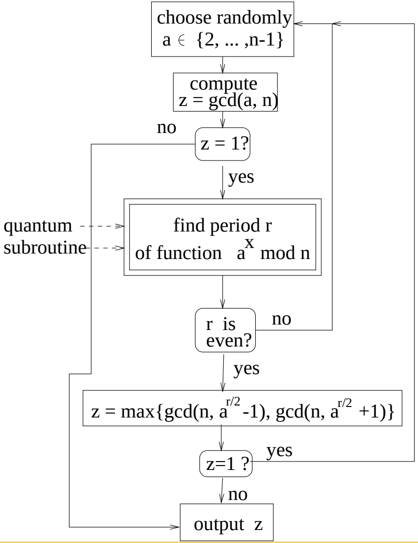
1. *Choose randomly* $1 < a < n$.
2. *Compute* $gcd(a,n)$. *If* $gcd(a,n) \neq 1$ *we have a factor.*
3. *Find period* $r$ *of function* $a^k \bmod n$.
4. *If* $r$ *is odd or* $a^{r/2} \equiv \pm 1 \pmod{n}$,*then go to step 1; otherwise stop.*

If this algorithm stops, then $a^{r/2}$ is a non-trivial solution of the equation

$$x^2 \equiv 1 \pmod{n}.$$

# A GENERAL SCHEME for Shor's ALGORITHM

The following flow diagram shows the general scheme of Shor's quantum factorization algorithm



# SHOR's FACTORIZATION ALGORITHM

1. For given $n, q = 2^d, a$ create states

$$\frac{1}{\sqrt{q}} \sum_{x=0}^{q-1} |n, a, q, x, \mathbf{0}\rangle \text{ and}$$

# SHOR's FACTORIZATION ALGORITHM

1. For given $n, q = 2^d, a$ create states

$$\frac{1}{\sqrt{q}} \sum_{x=0}^{q-1} |n, a, q, x, \mathbf{0}\rangle \text{ and } \frac{1}{\sqrt{q}} \sum_{x=0}^{q-1} |n, a, q, x, a^x \bmod n\rangle$$

# SHOR's FACTORIZATION ALGORITHM

1. For given $n, q = 2^d, a$ create states

$$\frac{1}{\sqrt{q}}\sum_{x=0}^{q-1}|n,a,q,x,\mathbf{0}\rangle \text{ and } \frac{1}{\sqrt{q}}\sum_{x=0}^{q-1}|n,a,q,x,a^x \bmod n\rangle$$

2. By measuring the last register the state collapses into the state

$$\frac{1}{\sqrt{A+1}}\sum_{j=0}^{A}|n,a,q,jr+l,y\rangle \text{ or, shortly } \frac{1}{\sqrt{A+1}}\sum_{j=0}^{A}|jr+l\rangle,$$

where $A$ is the largest integer such that $l + Ar \leq q$, $r$ is the period of $a^x \bmod n$ and $l$ is the offset.

# SHOR's FACTORIZATION ALGORITHM

1. For given $n, q = 2^d, a$ create states

$$\frac{1}{\sqrt{q}}\sum_{x=0}^{q-1}|n,a,q,x,\mathbf{0}\rangle \text{ and} \frac{1}{\sqrt{q}}\sum_{x=0}^{q-1}|n,a,q,x,a^x \bmod n\rangle$$

2. By measuring the last register the state collapses into the state

$$\frac{1}{\sqrt{A+1}}\sum_{j=0}^{A}|n,a,q,jr+l,y\rangle \text{ or, shortly } \frac{1}{\sqrt{A+1}}\sum_{j=0}^{A}|jr+l\rangle,$$

   where $A$ is the largest integer such that $l + Ar \leq q$, $r$ is the period of $a^x \bmod n$ and $l$ is the offset.

3. In case $A = \frac{q}{r} - 1$, the resulting state has the form. $\sqrt{\frac{r}{q}}\sum_{j=0}^{\frac{q}{r}-1}|jr+l\rangle$

# SHOR's FACTORIZATION ALGORITHM

1. For given $n, q = 2^d, a$ create states

$$\frac{1}{\sqrt{q}}\sum_{x=0}^{q-1}|n,a,q,x,\mathbf{0}\rangle \text{ and} \frac{1}{\sqrt{q}}\sum_{x=0}^{q-1}|n,a,q,x,a^x \bmod n\rangle$$

2. By measuring the last register the state collapses into the state

$$\frac{1}{\sqrt{A+1}}\sum_{j=0}^{A}|n,a,q,jr+l,y\rangle \text{ or, shortly } \frac{1}{\sqrt{A+1}}\sum_{j=0}^{A}|jr+l\rangle,$$

   where $A$ is the largest integer such that $l + Ar \leq q$, $r$ is the period of $a^x \bmod n$ and $l$ is the offset.

3. In case $A = \frac{q}{r} - 1$, the resulting state has the form. $\sqrt{\frac{r}{q}}\sum_{j=0}^{\frac{q}{r}-1}|jr+l\rangle$
4. By applying quantum Fourier transformation we get then the state

$$\frac{1}{\sqrt{r}}\sum_{j=0}^{r-1}e^{2\pi i l j/r}|j\frac{q}{r}\rangle.$$

# SHOR's FACTORIZATION ALGORITHM

1. For given $n, q = 2^d, a$ create states

$$\frac{1}{\sqrt{q}}\sum_{x=0}^{q-1}|n,a,q,x,\mathbf{0}\rangle \text{ and } \frac{1}{\sqrt{q}}\sum_{x=0}^{q-1}|n,a,q,x,a^x \bmod n\rangle$$

2. By measuring the last register the state collapses into the state

$$\frac{1}{\sqrt{A+1}}\sum_{j=0}^{A}|n,a,q,jr+l,y\rangle \text{ or, shortly } \frac{1}{\sqrt{A+1}}\sum_{j=0}^{A}|jr+l\rangle,$$

where $A$ is the largest integer such that $l + Ar \leq q$, $r$ is the period of $a^x \bmod n$ and $l$ is the offset.

3. In case $A = \frac{q}{r} - 1$, the resulting state has the form. $\sqrt{\frac{r}{q}}\sum_{j=0}^{\frac{q}{r}-1}|jr+l\rangle$
4. By applying quantum Fourier transformation we get then the state

$$\frac{1}{\sqrt{r}}\sum_{j=0}^{r-1}e^{2\pi i l j/r}|j\frac{q}{r}\rangle.$$

5. By measuring the resulting state we get $c = \frac{jq}{r}$ and if $gcd(j,r) = 1$, what is very likely, then from $c$ and $q$ we can determine the period $r$.

# DISCRETE FOURIER TRANSFORM

# DISCRETE FOURIER TRANSFORM

Discrete Fourier Transform maps a vector $\mathbf{a} = (a_0, a_1, \ldots, a_{n-1})^T$ into the vector $DFT(\mathbf{a}) = A_n\mathbf{a}$, where $A_n$ is an $n \times n$ matrix such that $A_n[i,j] = \frac{1}{\sqrt{n}}\omega^{ij}$ for $0 \leq i, j < n$ and $\omega = e^{2\pi i/n}$ is the $n$**th root of unity**.

# DISCRETE FOURIER TRANSFORM

Discrete Fourier Transform maps a vector $\mathbf{a} = (a_0, a_1, \dots, a_{n-1})^T$ into the vector $DFT(\mathbf{a}) = A_n\mathbf{a}$, where $A_n$ is an $n \times n$ matrix such that $A_n[i,j] = \frac{1}{\sqrt{n}}\omega^{ij}$ for $0 \leq i,j < n$ and $\omega = e^{2\pi i/n}$ is the $n$**th root of unity**.

The matrix $A_n$ has therefore the form

$$A_n = \frac{1}{\sqrt{n}} \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{(n-1)} \\ 1 & \omega^2 & \omega^4 & \dots & \omega^{2(n-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega^{(n-1)} & \omega^{2(n-1)} & \dots & \omega^{(n-1)^2} \end{pmatrix}.$$

# DISCRETE FOURIER TRANSFORM

Discrete Fourier Transform maps a vector $\mathbf{a} = (a_0, a_1, \dots, a_{n-1})^T$ into the vector $DFT(\mathbf{a}) = A_n\mathbf{a}$, where $A_n$ is an $n \times n$ matrix such that $A_n[i,j] = \frac{1}{\sqrt{n}}\omega^{ij}$ for $0 \leq i, j < n$ and $\omega = e^{2\pi i/n}$ is the $n$**th root of unity**.
The matrix $A_n$ has therefore the form

$$A_n = \frac{1}{\sqrt{n}} \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{(n-1)} \\ 1 & \omega^2 & \omega^4 & \dots & \omega^{2(n-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega^{(n-1)} & \omega^{2(n-1)} & \dots & \omega^{(n-1)^2} \end{pmatrix}.$$

The Inverse Discrete Fourier Transform is the mapping

$$DFT^{-1}(\mathbf{a}) = A_n^{-1}\mathbf{a},$$

where

$$A_n^{-1}[i,j] = \frac{1}{\sqrt{n}}\omega^{-ij}.$$

# QUANTUM FOURIER TRANSFORM

# QUANTUM FOURIER TRANSFORM

The Quantum Fourier Transform is a quantum variant of the **Discrete Fourier Transform** (DFT).

# QUANTUM FOURIER TRANSFORM

The Quantum Fourier Transform is a quantum variant of the **Discrete Fourier Transform** (DFT). DFT maps a $q$-dimensional complex vector

$$\{f(0), f(1), \ldots, f(q-1)\} \text{ into } \{\bar{f}(0), \bar{f}(1), \ldots, \bar{f}(q-1)\},$$

where for any $c \in \{0, \ldots, q-1\}$

$$\bar{f}(c) = \frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} e^{2\pi iac/q} f(a), \tag{1}$$

# QUANTUM FOURIER TRANSFORM

The Quantum Fourier Transform is a quantum variant of the **Discrete Fourier Transform** (DFT). DFT maps a $q$-dimensional complex vector

$$\{f(0), f(1), \ldots, f(q-1)\} \text{ into } \{\bar{f}(0), \bar{f}(1), \ldots, \bar{f}(q-1)\},$$

where for any $c \in \{0, \ldots, q-1\}$

$$\bar{f}(c) = \frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} e^{2\pi iac/q} f(a), \tag{1}$$

The quantum version of DFT (QFT) is the unitary transformation

$$\mathrm{QFT}_q : |a\rangle \to \frac{1}{\sqrt{q}} \sum_{c=0}^{q-1} e^{2\pi iac/q} |c\rangle \tag{2}$$

The quantum version of DFT (QFT) is the unitary transformation

$$\mathrm{QFT}_q : |a\rangle \to \frac{1}{\sqrt{q}} \sum_{c=0}^{q-1} e^{2\pi iac/q} |c\rangle \tag{3}$$

defined for $0 \leq a < q$, by the unitary matrix

$$F_q = \frac{1}{\sqrt{q}} \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{(q-1)} \\ 1 & \omega^2 & \omega^4 & \dots & \omega^{2(q-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega^{(q-1)} & \omega^{2(q-1)} & \dots & \omega^{(q-1)^2} \end{pmatrix},$$

where $\omega = e^{2\pi i/q}$ is the $q$**th root of unity**.

The quantum version of DFT (QFT) is the unitary transformation

$$\mathrm{QFT}_q : |a\rangle \to \frac{1}{\sqrt{q}} \sum_{c=0}^{q-1} e^{2\pi iac/q} |c\rangle \tag{3}$$

defined for $0 \leq a < q$, by the unitary matrix

$$F_q = \frac{1}{\sqrt{q}} \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{(q-1)} \\ 1 & \omega^2 & \omega^4 & \dots & \omega^{2(q-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega^{(q-1)} & \omega^{2(q-1)} & \dots & \omega^{(q-1)^2} \end{pmatrix},$$

where $\omega = e^{2\pi i/q}$ is the $q$**th root of unity**.
If applied to a quantum superposition, $\mathrm{QFT}_q$ performs as follows;

$$\mathrm{QFT}_q : \sum_{a=0}^{q-1} f(a)|a\rangle \to \sum_{c=0}^{q-1} \bar{f}(c)|c\rangle,$$

where $\bar{f}(c)$ is defined by (1).

The quantum version of DFT (QFT) is the unitary transformation

$$\mathrm{QFT}_q : |a\rangle \rightarrow \frac{1}{\sqrt{q}} \sum_{c=0}^{q-1} e^{2\pi iac/q} |c\rangle \tag{3}$$

defined for $0 \le a < q$, by the unitary matrix

$$F_q = \frac{1}{\sqrt{q}} \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{(q-1)} \\ 1 & \omega^2 & \omega^4 & \dots & \omega^{2(q-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega^{(q-1)} & \omega^{2(q-1)} & \dots & \omega^{(q-1)^2} \end{pmatrix},$$

where $\omega = e^{2\pi i/q}$ is the $q$**th root of unity**.
If applied to a quantum superposition, $\mathrm{QFT}_q$ performs as follows;

$$\mathrm{QFT}_q : \sum_{a=0}^{q-1} f(a)|a\rangle \rightarrow \sum_{c=0}^{q-1} \bar{f}(c)|c\rangle,$$

where $\bar{f}(c)$ is defined by (1).
Observe that

$$QFT_q : |\mathbf{0}\rangle \rightarrow \frac{1}{\sqrt{q}} \sum_{i=0}^{q-1} |i\rangle,$$

# SHOR's ALGORITHM - PHASE 1

Given an $m$ bit integer $n$ we choose a $n^2 \leq q = 2^d \leq 2n^2$

# SHOR's ALGORITHM - PHASE 1

Given an $m$ bit integer $n$ we choose a $n^2 \leq q = 2^d \leq 2n^2$and start with five registers in states $|n, a, q, \mathbf{0}, \mathbf{0}\rangle$.
Application of the Hadamard transformation to the fourth register yields state

$$\frac{1}{\sqrt{q}} \sum_{x=0}^{q-1} |n, a, q, x, \mathbf{0}\rangle.$$

# SHOR's ALGORITHM - PHASE 1

Given an $m$ bit integer $n$ we choose a $n^2 \leq q = 2^d \leq 2n^2$and start with five registers in states $|n, a, q, \mathbf{0}, \mathbf{0}\rangle$.
Application of the Hadamard transformation to the fourth register yields state

$$\frac{1}{\sqrt{q}} \sum_{x=0}^{q-1} |n, a, q, x, \mathbf{0}\rangle.$$

and using quantum parallelism we compute $a^x \bmod n$ for all $x$ in one step, to get

$$\frac{1}{\sqrt{q}} \sum_{x=0}^{q-1} |n, a, q, x, a^x \bmod n\rangle.$$

# SHOR's ALGORITHM - PHASE 1

Given an $m$ bit integer $n$ we choose a $n^2 \leq q = 2^d \leq 2n^2$and start with five registers in states $|n, a, q, \mathbf{0}, \mathbf{0}\rangle$.

Application of the Hadamard transformation to the fourth register yields state

$$\frac{1}{\sqrt{q}} \sum_{x=0}^{q-1} |n, a, q, x, \mathbf{0}\rangle.$$

and using quantum parallelism we compute $a^x \bmod n$ for all $x$ in one step, to get

$$\frac{1}{\sqrt{q}} \sum_{x=0}^{q-1} |n, a, q, x, a^x \bmod n\rangle.$$

As the next step we perform a measurement on the last register.

# SHOR's ALGORITHM - PHASE 1

Given an $m$ bit integer $n$ we choose a $n^2 \leq q = 2^d \leq 2n^2$and start with five registers in states $|n, a, q, \mathbf{0}, \mathbf{0}\rangle$.

Application of the Hadamard transformation to the fourth register yields state

$$\frac{1}{\sqrt{q}} \sum_{x=0}^{q-1} |n, a, q, x, \mathbf{0}\rangle.$$

and using quantum parallelism we compute $a^x \bmod n$ for all $x$ in one step, to get

$$\frac{1}{\sqrt{q}} \sum_{x=0}^{q-1} |n, a, q, x, a^x \bmod n\rangle.$$

As the next step we perform a measurement on the last register.Let $y$ be the value obtained, i.e. $y = a^l \bmod n$ for the smallest $l$ $(l_y)$ with this property.

# SHOR's ALGORITHM - PHASE 1

Given an $m$ bit integer $n$ we choose a $n^2 \leq q = 2^d \leq 2n^2$and start with five registers in states $|n, a, q, \mathbf{0}, \mathbf{0}\rangle$.

Application of the Hadamard transformation to the fourth register yields state

$$\frac{1}{\sqrt{q}} \sum_{x=0}^{q-1} |n, a, q, x, \mathbf{0}\rangle.$$

and using quantum parallelism we compute $a^x \bmod n$ for all $x$ in one step, to get

$$\frac{1}{\sqrt{q}} \sum_{x=0}^{q-1} |n, a, q, x, a^x \bmod n\rangle.$$

As the next step we perform a measurement on the last register.Let $y$ be the value obtained, i.e. $y = a^l \bmod n$ for the smallest $l$ ($l_y$) with this property. If $r$ is the period of $f_{n,a}$, then $a^{l_y} \equiv a^{jr+l_y} \pmod{n}$ for all $j$.

# SHOR's ALGORITHM - PHASE 1

Given an $m$ bit integer $n$ we choose a $n^2 \leq q = 2^d \leq 2n^2$and start with five registers in states $|n, a, q, \mathbf{0}, \mathbf{0}\rangle$.
Application of the Hadamard transformation to the fourth register yields state

$$\frac{1}{\sqrt{q}} \sum_{x=0}^{q-1} |n, a, q, x, \mathbf{0}\rangle.$$

and using quantum parallelism we compute $a^x \bmod n$ for all $x$ in one step, to get

$$\frac{1}{\sqrt{q}} \sum_{x=0}^{q-1} |n, a, q, x, a^x \bmod n\rangle.$$

As the next step we perform a measurement on the last register.Let $y$ be the value obtained, i.e. $y = a^l \bmod n$ for the smallest $l$ ($l_y$) with this property. If $r$ is the period of $f_{n,a}$, then $a^{l_y} \equiv a^{jr+l_y} \pmod{n}$ for all $j$. Therefore, the measurement actually selects the sequence of $x$'s values (in the fourth register), $l_y, l_y + r, l_y + 2r, \ldots, l_y + Ar$, where $A$ is the largest integer such that $l_y + Ar \leq q - 1$. Clearly, $A \approx \frac{q}{r}$.

# SHOR's ALGORITHM - PHASE 1

Given an $m$ bit integer $n$ we choose a $n^2 \leq q = 2^d \leq 2n^2$and start with five registers in states $|n, a, q, \mathbf{0}, \mathbf{0}\rangle$.
Application of the Hadamard transformation to the fourth register yields state

$$\frac{1}{\sqrt{q}} \sum_{x=0}^{q-1} |n, a, q, x, \mathbf{0}\rangle.$$

and using quantum parallelism we compute $a^x$ mod $n$ for all $x$ in one step, to get

$$\frac{1}{\sqrt{q}} \sum_{x=0}^{q-1} |n, a, q, x, a^x \bmod n\rangle.$$

As the next step we perform a measurement on the last register.Let $y$ be the value obtained, i.e. $y = a^l$ mod $n$ for the smallest $l$ ($l_y$) with this property. If $r$ is the period of $f_{n,a}$, then $a^{l_y} \equiv a^{jr+l_y} \pmod{n}$ for all $j$. Therefore, the measurement actually selects the sequence of $x$'s values (in the fourth register), $l_y, l_y + r, l_y + 2r, \ldots, l_y + Ar$, where $A$ is the largest integer such that $l_y + Ar \leq q - 1$. Clearly, $A \approx \frac{q}{r}$. The post-measurement state is then

$$|\phi_l\rangle = \frac{1}{\sqrt{A+1}} \sum_{j=0}^{A} |n, a, q, jr + l_y, y\rangle$$

# SHOR's ALGORITHM - PHASE 1

Given an $m$ bit integer $n$ we choose a $n^2 \leq q = 2^d \leq 2n^2$and start with five registers in states $|n, a, q, \mathbf{0}, \mathbf{0}\rangle$.
Application of the Hadamard transformation to the fourth register yields state

$$\frac{1}{\sqrt{q}} \sum_{x=0}^{q-1} |n, a, q, x, \mathbf{0}\rangle.$$

and using quantum parallelism we compute $a^x$ mod $n$ for all $x$ in one step, to get

$$\frac{1}{\sqrt{q}} \sum_{x=0}^{q-1} |n, a, q, x, a^x \bmod n\rangle.$$

As the next step we perform a measurement on the last register.Let $y$ be the value obtained, i.e. $y = a^l$ mod $n$ for the smallest $l$ ($l_y$) with this property. If $r$ is the period of $f_{n,a}$, then $a^{l_y} \equiv a^{jr+l_y} \pmod{n}$ for all $j$. Therefore, the measurement actually selects the sequence of $x$'s values (in the fourth register), $l_y, l_y + r, l_y + 2r, \ldots, l_y + Ar$, where $A$ is the largest integer such that $l_y + Ar \leq q - 1$. Clearly, $A \approx \frac{q}{r}$. The post-measurement state is then

$$|\phi_l\rangle = \frac{1}{\sqrt{A+1}} \sum_{j=0}^{A} |n, a, q, jr + l_y, y\rangle = \frac{1}{\sqrt{A+1}} \sum_{j=0}^{A} |jr + l_y\rangle. \tag{4}$$

# SHOR's ALGORITHM - SECOND PHASE

# SHOR's ALGORITHM - SECOND PHASE

From now on we consider only a special case. namely, that $r$ divides $q$. Then $A = \frac{q}{r} - 1$, last state can be written as $|\phi_l\rangle = \sqrt{\frac{r}{q}} \sum_{j=0}^{\frac{q}{r}-1} |jr + l_y\rangle$

# SHOR's ALGORITHM - SECOND PHASE

From now on we consider only a special case. namely, that $r$ divides $q$. Then $A = \frac{q}{r} - 1$, last state can be written as $|\phi_l\rangle = \sqrt{\frac{r}{q}} \sum_{j=0}^{\frac{q}{r}-1} |jr + l_y\rangle$ and after $\mathrm{QFT}_q$ is applied on $|\phi_l\rangle$ we get:

# SHOR's ALGORITHM - SECOND PHASE

From now on we consider only a special case. namely, that $r$ divides $q$. Then $A = \frac{q}{r} - 1$, last state can be written as $|\phi_l\rangle = \sqrt{\frac{r}{q}} \sum_{j=0}^{\frac{q}{r}-1} |jr + l_y\rangle$ and after $\mathrm{QFT}_q$ is applied on $|\phi_l\rangle$ we get:

$$\mathrm{QFT}_q|\phi_l\rangle = \frac{1}{\sqrt{q}} \sum_{c=0}^{q-1} \sqrt{\frac{r}{q}} \sum_{j=0}^{\frac{q}{r}-1} e^{2\pi ic(jr+l_y)/q}|c\rangle$$

# SHOR's ALGORITHM - SECOND PHASE

From now on we consider only a special case. namely, that $r$ divides $q$. Then $A = \frac{q}{r} - 1$, last state can be written as $|\phi_l\rangle = \sqrt{\frac{r}{q}} \sum_{j=0}^{\frac{q}{r}-1} |jr + l_y\rangle$ and after $\mathrm{QFT}_q$ is applied on $|\phi_l\rangle$ we get:

$$\mathrm{QFT}_q|\phi_l\rangle = \frac{1}{\sqrt{q}} \sum_{c=0}^{q-1} \sqrt{\frac{r}{q}} \sum_{j=0}^{\frac{q}{r}-1} e^{2\pi i c(jr+l_y)/q} |c\rangle$$

$$= \frac{\sqrt{r}}{q} \sum_{c=0}^{q-1} e^{2\pi i l_y c/q} \left( \sum_{j=0}^{\frac{q}{r}-1} e^{2\pi i jcr/q} \right) |c\rangle$$

# SHOR's ALGORITHM - SECOND PHASE

From now on we consider only a special case. namely, that $r$ divides $q$. Then $A = \frac{q}{r} - 1$, last state can be written as $|\phi_l\rangle = \sqrt{\frac{r}{q}} \sum_{j=0}^{\frac{q}{r}-1} |jr + l_y\rangle$ and after $\mathrm{QFT}_q$ is applied on $|\phi_l\rangle$ we get:

$$\mathrm{QFT}_q|\phi_l\rangle = \frac{1}{\sqrt{q}} \sum_{c=0}^{q-1} \sqrt{\frac{r}{q}} \sum_{j=0}^{\frac{q}{r}-1} e^{2\pi i c(jr+l_y)/q} |c\rangle$$

$$= \frac{\sqrt{r}}{q} \sum_{c=0}^{q-1} e^{2\pi i l_y c/q} \left( \sum_{j=0}^{\frac{q}{r}-1} e^{2\pi i jcr/q} \right) |c\rangle = \sum_{c=0}^{q-1} \alpha_c |c\rangle.$$

If $c$ is a multiple of $\frac{q}{r}$, then $e^{2\pi i jcr/q} = 1$

# SHOR's ALGORITHM - SECOND PHASE

From now on we consider only a special case. namely, that $r$ divides $q$. Then $A = \frac{q}{r} - 1$, last state can be written as $|\phi_l\rangle = \sqrt{\frac{r}{q}} \sum_{j=0}^{\frac{q}{r}-1} |jr + l_y\rangle$ and after QFT$_q$ is applied on $|\phi_l\rangle$ we get:

$$\text{QFT}_q|\phi_l\rangle = \frac{1}{\sqrt{q}} \sum_{c=0}^{q-1} \sqrt{\frac{r}{q}} \sum_{j=0}^{\frac{q}{r}-1} e^{2\pi i c(jr+l_y)/q} |c\rangle$$

$$= \frac{\sqrt{r}}{q} \sum_{c=0}^{q-1} e^{2\pi i l_y c/q} \left( \sum_{j=0}^{\frac{q}{r}-1} e^{2\pi i jcr/q} \right) |c\rangle = \sum_{c=0}^{q-1} \alpha_c |c\rangle.$$

If $c$ is a multiple of $\frac{q}{r}$, then $e^{2\pi i jcr/q} = 1$and if $c$ is not a multiple of $\frac{q}{r}$, then $\sum_{j=0}^{\frac{q}{r}-1} e^{2\pi i jcr/q} = 0$,

# SHOR's ALGORITHM - SECOND PHASE

From now on we consider only a special case. namely, that $r$ divides $q$. Then $A = \frac{q}{r} - 1$, last state can be written as $|\phi_l\rangle = \sqrt{\frac{r}{q}} \sum_{j=0}^{\frac{q}{r}-1} |jr + l_y\rangle$ and after $\mathrm{QFT}_q$ is applied on $|\phi_l\rangle$ we get:

$$\mathrm{QFT}_q|\phi_l\rangle = \frac{1}{\sqrt{q}} \sum_{c=0}^{q-1} \sqrt{\frac{r}{q}} \sum_{j=0}^{\frac{q}{r}-1} e^{2\pi i c(jr+l_y)/q}|c\rangle$$

$$= \frac{\sqrt{r}}{q} \sum_{c=0}^{q-1} e^{2\pi i l_y c/q} \left( \sum_{j=0}^{\frac{q}{r}-1} e^{2\pi i jcr/q} \right) |c\rangle = \sum_{c=0}^{q-1} \alpha_c |c\rangle.$$

If $c$ is a multiple of $\frac{q}{r}$, then $e^{2\pi i jcr/q} = 1$and if $c$ is not a multiple of $\frac{q}{r}$, then $\sum_{j=0}^{\frac{q}{r}-1} e^{2\pi i jcr/q} = 0$,because the above sum is over a set of $\frac{q}{r}$ roots of unity equally spaced around the unit circle.

# SHOR's ALGORITHM - SECOND PHASE

From now on we consider only a special case. namely, that $r$ divides $q$. Then $A = \frac{q}{r} - 1$, last state can be written as $|\phi_l\rangle = \sqrt{\frac{r}{q}} \sum_{j=0}^{\frac{q}{r}-1} |jr + l_y\rangle$ and after $\mathrm{QFT}_q$ is applied on $|\phi_l\rangle$ we get:

$$\mathrm{QFT}_q|\phi_l\rangle = \frac{1}{\sqrt{q}} \sum_{c=0}^{q-1} \sqrt{\frac{r}{q}} \sum_{j=0}^{\frac{q}{r}-1} e^{2\pi i c(jr+l_y)/q} |c\rangle$$

$$= \frac{\sqrt{r}}{q} \sum_{c=0}^{q-1} e^{2\pi i l_y c/q} \left( \sum_{j=0}^{\frac{q}{r}-1} e^{2\pi i jcr/q} \right) |c\rangle = \sum_{c=0}^{q-1} \alpha_c |c\rangle.$$

If $c$ is a multiple of $\frac{q}{r}$, then $e^{2\pi i jcr/q} = 1$and if $c$ is not a multiple of $\frac{q}{r}$, then $\sum_{j=0}^{\frac{q}{r}-1} e^{2\pi i jcr/q} = 0$,because the above sum is over a set of $\frac{q}{r}$ roots of unity equally spaced around the unit circle. Thus

$$\alpha_c = \begin{cases} \frac{1}{\sqrt{r}} e^{2\pi i l_y c/q}, & \text{if } c \text{ is a multiple of } \frac{q}{r}; \\ 0, & \text{otherwise;} \end{cases}$$

Therefore

$$|\phi_{out}\rangle = \mathrm{QFT}_q|\phi_l\rangle = \frac{1}{\sqrt{r}}\sum_{j=0}^{r-1} e^{2\pi i l_y j/r}|j\frac{q}{r}\rangle.$$

Therefore

$$|\phi_{out}\rangle = \mathrm{QFT}_q|\phi_l\rangle = \frac{1}{\sqrt{r}}\sum_{j=0}^{r-1} e^{2\pi i l_y j/r}|j\frac{q}{r}\rangle.$$

The key point now is that the trouble-making offset $l_y$ appears now in the phase factor $e^{2\pi i l_y j/r}$ and has influence neither on the probabilities nor on values obtained by the measurement.

# SHOR's ALGORTHM - PHASE 3

**Period extraction**

[1]As observed by Shor (1994) and shown by Cleve et al. (1998), the expected number of trials can be put down to a constant.

# SHOR's ALGORTHM - PHASE 3

**Period extraction**

Each measurement of the state $|\phi_{out}\rangle$ therefore yields one of the values $c = \lambda \frac{q}{r}$, $\lambda \in \{0, 1, \ldots r-1\}$, where each $\lambda$ is chosen with the same probability $\frac{1}{r}$.

[1] As observed by Shor (1994) and shown by Cleve et al. (1998), the expected number of trials can be put down to a constant.

# SHOR's ALGORTHM - PHASE 3

**Period extraction**

Each measurement of the state $|\phi_{out}\rangle$ therefore yields one of the values $c = \lambda\frac{q}{r}$, $\lambda \in \{0, 1, \ldots r-1\}$, where each $\lambda$ is chosen with the same probability $\frac{1}{r}$.

Observe also that in this case the QFT transforms a function with the period $r$ (and an offset $l$) to a function with the period $\frac{q}{r}$ and offset 0.

[1] As observed by Shor (1994) and shown by Cleve et al. (1998), the expected number of trials can be put down to a constant.

# SHOR's ALGORTHM - PHASE 3

## Period extraction

Each measurement of the state $|\phi_{out}\rangle$ therefore yields one of the values $c = \lambda\frac{q}{r}$, $\lambda \in \{0, 1, \ldots r-1\}$, where each $\lambda$ is chosen with the same probability $\frac{1}{r}$.

Observe also that in this case the QFT transforms a function with the period $r$ (and an offset $l$) to a function with the period $\frac{q}{r}$ and offset 0.

After each measurement we therefore know $c$ and $q$ and

$$\frac{c}{q} = \frac{\lambda}{r},$$

where $\lambda$ is randomly chosen.

[1] As observed by Shor (1994) and shown by Cleve et al. (1998), the expected number of trials can be put down to a constant.

# SHOR's ALGORTHM - PHASE 3

**Period extraction**

Each measurement of the state $|\phi_{out}\rangle$ therefore yields one of the values $c = \lambda\frac{q}{r}$, $\lambda \in \{0, 1, \dots r-1\}$, where each $\lambda$ is chosen with the same probability $\frac{1}{r}$.

Observe also that in this case the QFT transforms a function with the period $r$ (and an offset $l$) to a function with the period $\frac{q}{r}$ and offset 0.

After each measurement we therefore know $c$ and $q$ and

$$\frac{c}{q} = \frac{\lambda}{r},$$

where $\lambda$ is randomly chosen.

If $gcd(\lambda, r) = 1$, then from $q$ we can determine $r$ by dividing $q$ with $gcd(c, q)$.

[1] As observed by Shor (1994) and shown by Cleve et al. (1998), the expected number of trials can be put down to a constant.

# SHOR's ALGORTHM - PHASE 3

**Period extraction**

Each measurement of the state $|\phi_{out}\rangle$ therefore yields one of the values $c = \lambda\frac{q}{r}$, $\lambda \in \{0, 1, \dots r-1\}$, where each $\lambda$ is chosen with the same probability $\frac{1}{r}$.

Observe also that in this case the QFT transforms a function with the period $r$ (and an offset $l$) to a function with the period $\frac{q}{r}$ and offset 0.

After each measurement we therefore know $c$ and $q$ and

$$\frac{c}{q} = \frac{\lambda}{r},$$

where $\lambda$ is randomly chosen.

If $gcd(\lambda, r) = 1$, then from $q$ we can determine $r$ by dividing $q$ with $gcd(c, q)$. Since $\lambda$ is chosen randomly, the probability that $gcd(\lambda, r) = 1$ is greater than $\Omega(\frac{1}{\lg \lg r})$.

[1] As observed by Shor (1994) and shown by Cleve et al. (1998), the expected number of trials can be put down to a constant.

# SHOR's ALGORTHM - PHASE 3

## Period extraction

Each measurement of the state $|\phi_{out}\rangle$ therefore yields one of the values $c = \lambda\frac{q}{r}$, $\lambda \in \{0, 1, \ldots r-1\}$, where each $\lambda$ is chosen with the same probability $\frac{1}{r}$.

Observe also that in this case the QFT transforms a function with the period $r$ (and an offset $l$) to a function with the period $\frac{q}{r}$ and offset 0.

After each measurement we therefore know $c$ and $q$ and

$$\frac{c}{q} = \frac{\lambda}{r},$$

where $\lambda$ is randomly chosen.

If $gcd(\lambda, r) = 1$, then from $q$ we can determine $r$ by dividing $q$ with $gcd(c, q)$. Since $\lambda$ is chosen randomly, the probability that $gcd(\lambda, r) = 1$ is greater than $\Omega(\frac{1}{\lg \lg r})$. If the above computation is repeated $\mathcal{O}(\lg \lg r)$ times, then the success probability can be as close to 1 as desired and therefore $r$ can be determined efficiently.[1]

[1] As observed by Shor (1994) and shown by Cleve et al. (1998), the expected number of trials can be put down to a constant.

In the general case, i.e., if

$$A \neq \frac{q}{r} - 1,$$

there is only a more sophisticated computation of the resulting probabilities and a more sophisticated way to determine $r$ (using a continuous fraction method to extract the period from its approximation).

# ANALYSIS of SHORs FACTORIZATION ALGORITHM

# ANALYSIS of SHORs FACTORIZATION ALGORITHM

- Efficient implementations of $\mathrm{QFT}_q$, concerning the number of gates, are known for the the case $q = 2^d$ or $q$ is **smooth** (that is if factors of $q$ are smaller than $\mathcal{O}(\lg q)$).

# ANALYSIS of SHORs FACTORIZATION ALGORITHM

- Efficient implementations of $\mathrm{QFT}_q$, concerning the number of gates, are known for the the case $q = 2^d$ or $q$ is **smooth** (that is if factors of $q$ are smaller than $\mathcal{O}(\lg q)$).
- Efficient implementations of modular operations (including exponentiation) are known.

# ANALYSIS of SHORs FACTORIZATION ALGORITHM

- Efficient implementations of $\mathrm{QFT}_q$, concerning the number of gates, are known for the the case $q = 2^d$ or $q$ is **smooth** (that is if factors of $q$ are smaller than $\mathcal{O}(\lg q)$).
- Efficient implementations of modular operations (including exponentiation) are known.
- First estimation said that $300 \lg n$ gates are needed to factor $n$.

# ANALYSIS of SHORs FACTORIZATION ALGORITHM

- Efficient implementations of $\mathrm{QFT}_q$, concerning the number of gates, are known for the the case $q = 2^d$ or $q$ is **smooth** (that is if factors of $q$ are smaller than $\mathcal{O}(\lg q)$).
- Efficient implementations of modular operations (including exponentiation) are known.
- First estimation said that $300 \lg n$ gates are needed to factor $n$.
- An estimation said that to factor 130 digit integers would require two weeks on an ideal quantum computer with switching frequency 1 MHz.

# ANALYSIS of SHORs FACTORIZATION ALGORITHM

- Efficient implementations of $\mathrm{QFT}_q$, concerning the number of gates, are known for the the case $q = 2^d$ or $q$ is **smooth** (that is if factors of $q$ are smaller than $\mathcal{O}(\lg q)$).
- Efficient implementations of modular operations (including exponentiation) are known.
- First estimation said that $300 \lg n$ gates are needed to factor $n$.
- An estimation said that to factor 130 digit integers would require two weeks on an ideal quantum computer with switching frequency 1 MHz. However, to factor 260-digit number only 16 times larger time would be needed.

# ANALYSIS of SHORs FACTORIZATION ALGORITHM

- Efficient implementations of $\text{QFT}_q$, concerning the number of gates, are known for the the case $q = 2^d$ or $q$ is **smooth** (that is if factors of $q$ are smaller than $\mathcal{O}(\lg q)$).
- Efficient implementations of modular operations (including exponentiation) are known.
- First estimation said that $300 \lg n$ gates are needed to factor $n$.
- An estimation said that to factor 130 digit integers would require two weeks on an ideal quantum computer with switching frequency 1 MHz. However, to factor 260-digit number only 16 times larger time would be needed.
- It has been shown that there is polynomial time factorization even in the case only one pure qubit is available and the rest of quantumness available is in mixed states.
- To factor an integer $n$ Shor's algorithm uses $\mathcal{O}(\lg^3 n)$ steps and success probability is guaranteed to be at least $\Omega(\frac{1}{\lg \lg n})$.

# ANALYSIS of SHORs FACTORIZATION ALGORITHM

- Efficient implementations of $\mathrm{QFT}_q$, concerning the number of gates, are known for the the case $q = 2^d$ or $q$ is **smooth** (that is if factors of $q$ are smaller than $\mathcal{O}(\lg q)$).
- Efficient implementations of modular operations (including exponentiation) are known.
- First estimation said that $300 \lg n$ gates are needed to factor $n$.
- An estimation said that to factor 130 digit integers would require two weeks on an ideal quantum computer with switching frequency 1 MHz. However, to factor 260-digit number only 16 times larger time would be needed.
- It has been shown that there is polynomial time factorization even in the case only one pure qubit is available and the rest of quantumness available is in mixed states.
- To factor an integer $n$ Shor's algorithm uses $\mathcal{O}(\lg^3 n)$ steps and success probability is guaranteed to be at least $\Omega(\frac{1}{\lg \lg n})$.
- An analysis of Shor's algorithm therefore shows that by running the algorithm $\mathcal{O}(\lg \lg n)$ times, therefore in total in $\mathcal{O}(\lg^3 n \lg \lg n)$ times we have very high success probability.

# EFFICIANT IMPLEMENTATION of QFT

# EFFICIANT IMPLEMENTATION of QFT

The clue to the design of a quantum circuit to implement the QFT

$$|x\rangle \to \frac{1}{\sqrt{2^m}} \sum_{y=0}^{2^m-1} e^{\frac{2\pi ixy}{2^m}} |y\rangle$$

for $|x\rangle = |x_{m-1}\rangle|x_{m-2}\rangle \dots |x_0\rangle$, wher $x_i$s are bits, is the decomposition

# EFFICIANT IMPLEMENTATION of QFT

The clue to the design of a quantum circuit to implement the QFT

$$|x\rangle \rightarrow \frac{1}{\sqrt{2^m}} \sum_{y=0}^{2^m-1} e^{\frac{2\pi ixy}{2^m}} |y\rangle$$

for $|x\rangle = |x_{m-1}\rangle|x_{m-2}\rangle \dots |x_0\rangle$, wher $x_i$s are bits, is the decomposition

$$\sum_{y=0}^{2^m-1} e^{\frac{2\pi ixy}{2^m}} |y\rangle = (|0\rangle + e^{\frac{\pi ix}{2^0}} |1\rangle)(|0\rangle + e^{\frac{\pi ix}{2^1}} |1\rangle) \dots (|0\rangle + e^{\frac{\pi ix}{2^{m-1}}} |1\rangle)$$

# EFFICIANT IMPLEMENTATION of QFT

The clue to the design of a quantum circuit to implement the QFT

$$|x\rangle \to \frac{1}{\sqrt{2^m}} \sum_{y=0}^{2^m-1} e^{\frac{2\pi ixy}{2^m}} |y\rangle$$

for $|x\rangle = |x_{m-1}\rangle|x_{m-2}\rangle \dots |x_0\rangle$, wher $x_i$s are bits, is the decomposition

$$\sum_{y=0}^{2^m-1} e^{\frac{2\pi ixy}{2^m}} |y\rangle = (|0\rangle + e^{\frac{\pi ix}{2^0}}|1\rangle)(|0\rangle + e^{\frac{\pi ix}{2^1}}|1\rangle)\dots(|0\rangle + e^{\frac{\pi ix}{2^{m-1}}}|1\rangle)$$

The exponent in th $l$-th factor of the above decomposition can be written as follows

$$\exp(\frac{\pi i(2^{m-1}x_{m-1} + 2^{m-2}x_{m-2} + \dots + 2x_1 + x_0)}{2^{l-1}})$$

$$= \exp(\frac{\pi i(2^{l-1}x_{l-1} + 2^{l-2}x_{l-2} + \dots + 2x_1 + x_0)}{2^{l-1}})$$

$$= (-1)^{x_{l-1}} exp(\frac{\pi ix_{l-2}}{2}) \dots exp(\frac{\pi ix_1}{2^{l-2}}) exp(\frac{\pi ix_0}{2^{l-1}})$$

# CIRCUIT for QFT

# CIRCUIT for QFT

If the unitary

$$\phi_{kl} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{\frac{\pi i}{2^{l-k}}} \end{pmatrix}$$

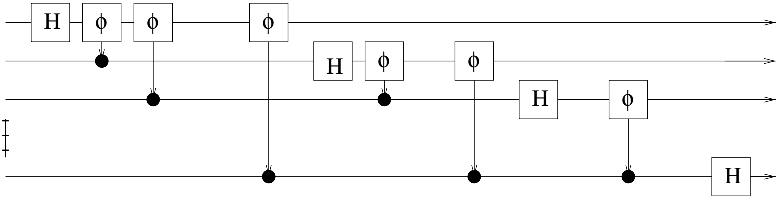
is considered which acts on the $l$th and $k$th qubit,

# CIRCUIT for QFT

If the unitary

$$\phi_{kl} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{\frac{\pi i}{2^{l-k}}} \end{pmatrix}$$

is considered which acts on the $l$th and $k$th qubit, then the resulting circuit for QFT has the form:




It has therefoe $O(n^2)$ gates.

# HIDDEN SUBGROUP PROBLEM

Another famous Shor's algorithm is the one to compute discrete logarithm for modular computation - also the problem for which we do not know an efficient classical algorithm.

These two problems, and many other including those discussed in previous lecture,

# HIDDEN SUBGROUP PROBLEM

Another famous Shor's algorithm is the one to compute discrete logarithm for modular computation - also the problem for which we do not know an efficient classical algorithm.

These two problems, and many other including those discussed in previous lecture, deal with a special clase of so called **Hidden subgroup problems**.

**Given is:** An (efficiently computable) function $f : G \to R$, where $G$ is a group and $R$ is a finite set.

**Given is a promise:** There exists a subgroup $G_0 \leq G$ such that $f$ is constant and distinct on the cossets of $G$ defined by $G_0$.

**Task:** Find a generating set for $G_0$ (in a polynomial time (in $\lg |G|$) with respect to the number of calls to the oracle for $f$ and in the overall polynomial time).

# HIDDEN SUBGROUP PROBLEM

Another famous Shor's algorithm is the one to compute discrete logarithm for modular computation - also the problem for which we do not know an efficient classical algorithm.

These two problems, and many other including those discussed in previous lecture, deal with a special clase of so called **Hidden subgroup problems**.

**Given is:** An (efficiently computable) function $f : G \rightarrow R$, where $G$ is a group and $R$ is a finite set.
**Given is a promise:** There exists a subgroup $G_0 \leq G$ such that $f$ is constant and distinct on the cossets of $G$ defined by $G_0$.

**Task:** Find a generating set for $G_0$ (in a polynomial time (in $\lg |G|$) with respect to the number of calls to the oracle for $f$ and in the overall polynomial time).

It has been shown that Hidden subroup problem has efficient solution in the case the group is commutative.

# HIDDEN SUBGROUP PROBLEM

Another famous Shor's algorithm is the one to compute discrete logarithm for modular computation - also the problem for which we do not know an efficient classical algorithm.

These two problems, and many other including those discussed in previous lecture, deal with a special clase of so called **Hidden subgroup problems**.

**Given is:** An (efficiently computable) function $f : G \to R$, where $G$ is a group and $R$ is a finite set.
**Given is a promise:** There exists a subgroup $G_0 \leq G$ such that $f$ is constant and distinct on the cossets of $G$ defined by $G_0$.

**Task:** Find a generating set for $G_0$ (in a polynomial time (in $\lg |G|$) with respect to the number of calls to the oracle for $f$ and in the overall polynomial time).

It has been shown that Hidden subroup problem has efficient solution in the case the group is commutative. The case of non-commutative group is open.

# HIDDEN SUBGROUP PROBLEM

Another famous Shor's algorithm is the one to compute discrete logarithm for modular computation - also the problem for which we do not know an efficient classical algorithm.

These two problems, and many other including those discussed in previous lecture, deal with a special clase of so called **Hidden subgroup problems**.

**Given is:** An (efficiently computable) function $f : G \to R$, where $G$ is a group and $R$ is a finite set.
**Given is a promise:** There exists a subgroup $G_0 \leq G$ such that $f$ is constant and distinct on the cossets of $G$ defined by $G_0$.

**Task:** Find a generating set for $G_0$ (in a polynomial time (in $\lg |G|$) with respect to the number of calls to the oracle for $f$ and in the overall polynomial time).

It has been shown that Hidden subroup problem has efficient solution in the case the group is commutative. The case of non-commutative group is open. Positive solution is shown to be very unlikely.

# QUANTUM SEARCHING
# in
# UNORDERED SETS

# GROVERs SEARCH PROBLEM

**Problem - a popular formulation:** *In an unsorted database of $N$ items there is exactly one, $x_0$, satisfying an easy to verify condition $P$. Find $x_0$.*

# GROVERs SEARCH PROBLEM

**Problem - a popular formulation:** *In an unsorted database of $N$ items there is exactly one, $x_0$, satisfying an easy to verify condition $P$. Find $x_0$.*

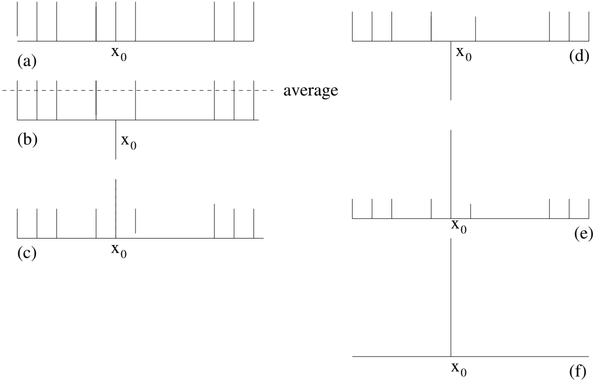
**Classical algorithms** need in average $\frac{N}{2}$ checks.

# GROVERs SEARCH PROBLEM

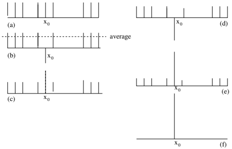
**Problem - a popular formulation:** *In an unsorted database of $N$ items there is exactly one, $x_0$, satisfying an easy to verify condition $P$. Find $x_0$.*

**Classical algorithms** need in average $\frac{N}{2}$ checks.
**Grover's quantum algorithm** exists that needs $\mathcal{O}(\sqrt{N})$ steps.

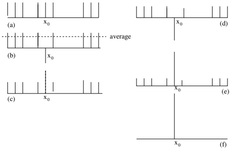
Here is the basic idea of the algorithm - "cooking" a solution.



# COOKING SOLUTION bY GROVER ALGORITHM

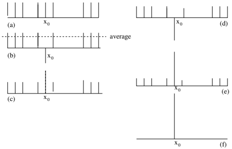


# COOKING SOLUTION bY GROVER ALGORITHM



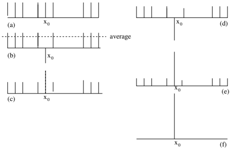

The figure above demonstrates some steps of the Grover algorithm.

# COOKING SOLUTION bY GROVER ALGORITHM




The figure above demonstrates some steps of the Grover algorithm.

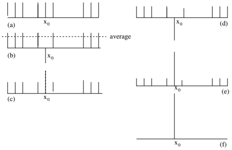
- Starting state, Figure (a), is equally weighted superposition of all basis states. State $|x_0\rangle$ is the one with $f(x_0) = 1$.




The figure above demonstrates some steps of the Grover algorithm.

- Starting state, Figure (a), is equally weighted superposition of all basis states. State $|x_0\rangle$ is the one with $f(x_0) = 1$.
- Next step, Figure (b), is the state obtained by multiplying with $-1$ the amplitude of the state $|x_0\rangle$.

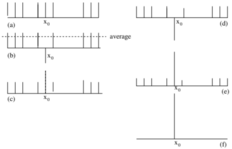
# COOKING SOLUTION bY GROVER ALGORITHM




The figure above demonstrates some steps of the Grover algorithm.

- Starting state, Figure (a), is equally weighted superposition of all basis states. State $|x_0\rangle$ is the one with $f(x_0) = 1$.
- Next step, Figure (b), is the state obtained by multiplying with $-1$ the amplitude of the state $|x_0\rangle$.
- Figure (c) shows the state after inversion over the average - the amplitude at $|x_0\rangle$ is increased and amplitudes other basis states are decreased.

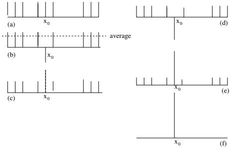
# COOKING SOLUTION bY GROVER ALGORITHM




The figure above demonstrates some steps of the Grover algorithm.

- Starting state, Figure (a), is equally weighted superposition of all basis states. State $|x_0\rangle$ is the one with $f(x_0) = 1$.
- Next step, Figure (b), is the state obtained by multiplying with $-1$ the amplitude of the state $|x_0\rangle$.
- Figure (c) shows the state after inversion over the average - the amplitude at $|x_0\rangle$ is increased and amplitudes other basis states are decreased.
- As the next step, Figure (d), depicts situation that amplitude at the basis state $|x_0\rangle$ is negated and the next step.

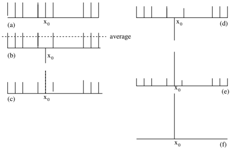
# COOKING SOLUTION bY GROVER ALGORITHM




The figure above demonstrates some steps of the Grover algorithm.

- Starting state, Figure (a), is equally weighted superposition of all basis states. State $|x_0\rangle$ is the one with $f(x_0) = 1$.
- Next step, Figure (b), is the state obtained by multiplying with $-1$ the amplitude of the state $|x_0\rangle$.
- Figure (c) shows the state after inversion over the average - the amplitude at $|x_0\rangle$ is increased and amplitudes other basis states are decreased.
- As the next step, Figure (d), depicts situation that amplitude at the basis state $|x_0\rangle$ is negated and the next step.
- Figure (e), shows the result after another inversion about the average. In case this process iterates for a proper number of steps we get that the amplitude at state $|x_0\rangle$ is (almost) 1 and amplitudes at other states are (almost) 0.

# COOKING SOLUTION bY GROVER ALGORITHM




The figure above demonstrates some steps of the Grover algorithm.

- Starting state, Figure (a), is equally weighted superposition of all basis states. State $|x_0\rangle$ is the one with $f(x_0) = 1$.
- Next step, Figure (b), is the state obtained by multiplying with $-1$ the amplitude of the state $|x_0\rangle$.
- Figure (c) shows the state after inversion over the average - the amplitude at $|x_0\rangle$ is increased and amplitudes other basis states are decreased.
- As the next step, Figure (d), depicts situation that amplitude at the basis state $|x_0\rangle$ is negated and the next step.
- Figure (e), shows the result after another inversion about the average. In case this process iterates for a proper number of steps we get that the amplitude at state $|x_0\rangle$ is (almost) 1 and amplitudes at other states are (almost) 0. A measurement in such a situation produces $x_0$ as the classical outcome.

# INVERSION ABOUT AVERAGE - TOOL

# INVERSION ABOUT AVERAGE - TOOL

Inversion about the average is the unitary transformation

$$D_n : \sum_{i=0}^{2^n-1} a_i|\phi_i\rangle \to \sum_{i=0}^{2^n-1} (2E - a_i)|\phi_i\rangle,$$

where $E$ is the average of $\{a_i \,|\, 0 \leq i < 2^n\}$.

# INVERSION ABOUT AVERAGE - TOOL

Inversion about the average is the unitary transformation

$$D_n : \sum_{i=0}^{2^n-1} a_i|\phi_i\rangle \to \sum_{i=0}^{2^n-1} (2E - a_i)|\phi_i\rangle,$$

where $E$ is the average of $\{a_i \,|\, 0 \le i < 2^n\}$. This important transformation can be performed by the matrix

$$-H_n V_0^n H_n = D_n = \begin{pmatrix} -1 + \frac{2}{2^n} & \frac{2}{2^n} & \dots & \frac{2}{2^n} \\ \frac{2}{2^n} & -1 + \frac{2}{2^n} & \ddots & \frac{2}{2^n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{2}{2^n} & \frac{2}{2^n} & \dots & -1 + \frac{2}{2^n} \end{pmatrix}.$$

# INVERSION ABOUT AVERAGE - TOOL

Inversion about the average is the unitary transformation

$$D_n : \sum_{i=0}^{2^n-1} a_i|\phi_i\rangle \to \sum_{i=0}^{2^n-1} (2E - a_i)|\phi_i\rangle,$$

where $E$ is the average of $\{a_i \,|\, 0 \leq i < 2^n\}$. This important transformation can be performed by the matrix

$$-H_n V_0^n H_n = D_n = \begin{pmatrix} -1+\frac{2}{2^n} & \frac{2}{2^n} & \dots & \frac{2}{2^n} \\ \frac{2}{2^n} & -1+\frac{2}{2^n} & \ddots & \frac{2}{2^n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{2}{2^n} & \frac{2}{2^n} & \dots & -1+\frac{2}{2^n} \end{pmatrix}.$$

The matrix $D_n$ is clearly unitary and it can be shown to have the form $D_n = -H_n V_0^n H_n$, where

$$V_0^n[i,j] = 0 \text{ if } i \neq j,\, V_0^n[1,1] = -1 \text{ and } V_0^n[i,i] = 1 \text{ if } 1 < i \leq n.$$

# APPLICATION of the INVERSION ABOUT AVERAGE

# APPLICATION of the INVERSION ABOUT AVERAGE

Let us consider again the unitary transformation

$$D_n : \sum_{i=0}^{2^n-1} a_i|\phi_i\rangle \to \sum_{i=0}^{2^n-1} (2E - a_i)|\phi_i\rangle,$$

# APPLICATION of the INVERSION ABOUT AVERAGE

Let us consider again the unitary transformation

$$D_n : \sum_{i=0}^{2^n-1} a_i|\phi_i\rangle \to \sum_{i=0}^{2^n-1}(2E - a_i)|\phi_i\rangle,$$

and the following example:

**Example:** Let $a_i = a$ if $i \neq x_0$ and $a_{x_0} = -a$. Then

$$E = a - \frac{2}{2^n}a$$

$$2E - a_i = \begin{cases} a - \frac{4}{2^n}a \text{ if } i \neq x_0 \\ 2E - a_{x_0} = 3a - \frac{4}{2^n}a; \text{ otherwise} \end{cases}$$

# APPLICATION of the INVERSION ABOUT AVERAGE

Let us consider again the unitary transformation

$$D_n : \sum_{i=0}^{2^n-1} a_i|\phi_i\rangle \to \sum_{i=0}^{2^n-1}(2E - a_i)|\phi_i\rangle,$$

and the following example:
**Example:** Let $a_i = a$ if $i \neq x_0$ and $a_{x_0} = -a$. Then

$$E = a - \frac{2}{2^n}a$$

$$2E - a_i = \begin{cases} a - \frac{4}{2^n}a \text{ if } i \neq x_0 \\ 2E - a_{x_0} = 3a - \frac{4}{2^n}a; \text{ otherwise} \end{cases}$$

Therefore, the value of $2E - a_i$ is smaller than $a$ if $i \neq i_0$, and increases otherwise - if $i = i_0$.

# GROVERs ALGORITHM

Start in the state

$$|\phi\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle$$

# GROVERs ALGORITHM

Start in the state

$$|\phi\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle$$

and iterate $\lfloor \frac{\pi}{4}\sqrt{2^n} \rfloor$ times the transformation (so called Grover's iterate):

$$-\underbrace{H_n V_0^n H_n V_f}\,|\phi\rangle \rightarrow |\phi\rangle.$$

Afterwords, measure the register to get some $x_1$ (hopefully $x_0$) and check whether $f(x_1) = 1$. If not, repeat the procedure.

It has been shown that the above algorithm is optimal for finding the solution with probability $> \frac{1}{2}$.

In the case that there are $t$ solutions, repeat the above iteration

$$\left\lfloor \frac{\pi}{4}\sqrt{\frac{2^n}{t}} \right\rfloor \quad \text{times.}$$

# ANALYSIS of THE GROVER ALORITHM

# ANALYSIS of THE GROVER ALORITHM

Denote

$$X_1 = \{x \mid f(x) = 1\} \qquad X_0 = \{x \mid f(x) = 0\}$$

# ANALYSIS of THE GROVER ALORITHM

Denote

$$X_1 = \{x \mid f(x) = 1\} \quad X_0 = \{x \mid f(x) = 0\}$$

and denote the state after $j$th iteration of Grover's iterate $-H_n V_0^n H_n V_f$

# ANALYSIS of THE GROVER ALORITHM

Denote

$$X_1 = \{x \,|\, f(x) = 1\} \qquad X_0 = \{x \,|\, f(x) = 0\}$$

and denote the state after $j$th iteration of Grover's iterate $-H_n V_0^n H_n V_f$ as

$$|\phi_j\rangle = k_j \sum_{x \in X_1} |x\rangle + l_j \sum_{x \in X_0} |x\rangle$$

with

$$k_0 = \frac{1}{\sqrt{2^n}} = l_0.$$

# ANALYSIS of THE GROVER ALORITHM

Denote

$$X_1 = \{x \mid f(x) = 1\} \quad X_0 = \{x \mid f(x) = 0\}$$

and denote the state after $j$th iteration of Grover's iterate $-H_n V_0^n H_n V_f$ as

$$|\phi_j\rangle = k_j \sum_{x \in X_1} |x\rangle + l_j \sum_{x \in X_0} |x\rangle$$

with

$$k_0 = \frac{1}{\sqrt{2^n}} = l_0.$$

Since

$$|\phi_{j+1}\rangle = -H_n V_0^n H_n V_f |\phi_j\rangle,$$

# ANALYSIS of THE GROVER ALORITHM

Denote

$$X_1 = \{x \mid f(x) = 1\} \quad X_0 = \{x \mid f(x) = 0\}$$

and denote the state after $j$th iteration of Grover's iterate $-H_n V_0^n H_n V_f$ as

$$|\phi_j\rangle = k_j \sum_{x \in X_1} |x\rangle + l_j \sum_{x \in X_0} |x\rangle$$

with

$$k_0 = \frac{1}{\sqrt{2^n}} = l_0.$$

Since

$$|\phi_{j+1}\rangle = -H_n V_0^n H_n V_f |\phi_j\rangle,$$

it holds

$$k_{j+1} = \frac{2^n - 2t}{2^n} k_j + \frac{2(2^n - t)}{2^n} l_j, \quad l_{j+1} = \frac{2^n - 2t}{2^n} l_j - \frac{2t}{2^n} k_j$$

# ANALYSIS of THE GROVER ALORITHM

Denote

$$X_1 = \{x \mid f(x) = 1\} \quad X_0 = \{x \mid f(x) = 0\}$$

and denote the state after $j$th iteration of Grover's iterate $-H_n V_0^n H_n V_f$ as

$$|\phi_j\rangle = k_j \sum_{x \in X_1} |x\rangle + l_j \sum_{x \in X_0} |x\rangle$$

with

$$k_0 = \frac{1}{\sqrt{2^n}} = l_0.$$

Since

$$|\phi_{j+1}\rangle = -H_n V_0^n H_n V_f |\phi_j\rangle,$$

it holds

$$k_{j+1} = \frac{2^n - 2t}{2^n} k_j + \frac{2(2^n - t)}{2^n} l_j, \quad l_{j+1} = \frac{2^n - 2t}{2^n} l_j - \frac{2t}{2^n} k_j$$

what yields

$$k_j = \frac{1}{\sqrt{t}} \sin((2j+1)\theta)$$

$$l_j = \frac{1}{\sqrt{2^n - t}} \cos((2j+1)\theta)$$

Recurrence relations have therefore as the solution

$$k_j = \frac{1}{\sqrt{t}} \sin((2j+1)\theta), \quad l_j = \frac{1}{\sqrt{2^n - t}} \cos((2j+1)\theta)$$

where

$$\sin^2\theta = \frac{t}{2^n}.$$

The aim now is to find such an $j$ which maximizes $k_j$ and minimizes $l_j$.

Recurrence relations have therefore as the solution

$$k_j = \frac{1}{\sqrt{t}} \sin((2j+1)\theta), \quad l_j = \frac{1}{\sqrt{2^n - t}} \cos((2j+1)\theta)$$

where

$$\sin^2 \theta = \frac{t}{2^n}.$$

The aim now is to find such an $j$ which maximizes $k_j$ and minimizes $l_j$. For doing that we need to take $j$ such that $\cos((2j+1)\theta) = 0$, that is $(2j+1)\theta = (2m+1)\frac{\pi}{2}$.

Recurrence relations have therefore as the solution

$$k_j = \frac{1}{\sqrt{t}} \sin((2j+1)\theta), \quad l_j = \frac{1}{\sqrt{2^n - t}} \cos((2j+1)\theta)$$

where

$$\sin^2\theta = \frac{t}{2^n}.$$

The aim now is to find such an $j$ which maximizes $k_j$ and minimizes $l_j$. For doing that we need to take $j$ such that $\cos((2j+1)\theta) = 0$, that is $(2j+1)\theta = (2m+1)\frac{\pi}{2}$.

Hence

$$j = \frac{\pi}{4\theta} - \frac{1}{2} + \frac{m\pi}{2\theta}$$

what yields

$$j_0 = \lceil \frac{\pi}{4\theta} \rceil,$$

Recurrence relations have therefore as the solution

$$k_j = \frac{1}{\sqrt{t}} \sin((2j+1)\theta), \quad l_j = \frac{1}{\sqrt{2^n - t}} \cos((2j+1)\theta)$$

where

$$\sin^2 \theta = \frac{t}{2^n}.$$

The aim now is to find such an $j$ which maximizes $k_j$ and minimizes $l_j$. For doing that we need to take $j$ such that $\cos((2j+1)\theta) = 0$, that is $(2j+1)\theta = (2m+1)\frac{\pi}{2}$.

Hence

$$j = \frac{\pi}{4\theta} - \frac{1}{2} + \frac{m\pi}{2\theta}$$

what yields

$$j_0 = \lceil \frac{\pi}{4\theta} \rceil,$$

and because $\sin^2 \theta = \frac{t}{2^n}$ we have

$$0 \leq \sin \theta \leq \sqrt{\frac{t}{2^n}}$$

Recurrence relations have therefore as the solution

$$k_j = \frac{1}{\sqrt{t}} \sin((2j+1)\theta), \quad l_j = \frac{1}{\sqrt{2^n - t}} \cos((2j+1)\theta)$$

where

$$\sin^2\theta = \frac{t}{2^n}.$$

The aim now is to find such an $j$ which maximizes $k_j$ and minimizes $l_j$. For doing that we need to take $j$ such that $\cos((2j+1)\theta) = 0$, that is $(2j+1)\theta = (2m+1)\frac{\pi}{2}$.

Hence

$$j = \frac{\pi}{4\theta} - \frac{1}{2} + \frac{m\pi}{2\theta}$$

what yields

$$j_0 = \lceil \frac{\pi}{4\theta} \rceil,$$

and because $\sin^2\theta = \frac{t}{2^n}$ we have

$$0 \leq \sin\theta \leq \sqrt{\frac{t}{2^n}}$$

and therefore

$$j_0 = \mathcal{O}\left(\sqrt{\frac{2^n}{t}}\right).$$

# QUANTUM SEARCH in ORDERED LISTS

A related problem to that of a search in an unordered list is a search in an ordered list of $n$ items.

- The best upper bound known today is $\frac{3}{4}\lg n$.
- The best lower bound known today is $\frac{1}{12}\lg n - \mathcal{O}(1)$.

# EFFICIENCY of GROVER's SEARCH

# EFFICIENCY of GROVER's SEARCH

There are at least four different proofs that Grover's search is asymptotically optimal.

# EFFICIENCY of GROVER's SEARCH

There are at least four different proofs that Grover's search is asymptotically optimal.

Quite a bit is known about the relation between the error $\varepsilon$ and the number $T$ of queries when searching an unordered list of $n$ elements.

# EFFICIENCY of GROVER's SEARCH

There are at least four different proofs that Grover's search is asymptotically optimal.

Quite a bit is known about the relation between the error $\varepsilon$ and the number $T$ of queries when searching an unordered list of $n$ elements.

- $\varepsilon$ can be an arbitrary small constant if $\mathcal{O}(\sqrt{n})$ queries are used, but not when $o(\sqrt{n})$ queries are used.

# EFFICIENCY of GROVER's SEARCH

There are at least four different proofs that Grover's search is asymptotically optimal.

Quite a bit is known about the relation between the error $\varepsilon$ and the number $T$ of queries when searching an unordered list of $n$ elements.

- $\varepsilon$ can be an arbitrary small constant if $\mathcal{O}(\sqrt{n})$ queries are used, but not when $o(\sqrt{n})$ queries are used.
- $\varepsilon$ can be at most $\frac{1}{2^{n^\alpha}}$ using $\mathcal{O}(n^{0.5+\alpha})$ queries.

# EFFICIENCY of GROVER's SEARCH

There are at least four different proofs that Grover's search is asymptotically optimal.

Quite a bit is known about the relation between the error $\varepsilon$ and the number $T$ of queries when searching an unordered list of $n$ elements.

- $\varepsilon$ can be an arbitrary small constant if $\mathcal{O}(\sqrt{n})$ queries are used, but not when $o(\sqrt{n})$ queries are used.
- $\varepsilon$ can be at most $\frac{1}{2^{n^\alpha}}$ using $\mathcal{O}(n^{0.5+\alpha})$ queries.
- To achieve no error ($\varepsilon = 0$), $\theta(n)$ queries are needed.

# APPLICATIONS of GROVER's SEARCH

# APPLICATIONS of GROVER's SEARCH

There is a variety of applications of Grover's search algorithm. Let us mention some of them.

# APPLICATIONS of GROVER's SEARCH

There is a variety of applications of Grover's search algorithm. Let us mention some of them.

- **Extremes of functions computation** (minimum, maximum).

# APPLICATIONS of GROVER's SEARCH

There is a variety of applications of Grover's search algorithm. Let us mention some of them.

- **Extremes of functions computation** (minimum, maximum).
- **Collision problem** Task is to find, for a given black-box function $f : X \to Y$, two different $x \neq y$ such that $f(x) = f(y)$, given a promise that such a pair exist.

# APPLICATIONS of GROVER's SEARCH

There is a variety of applications of Grover's search algorithm. Let us mention some of them.

- **Extremes of functions computation** (minimum, maximum).
- **Collision problem** Task is to find, for a given black-box function $f : X \to Y$, two different $x \neq y$ such that $f(x) = f(y)$, given a promise that such a pair exist.

  On a more general level an analogical problem deals with the so-called **$r$-to-one functions** every element of their image has exactly $r$ pre-images.

# APPLICATIONS of GROVER's SEARCH

There is a variety of applications of Grover's search algorithm. Let us mention some of them.

- **Extremes of functions computation** (minimum, maximum).
- **Collision problem** Task is to find, for a given black-box function $f : X \rightarrow Y$, two different $x \neq y$ such that $f(x) = f(y)$, given a promise that such a pair exist.

  On a more general level an analogical problem deals with the so-called ***r*-to-one functions** every element of their image has exactly $r$ pre-images. It has been shown that there is a quantum algorithm to solve collision problem for $r$-to-one functions in quantum time $\mathcal{O}((n/r)^{1/3})$. It has been shown in 2003 by Shi that the above upper bound cannot be asymptotically improved.

# APPLICATIONS of GROVER's SEARCH

There is a variety of applications of Grover's search algorithm. Let us mention some of them.

- **Extremes of functions computation** (minimum, maximum).
- **Collision problem** Task is to find, for a given black-box function $f : X \to Y$, two different $x \neq y$ such that $f(x) = f(y)$, given a promise that such a pair exist.

  On a more general level an analogical problem deals with the so-called ***r*-to-one functions** every element of their image has exactly $r$ pre-images. It has been shown that there is a quantum algorithm to solve collision problem for $r$-to-one functions in quantum time $\mathcal{O}((n/r)^{1/3})$. It has been shown in 2003 by Shi that the above upper bound cannot be asymptotically improved.
- **Verification of predicate calculus formulas.** Grover's search algorithm can be seen as a method to verify formulas

  $$\exists x P(x),$$

  where $P$ is a black-box predicate.

# APPLICATIONS of GROVER's SEARCH

There is a variety of applications of Grover's search algorithm. Let us mention some of them.

- **Extremes of functions computation** (minimum, maximum).
- **Collision problem** Task is to find, for a given black-box function $f : X \rightarrow Y$, two different $x \neq y$ such that $f(x) = f(y)$, given a promise that such a pair exist.

  On a more general level an analogical problem deals with the so-called ***r*-to-one functions** every element of their image has exactly $r$ pre-images. It has been shown that there is a quantum algorithm to solve collision problem for $r$-to-one functions in quantum time $\mathcal{O}((n/r)^{1/3})$. It has been shown in 2003 by Shi that the above upper bound cannot be asymptotically improved.
- **Verification of predicate calculus formulas.** Grover's search algorithm can be seen as a method to verify formulas

  $$\exists x P(x),$$

  where $P$ is a black-box predicate.
  It has been shown that also more generalized formulas of the type

  $$\forall x_1 \exists y_1 \forall x_2 \exists y_2 \ldots \forall x_k \exists y_k P(x_1, y_1, x_2, y_2, \ldots, x_k, y_k)$$

# QUANTUM MINIMUM FINDING ALGORITHM

# QUANTUM MINIMUM FINDING ALGORITHM

**Problem:** Let $s = s_1, s_2, \ldots, s_n$ be an unsorted sequence of distinct elements. Find an $m$ such that $s_m$ is minimal.

# QUANTUM MINIMUM FINDING ALGORITHM

**Problem:** Let $s = s_1, s_2, \ldots, s_n$ be an unsorted sequence of distinct elements. Find an $m$ such that $s_m$ is minimal.

Classical search algorithm needs $\theta(n)$ comparisons.

# QUANTUM MINIMUM FINDING ALGORITHM

**Problem:** Let $s = s_1, s_2, \ldots, s_n$ be an unsorted sequence of distinct elements. Find an $m$ such that $s_m$ is minimal.

Classical search algorithm needs $\theta(n)$ comparisons.

QUANTUM SEARCH ALGORITHM

# QUANTUM MINIMUM FINDING ALGORITHM

**Problem:** Let $s = s_1, s_2, \ldots, s_n$ be an unsorted sequence of distinct elements. Find an $m$ such that $s_m$ is minimal.

Classical search algorithm needs $\theta(n)$ comparisons.

## QUANTUM SEARCH ALGORITHM

1. Choose as a first "threshold" a random $y \in \{1, \ldots, n\}$.

# QUANTUM MINIMUM FINDING ALGORITHM

**Problem:** Let $s = s_1, s_2, \ldots, s_n$ be an unsorted sequence of distinct elements. Find an $m$ such that $s_m$ is minimal.

Classical search algorithm needs $\theta(n)$ comparisons.

## QUANTUM SEARCH ALGORITHM

1. Choose as a first "threshold" a random $y \in \{1, \ldots, n\}$.
2. Repeat the following three steps until the total running time is more than $22.5\sqrt{n} + 1.4\lg^2 n$.

# QUANTUM MINIMUM FINDING ALGORITHM

**Problem:** Let $s = s_1, s_2, \ldots, s_n$ be an unsorted sequence of distinct elements. Find an $m$ such that $s_m$ is minimal.

Classical search algorithm needs $\theta(n)$ comparisons.

## QUANTUM SEARCH ALGORITHM

1. Choose as a first "threshold" a random $y \in \{1, \ldots, n\}$.
2. Repeat the following three steps until the total running time is more than $22.5\sqrt{n} + 1.4 \lg^2 n$.
   1. Initialize
   $$|\psi_0\rangle = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} |i\rangle|y\rangle$$
   and consider an index $i$ as **marked** if $s_i < s_y$.

# QUANTUM MINIMUM FINDING ALGORITHM

**Problem:** Let $s = s_1, s_2, \ldots, s_n$ be an unsorted sequence of distinct elements. Find an $m$ such that $s_m$ is minimal.

Classical search algorithm needs $\theta(n)$ comparisons.

## QUANTUM SEARCH ALGORITHM

1. Choose as a first "threshold" a random $y \in \{1, \ldots, n\}$.
2. Repeat the following three steps until the total running time is more than $22.5\sqrt{n} + 1.4 \lg^2 n$.
    1. Initialize
    $$|\psi_0\rangle = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} |i\rangle|y\rangle$$
    and consider an index $i$ as **marked** if $s_i < s_y$.
    2. Apply Grover search to the first register to find an marked element.

# QUANTUM MINIMUM FINDING ALGORITHM

**Problem:** Let $s = s_1, s_2, \ldots, s_n$ be an unsorted sequence of distinct elements. Find an $m$ such that $s_m$ is minimal.

Classical search algorithm needs $\theta(n)$ comparisons.

## QUANTUM SEARCH ALGORITHM

1. Choose as a first "threshold" a random $y \in \{1, \ldots, n\}$.
2. Repeat the following three steps until the total running time is more than $22.5\sqrt{n} + 1.4 \lg^2 n$.
   1. Initialize
      $$|\psi_0\rangle = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} |i\rangle|y\rangle$$
      and consider an index $i$ as **marked** if $s_i < s_y$.
   2. Apply Grover search to the first register to find an marked element.
   3. Measure the first register. If $y'$ is the outcome and $s_{y'} < s_y$, take as a new threshold the index $y'$.
3. Return as the output the last threshold $y$.

# SHOR's DISCRETE LOGARITHM ALGORITHM

# SHOR's DISCRETE LOGARITHM ALGORITHM

Shor's quantum algorithm for discrete logarithm will be again presented only for a special case.

# SHOR's DISCRETE LOGARITHM ALGORITHM

Shor's quantum algorithm for discrete logarithm will be again presented only for a special case.

The task is to determine an $r$ such that $g^r \equiv x (\bmod\ p)$ given a prime $p$, a generator $g$ of the group $\mathbf{Z}_p^*$ and an $0 < x < p$.

# SHOR's DISCRETE LOGARITHM ALGORITHM

Shor's quantum algorithm for discrete logarithm will be again presented only for a special case.

The task is to determine an $r$ such that $g^r \equiv x (\bmod\ p)$ given a prime $p$, a generator $g$ of the group $\mathbf{Z}_p^*$ and an $0 < x < p$. The special case we consider is that $p-1$ is smooth.

Using $\mathrm{QFT}_{p-1}$ twice, on the third and fourth sub-register of the state $|x, g, \mathbf{0}, \mathbf{0}, \mathbf{0}\rangle$, we get

$$|\phi\rangle = \frac{1}{p-1}\sum_{a=0}^{p-2}\sum_{b=0}^{p-2}|x, g, a, b, \mathbf{0}\rangle,$$

a uniform distribution of all pairs $(a, b)$, $0 \leq a, b \leq p-2$.

# SHOR's DISCRETE LOGARITHM ALGORITHM

Shor's quantum algorithm for discrete logarithm will be again presented only for a special case.

The task is to determine an $r$ such that $g^r \equiv x (\bmod\ p)$ given a prime $p$, a generator $g$ of the group $\mathbf{Z}_p^*$ and an $0 < x < p$. The special case we consider is that $p-1$ is smooth.

Using $\mathrm{QFT}_{p-1}$ twice, on the third and fourth sub-register of the state $|x, g, \mathbf{0}, \mathbf{0}, \mathbf{0}\rangle$, we get

$$|\phi\rangle = \frac{1}{p-1} \sum_{a=0}^{p-2} \sum_{b=0}^{p-2} |x, g, a, b, \mathbf{0}\rangle,$$

a uniform distribution of all pairs $(a, b)$, $0 \leq a, b \leq p-2$. By applying to $|\phi\rangle$ the unitary mapping

$$(x, g, a, b, \mathbf{0}) \rightarrow (x, g, a, b, g^a x^{-b} \bmod p)$$

# SHOR's DISCRETE LOGARITHM ALGORITHM

Shor's quantum algorithm for discrete logarithm will be again presented only for a special case.

The task is to determine an $r$ such that $g^r \equiv x \pmod{p}$ given a prime $p$, a generator $g$ of the group $\mathbf{Z}_p^*$ and an $0 < x < p$. The special case we consider is that $p-1$ is smooth.

Using $\mathrm{QFT}_{p-1}$ twice, on the third and fourth sub-register of the state $|x, g, \mathbf{0}, \mathbf{0}, \mathbf{0}\rangle$, we get

$$|\phi\rangle = \frac{1}{p-1} \sum_{a=0}^{p-2} \sum_{b=0}^{p-2} |x, g, a, b, \mathbf{0}\rangle,$$

a uniform distribution of all pairs $(a, b)$, $0 \leq a, b \leq p-2$. By applying to $|\phi\rangle$ the unitary mapping

$$(x, g, a, b, \mathbf{0}) \rightarrow (x, g, a, b, g^a x^{-b} \bmod p)$$

we get

$$|\phi'\rangle = \frac{1}{p-1} \sum_{a=0}^{p-2} \sum_{b=0}^{p-2} |x, g, a, b, g^a x^{-b} \bmod p\rangle.$$

Since parameters $x, g$ will not be changed in the following computations we will not write them explicitly in what follows.

Since parameters $x, g$ will not be changed in the following computations we will not write them explicitly in what follows.

Therefore the state we got at the previous computation is:

$$|\phi'\rangle = \frac{1}{p-1}\sum_{a=0}^{p-2}\sum_{b=0}^{p-2}|a, b, g^a x^{-b} \bmod p\rangle.$$

Since parameters $x, g$ will not be changed in the following computations we will not write them explicitly in what follows.

Therefore the state we got at the previous computation is:

$$|\phi'\rangle = \frac{1}{p-1} \sum_{a=0}^{p-2} \sum_{b=0}^{p-2} |a, b, g^a x^{-b} \bmod p\rangle.$$

As the next step we apply $\mathrm{QFT}_{p-1}$ on $|\phi'\rangle$ twice, once to map each $a$ to each $c$ with the amplitude $\frac{1}{\sqrt{p-1}} e^{2\pi i ac/(p-1)}$ and once to map each $b$ to each $d$ with amplitude $\frac{1}{\sqrt{p-1}} e^{2\pi i bd/(p-1)}$.

Since parameters $x, g$ will not be changed in the following computations we will not write them explicitly in what follows.

Therefore the state we got at the previous computation is:

$$|\phi'\rangle = \frac{1}{p-1}\sum_{a=0}^{p-2}\sum_{b=0}^{p-2}|a, b, g^a x^{-b} \bmod p\rangle.$$

As the next step we apply $\mathrm{QFT}_{p-1}$ on $|\phi'\rangle$ twice, once to map each $a$ to each $c$ with the amplitude $\frac{1}{\sqrt{p-1}}e^{2\pi iac/(p-1)}$ and once to map each $b$ to each $d$ with amplitude $\frac{1}{\sqrt{p-1}}e^{2\pi ibd/(p-1)}$.

The resulting state will be:

$$|\phi_1\rangle = \frac{1}{(p-1)^2}\sum_{a,b,c,d=0}^{p-2} e^{\frac{2\pi i}{p-1}(ac+bd)}|c, d, g^a x^{-b} \bmod p\rangle.$$

Hence

$$|\phi_1\rangle = \frac{1}{(p-1)^2}\sum_{a,b,c,d=0}^{p-2} e^{\frac{2\pi i}{p-1}(ac+bd)}|c,d,g^a x^{-b} \bmod p\rangle.$$

Hence

$$|\phi_1\rangle = \frac{1}{(p-1)^2} \sum_{a,b,c,d=0}^{p-2} e^{\frac{2\pi i}{p-1}(ac+bd)} |c, d, g^a x^{-b} \bmod p\rangle.$$

Let us now measure the last register and denote by $y$ the value we get.

Hence

$$|\phi_1\rangle = \frac{1}{(p-1)^2} \sum_{a,b,c,d=0}^{p-2} e^{\frac{2\pi i}{p-1}(ac+bd)} |c, d, g^a x^{-b} \bmod p\rangle.$$

Let us now measure the last register and denote by $y$ the value we get.

The state $|\phi_1\rangle$ then collapses into the state (before the normalization)

$$|\phi_2\rangle = \sum_{c,d=0}^{p-1} \alpha(c,d)|c, d, y\rangle,$$

Hence

$$|\phi_1\rangle = \frac{1}{(p-1)^2} \sum_{a,b,c,d=0}^{p-2} e^{\frac{2\pi i}{p-1}(ac+bd)} |c, d, g^a x^{-b} \bmod p\rangle.$$

Let us now measure the last register and denote by $y$ the value we get.

The state $|\phi_1\rangle$ then collapses into the state (before the normalization)

$$|\phi_2\rangle = \sum_{c,d=0}^{p-1} \alpha(c,d)|c, d, y\rangle,$$

where

$$\alpha(c,d) = \frac{1}{(p-1)^2} \sum_{\{(a,b)\,|\, y=g^a x^{-b} \bmod p\}} e^{\frac{2\pi i}{p-1}(ac+bd)}.$$

Hence

$$|\phi_1\rangle = \frac{1}{(p-1)^2} \sum_{a,b,c,d=0}^{p-2} e^{\frac{2\pi i}{p-1}(ac+bd)} |c, d, g^a x^{-b} \bmod p\rangle.$$

Let us now measure the last register and denote by $y$ the value we get.

The state $|\phi_1\rangle$ then collapses into the state (before the normalization)

$$|\phi_2\rangle = \sum_{c,d=0}^{p-1} \alpha(c,d)|c, d, y\rangle,$$

where

$$\alpha(c,d) = \frac{1}{(p-1)^2} \sum_{\{(a,b)\,|\,y=g^a x^{-b} \bmod p\}} e^{\frac{2\pi i}{p-1}(ac+bd)}.$$

We now claim that if $y = g^a x^{-b} \bmod p$, then $y = g^k$ for some $k$ such that

$$a - rb \equiv k \pmod{p-1}.$$

Hence

$$|\phi_1\rangle = \frac{1}{(p-1)^2} \sum_{a,b,c,d=0}^{p-2} e^{\frac{2\pi i}{p-1}(ac+bd)}|c, d, g^a x^{-b} \bmod p\rangle.$$

Let us now measure the last register and denote by $y$ the value we get.

The state $|\phi_1\rangle$ then collapses into the state (before the normalization)

$$|\phi_2\rangle = \sum_{c,d=0}^{p-1} \alpha(c,d)|c, d, y\rangle,$$

where

$$\alpha(c,d) = \frac{1}{(p-1)^2} \sum_{\{(a,b)\,|\,y=g^a x^{-b} \bmod p\}} e^{\frac{2\pi i}{p-1}(ac+bd)}.$$

We now claim that if $y = g^a x^{-b} \bmod p$, then $y = g^k$ for some $k$ such that

$$a - rb \equiv k \pmod{p-1}.$$

Indeed,

$$y = g^a x^{-b} \equiv g^a (g^r)^{-b} = g^{a-rb}.$$

Hence

$$|\phi_1\rangle = \frac{1}{(p-1)^2}\sum_{a,b,c,d=0}^{p-2} e^{\frac{2\pi i}{p-1}(ac+bd)}|c,d,g^a x^{-b} \bmod p\rangle.$$

Let us now measure the last register and denote by $y$ the value we get.

The state $|\phi_1\rangle$ then collapses into the state (before the normalization)

$$|\phi_2\rangle = \sum_{c,d=0}^{p-1} \alpha(c,d)|c,d,y\rangle,$$

where

$$\alpha(c,d) = \frac{1}{(p-1)^2}\sum_{\{(a,b)\,|\,y=g^a x^{-b} \bmod p\}} e^{\frac{2\pi i}{p-1}(ac+bd)}.$$

We now claim that if $y = g^a x^{-b} \bmod p$, then $y = g^k$ for some $k$ such that

$$a - rb \equiv k \pmod{p-1}.$$

Indeed,

$$y = g^a x^{-b} \equiv g^a (g^r)^{-b} = g^{a-rb}.$$

and, therefore, if $a - rb \equiv k \pmod{p-1}$, then

$$g^{a-br} \equiv g^k \pmod{p}$$

Therefore

$$\alpha(c,d) = \frac{1}{(p-1)^2} \sum_{\{(a,b)\,|\, a-rb\equiv k \pmod{p-1}\}} e^{\frac{2\pi i}{p-1}(ac+bd)}$$

Therefore

$$\alpha(c,d) = \frac{1}{(p-1)^2} \sum_{\{(a,b)\,|\,a-rb\equiv k \pmod{p-1}\}} e^{\frac{2\pi i}{p-1}(ac+bd)}$$

The probability $Pr$ that, for fixed $c$ and $d$, we get by measurement of $|\phi_2\rangle$ a value $y$ is therefore

$$Pr = \left| \frac{1}{(p-1)^2} \sum_{a,b=0}^{p-2} \{ e^{\frac{2\pi i}{p-1}(ac+bd)} \,|\, a-rb \equiv k \pmod{p-1} \} \right|^2 .$$

Therefore

$$\alpha(c,d) = \frac{1}{(p-1)^2} \sum_{\{(a,b)\,|\,a-rb\equiv k \pmod{p-1}\}} e^{\frac{2\pi i}{p-1}(ac+bd)}$$

The probability $Pr$ that, for fixed $c$ and $d$, we get by measurement of $|\phi_2\rangle$ a value $y$ is therefore

$$Pr = \left| \frac{1}{(p-1)^2} \sum_{a,b=0}^{p-2} \{ e^{\frac{2\pi i}{p-1}(ac+bd)} \,|\, a-rb \equiv k \pmod{p-1} \} \right|^2 .$$

By substituting $a = k + rb + j_b(p-1)$ we get as the probability

$$Pr = \left| \frac{1}{(p-1)^2} \sum_{b=0}^{p-2} e^{\frac{2\pi i}{p-1}(kc+cj_b(p-1)+b(d+rc))} \right|^2 = \left| \frac{1}{(p-1)^2} e^{\frac{2\pi ikc}{p-1}} \sum_{b=0}^{p-2} e^{\frac{2\pi i}{p-1}(b(d+rc))} \right|^2$$

Therefore

$$\alpha(c,d)=\frac{1}{(p-1)^2}\sum_{\{(a,b)\,|\,a-rb\equiv k \pmod{p-1}\}} e^{\frac{2\pi i}{p-1}(ac+bd)}$$

The probability $Pr$ that, for fixed $c$ and $d$, we get by measurement of $|\phi_2\rangle$ a value $y$ is therefore

$$Pr=\left|\frac{1}{(p-1)^2}\sum_{a,b=0}^{p-2}\{e^{\frac{2\pi i}{p-1}(ac+bd)}\,|a-rb\equiv k \pmod{p-1}\}\right|^2.$$

By substituting $a=k+rb+j_b(p-1)$ we get as the probability

$$Pr=\left|\frac{1}{(p-1)^2}\sum_{b=0}^{p-2} e^{\frac{2\pi i}{p-1}(kc+cj_b(p-1)+b(d+rc))}\right|^2=\left|\frac{1}{(p-1)^2}e^{\frac{2\pi ikc}{p-1}}\sum_{b=0}^{p-2} e^{\frac{2\pi i}{p-1}(b(d+rc))}\right|^2$$

what equals

$$Pr=\left|\frac{1}{(p-1)^2}\sum_{b=0}^{p-2} e^{\frac{2\pi i}{p-1}(b(d+rc))}\right|^2=\left|\frac{1}{(p-1)^2}\sum_{b=0}^{p-2}(e^{\frac{2\pi i}{p-1}(d+rc)})^b\right|^2$$

The above probability $Pr$ is therefore 0 if

$$d + rc \not\equiv 0 \bmod (p-1)$$

because, as in the previous algorithm, in such a case the sum in the above expression is over a set of complex numbers equally spaced around the unit circle.

The above probability $Pr$ is therefore 0 if

$$d + rc \not\equiv 0 \bmod (p-1)$$

because, as in the previous algorithm, in such a case the sum in the above expression is over a set of complex numbers equally spaced around the unit circle.

On the other hand, if

$$d \equiv -rc \pmod{p-1},$$

then the above sum does not depend on $b$ and it is equal to

$$(p-1)^{-1} e^{(2\pi ikc/(p-1))}.$$

The above probability $Pr$ is therefore 0 if

$$d + rc \not\equiv 0 \bmod (p-1)$$

because, as in the previous algorithm, in such a case the sum in the above expression is over a set of complex numbers equally spaced around the unit circle.

On the other hand, if

$$d \equiv -rc \pmod{p-1},$$

then the above sum does not depend on $b$ and it is equal to

$$(p-1)^{-1}e^{(2\pi ikc/(p-1))}.$$

The square of its absolute value, the probability, is therefore $\frac{1}{(p-1)^2}$.

The above probability $Pr$ is therefore 0 if

$$d + rc \not\equiv 0 \bmod (p-1)$$

because, as in the previous algorithm, in such a case the sum in the above expression is over a set of complex numbers equally spaced around the unit circle.

On the other hand, if

$$d \equiv -rc \pmod{p-1},$$

then the above sum does not depend on $b$ and it is equal to

$$(p-1)^{-1} e^{(2\pi ikc/(p-1))}.$$

The square of its absolute value, the probability, is therefore $\frac{1}{(p-1)^2}$.

Consequence: the measurements on the first and second register provide a (random) $c < p-1$ and a $d$ such that

$$d \equiv -rc \pmod{p-1}.$$

If $gcd(c, p-1) = 1$, $r$ can now be obtained as a unique solution of the above congruence equation.

The above probability $Pr$ is therefore 0 if

$$d + rc \not\equiv 0 \text{ mod } (p-1)$$

because, as in the previous algorithm, in such a case the sum in the above expression is over a set of complex numbers equally spaced around the unit circle.

On the other hand, if

$$d \equiv -rc \pmod{p-1},$$

then the above sum does not depend on $b$ and it is equal to

$$(p-1)^{-1} e^{(2\pi ikc/(p-1))}.$$

The square of its absolute value, the probability, is therefore $\frac{1}{(p-1)^2}$.

Consequence: the measurements on the first and second register provide a (random) $c < p-1$ and a $d$ such that

$$d \equiv -rc \pmod{p-1}.$$

If $gcd(c, p-1) = 1$, $r$ can now be obtained as a unique solution of the above congruence equation.

The number of computations needed to be performed, in order to get the probability close to 1 for finding $r$, is polynomial in $\lg\lg p$.

# ANOTHE COMMENTS on SHOR's FACTORIZATION ALGORITHM

# ANOTHE COMMENTS on SHOR's FACTORIZATION ALGORITHM

- To factor an integer $n$ Shor's algorithm uses $\mathcal{O}(\lg^3 n)$ steps and success probability is guaranteed to be at least $\Omega(\frac{1}{\lg \lg n})$.

# ANOTHE COMMENTS on SHOR's FACTORIZATION ALGORITHM

- To factor an integer $n$ Shor's algorithm uses $\mathcal{O}(\lg^3 n)$ steps and success probability is guaranteed to be at least $\Omega(\frac{1}{\lg\lg n})$.
- An analysis of Shor's algorithm therefore shows that by running the algorithm $\mathcal{O}(\lg\lg n)$ times, therefore in total in $\mathcal{O}(\lg^3 n \lg\lg n)$ times we have very high success probability.

# ANOTHE COMMENTS on SHOR's FACTORIZATION ALGORITHM

- To factor an integer $n$ Shor's algorithm uses $\mathcal{O}(\lg^3 n)$ steps and success probability is guaranteed to be at least $\Omega(\frac{1}{\lg \lg n})$.
- An analysis of Shor's algorithm therefore shows that by running the algorithm $\mathcal{O}(\lg \lg n)$ times, therefore in total in $\mathcal{O}(\lg^3 n \lg \lg n)$ times we have very high success probability.
- Shor's algorithms make some of the important current cryptosystems, as RSA, ElGamal and so on vulnerable to attacks using quantum computers.

# ANOTHE COMMENTS on SHOR's FACTORIZATION ALGORITHM

- To factor an integer $n$ Shor's algorithm uses $\mathcal{O}(\lg^3 n)$ steps and success probability is guaranteed to be at least $\Omega(\frac{1}{\lg \lg n})$.
- An analysis of Shor's algorithm therefore shows that by running the algorithm $\mathcal{O}(\lg \lg n)$ times, therefore in total in $\mathcal{O}(\lg^3 n \lg \lg n)$ times we have very high success probability.
- Shor's algorithms make some of the important current cryptosystems, as RSA, ElGamal and so on vulnerable to attacks using quantum computers.
- Shor's result have been generalized to show that a large range of cryptosystems, including elliptic curve cryptosystems, would be vulnerable to attacks using quantum computers.

# FOURIER TRANSFORM on ABELIAN GROUPS

We show now basics how the concept of Fourier Transform is defined on any finite Abelian group.

# CHARACTERS on ABELIAN GROUPS

Let $G$ be an Abelian additive group and $|G| = n$. A character $\chi$ of $G$ is any morphism $\chi : G \to \mathbf{C}/0$. That means that for any $g_1, g_2 \in G$ it holds:

$$\chi(g_1 + g_2) = \chi(g_1)\chi(g_2)).$$

# CHARACTERS on ABELIAN GROUPS

Let $G$ be an Abelian additive group and $|G| = n$. A character $\chi$ of $G$ is any morphism $\chi : G \rightarrow \mathbf{C}/0$. That means that for any $g_1, g_2 \in G$ it holds:

$$\chi(g_1 + g_2) = \chi(g_1)\chi(g_2)).$$

This implies that $\chi(0) = 1$ and $1 = \chi(ng) = \chi(g)^n$ for any $g \in G$.

# CHARACTERS on ABELIAN GROUPS

Let $G$ be an Abelian additive group and $|G| = n$. A character $\chi$ of $G$ is any morphism $\chi : G \to \mathbf{C}/0$. That means that for any $g_1, g_2 \in G$ it holds:

$$\chi(g_1 + g_2) = \chi(g_1)\chi(g_2)).$$

This implies that $\chi(0) = 1$ and $1 = \chi(ng) = \chi(g)^n$ for any $g \in G$. Therefore, all values of $\chi$ are $n$th roots of unity.

# CHARACTERS on ABELIAN GROUPS

Let $G$ be an Abelian additive group and $|G| = n$. A character $\chi$ of $G$ is any morphism $\chi : G \rightarrow \mathbf{C}/0$. That means that for any $g_1, g_2 \in G$ it holds:

$$\chi(g_1 + g_2) = \chi(g_1)\chi(g_2)).$$

This implies that $\chi(0) = 1$ and $1 = \chi(ng) = \chi(g)^n$ for any $g \in G$. Therefore, all values of $\chi$ are $n$th roots of unity.

If we define multiplication of characters $\chi_1$ and $\chi_2$ by $\chi_1\chi_2(g) = \chi_1(g)\chi_2(g)$, then characters form so-called dual group $\hat{\mathrm{G}}$.

# CHARACTERS on ABELIAN GROUPS

Let $G$ be an Abelian additive group and $|G| = n$. A character $\chi$ of $G$ is any morphism $\chi : G \rightarrow \mathbf{C}/0$. That means that for any $g_1, g_2 \in G$ it holds:

$$\chi(g_1 + g_2) = \chi(g_1)\chi(g_2)).$$

This implies that $\chi(0) = 1$ and $1 = \chi(ng) = \chi(g)^n$ for any $g \in G$. Therefore, all values of $\chi$ are $n$th roots of unity.

If we define multiplication of characters $\chi_1$ and $\chi_2$ by $\chi_1\chi_2(g) = \chi_1(g)\chi_2(g)$, then characters form so-called dual group $\hat{\mathrm{G}}$. Groups $G$ and $\hat{\mathrm{G}}$ are isomorphic for all Abelian groups $G$.

# CHARACTERS on ABELIAN GROUPS

Let $G$ be an Abelian additive group and $|G| = n$. A character $\chi$ of $G$ is any morphism $\chi : G \to \mathbf{C}/0$. That means that for any $g_1, g_2 \in G$ it holds:

$$\chi(g_1 + g_2) = \chi(g_1)\chi(g_2)).$$

This implies that $\chi(0) = 1$ and $1 = \chi(ng) = \chi(g)^n$ for any $g \in G$. Therefore, all values of $\chi$ are $n$th roots of unity.

If we define multiplication of characters $\chi_1$ and $\chi_2$ by $\chi_1\chi_2(g) = \chi_1(g)\chi_2(g)$, then characters form so-called dual group $\hat{\mathrm{G}}$. Groups $G$ and $\hat{\mathrm{G}}$ are isomorphic for all Abelian groups $G$.

Example 1 Any cyclic group of $n$ elements is isomorphic to the group $\mathbf{Z}_n$ and all its characters have the form, for some $y \in \mathbf{Z}_n$:

$$\chi_y(x) = e^{\frac{2\pi ixy}{n}}.$$

# CHARACTERS on ABELIAN GROUPS

Let $G$ be an Abelian additive group and $|G| = n$. A character $\chi$ of $G$ is any morphism $\chi : G \rightarrow \mathbf{C}/0$. That means that for any $g_1, g_2 \in G$ it holds:

$$\chi(g_1 + g_2) = \chi(g_1)\chi(g_2)).$$

This implies that $\chi(0) = 1$ and $1 = \chi(ng) = \chi(g)^n$ for any $g \in G$. Therefore, all values of $\chi$ are $n$th roots of unity.

If we define multiplication of characters $\chi_1$ and $\chi_2$ by $\chi_1\chi_2(g) = \chi_1(g)\chi_2(g)$, then characters form so-called dual group $\hat{\mathrm{G}}$. Groups $G$ and $\hat{\mathrm{G}}$ are isomorphic for all Abelian groups $G$.

Example 1 Any cyclic group of $n$ elements is isomorphic to the group $\mathbf{Z}_n$ and all its characters have the form, for some $y \in \mathbf{Z}_n$:

$$\chi_y(x) = e^{\frac{2\pi ixy}{n}}.$$

Example 2 In the additive group $\mathbf{F_2}^m$, of all binary strings of length $m$, all characters have the form, for some binary $m$-bit strings $x$ and $y$:

$$\chi_y(x) = (-1)^{x \cdot y},$$

where $x \cdot y = \sum_{i=1}^{m} x_i y_i \bmod 2$

# ORTHOGONALITY of CHARACTERS

Any function $f : G \to \mathbf{C}$ on an Abelian group $G = \{g_1, \dots, g_n\}$ can be specified by the vector $(f(g_1), \dots, f(g_n))$, and if the scalar product of two functions is defined in the standard way as

$$\langle f|g\rangle = \sum_{i=1}^{n} f^*(g_i)h(g_i),$$

then for any characters $\chi_1$ and $\chi_2$ on $G$ it holds

$$\langle \chi_i|\chi_j\rangle = \begin{cases} 0, & \text{if } i \neq j \\ n, & \text{if } i = j \end{cases}$$

# ORTHOGONALITY of CHARACTERS

Any function $f : G \to \mathbf{C}$ on an Abelian group $G = \{g_1, \ldots, g_n\}$ can be specified by the vector $(f(g_1), \ldots, f(g_n))$, and if the scalar product of two functions is defined in the standard way as

$$\langle f|g\rangle = \sum_{i=1}^{n} f^*(g_i)h(g_i),$$

then for any characters $\chi_1$ and $\chi_2$ on $G$ it holds

$$\langle \chi_i|\chi_j\rangle = \begin{cases} 0, \text{ if } i \neq j \\ n, \text{ if } i = j \end{cases}$$

Therefore, the functions $\{B_i = \frac{1}{\sqrt{n}}\chi_i\}$ form an orthonormal basis on the set of all functions $f : G \to \mathbf{C}$.

# FOURIER TRANSFORMS

# FOURIER TRANSFORMS

Any function $f : G \rightarrow \mathbf{C}$ has a unique representation with respect to the basis $\{B_i = \frac{1}{\sqrt{n}}\chi_i\}_{i=1}^n$,

$$f = \hat{\mathrm{f}}_1 B_1 + \ldots + \hat{\mathrm{f}}_n B_n$$

In such a case the function $\hat{\mathrm{f}}$: $G \rightarrow \mathbf{C}$ defined by

$$\hat{\mathrm{f}}(g_i) = \hat{\mathrm{f}}_i$$

is called the Fourier transform of $f$.

# FOURIER TRANSFORMS

Any function $f : G \rightarrow \mathbf{C}$ has a unique representation with respect to the basis $\{B_i = \frac{1}{\sqrt{n}}\chi_i\}_{i=1}^{n}$,

$$f = \hat{\mathrm{f}}_1 B_1 + \ldots + \hat{\mathrm{f}}_n B_n$$

In such a case the function $\hat{\mathrm{f}}$: $G \rightarrow \mathbf{C}$ defined by

$$\hat{\mathrm{f}}(g_i) = \hat{\mathrm{f}}_i$$

is called the Fourier transform of $f$.

Since $\hat{\mathrm{f}}_i = \langle B_i | f \rangle$, we get

$$\hat{\mathrm{f}}(g_i) = \frac{1}{\sqrt{n}} \sum_{k=1}^{n} \chi_i^*(g_k) f(g_k),$$

# FOURIER TRANSFORMS

Any function $f : G \to \mathbf{C}$ has a unique representation with respect to the basis $\{B_i = \frac{1}{\sqrt{n}}\chi_i\}_{i=1}^n$,

$$f = \hat{\mathrm{f}}_1 B_1 + \ldots + \hat{\mathrm{f}}_n B_n$$

In such a case the function $\hat{\mathrm{f}}$: $G \to \mathbf{C}$ defined by

$$\hat{\mathrm{f}}(g_i) = \hat{\mathrm{f}}_i$$

is called the Fourier transform of $f$.

Since $\hat{\mathrm{f}}_i = \langle B_i | f \rangle$, we get

$$\hat{\mathrm{f}}(g_i) = \frac{1}{\sqrt{n}} \sum_{k=1}^{n} \chi_i^*(g_k) f(g_k),$$

Therefore in $\mathbf{Z}_n$ the Fourier transform has the form

$$\hat{\mathrm{f}}(x) = \frac{1}{\sqrt{n}} \sum_{y \in \mathbf{Z}_n} e^{-\frac{2\pi ixy}{n}} f(y)$$

and in $\mathbf{F}_2^m$ the Fourier transform has the form

$$\hat{\mathrm{f}}(x) = \frac{1}{\sqrt{2^m}} \sum_{y \in \mathbf{F}_2^m} (-1)^{x \cdot y} f(y).$$