LECTURE 6
PHASE ESTIMATION, SHOR'S ALGORITHM AND HIDDEN SUBGROUP PROBLEM

INF587 Quantum computer science and applications

Thomas Debris-Alazard

Inria, École Polytechnique

Presentation of Shor's algorithm and hidden Abelian subgroup problem!

It will rely (partly) on:

- phase estimation and consequences: **QFT** over finite Abelian groups and order finding

1. Phase Estimation
2. Application 1: Quantum Fourier Transform on $\mathbb{Z}/N\mathbb{Z}$ and any Finite Abelian Group
3. Application 2: Order Finding
4. Shor's Algorithm
5. Hidden Subgroup Problem (HSP)

PHASE ESTIMATION

Phase estimation:

- **Input:** a unitary **U** with eigenstate $|u\rangle$:

$$\mathsf{U}\,|u\rangle = e^{2i\pi\varphi}\,|u\rangle$$

- **Output:** $\varphi \in [0,1)$, *i.e.*, the knowledge of the associate eigenvalue of $|u\rangle$

$\longrightarrow$ Essential for computing $\mathbf{QFT}_{\mathbb{Z}/N\mathbb{Z}}$ and Shor's algorithm!

Proposition:

We can determine (by using $\mathbf{QFT}_{\mathbb{Z}/2^t\mathbb{Z}}$) the first n bits of φ with probability $1-\varepsilon$ using

$$O(t^2) \text{ elementary gates where } t = n + \left\lceil \log\left(2 + \frac{1}{2\varepsilon}\right)\right\rceil$$

$\longrightarrow$ n bits of φ with probability $1 - e^{-Cn}$ but working in the space of t-qubits with $t = O(n)$

Notation:

Given $j_1, j_2, \ldots, j_m \in \{0,1\}$:

$$0.j_1 j_2 \ldots j_m \overset{\text{def}}{=} \frac{j_1}{2} + \frac{j_2}{4} + \cdots + \frac{j_m}{2^m} = \sum_{i=1}^{m} \frac{j_i}{2^i}$$

Example:

$$0.101 = \frac{1}{2} + \frac{1}{8} = 0.625, \quad 0.111 = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} = 0.875 \quad \text{and} \quad 0.011 = \frac{1}{4} + \frac{1}{8} = 0.325$$

$$2^m\ 0.j_1 j_2 \ldots j_m = 2^{m-1} j_1 + 2^{m-2} j_2 + \cdots + j_m = j_1 \ldots j_m \in [\![0, 2^m - 1]\!]$$

$$\Big(\text{binary representation with } m \text{ bits}\Big)$$

$$2^\ell\ 0.j_1 j_2 \ldots j_m = \underbrace{2^{\ell-1} j_1 + \cdots + j_\ell}_{\in \mathbb{N}} + 0.j_{\ell+1} \ldots j_m$$

$$\longrightarrow \mathrm{e}^{2i\pi 2^\ell 0.j_1 j_2 \ldots j_m} = \mathrm{e}^{2i\pi 0.j_{\ell+1} \ldots j_m}$$

The quantum algorithm to determine the phase starts from ($|u\rangle$ being the eigenstate)

$$\left|0^t\right\rangle |u\rangle$$

$\longrightarrow$ t function of: (i) accuracy and (ii) probability we wish to be successful

Phase estimation, two stages algorithm:

1. Build the following quantum state:

$$\frac{1}{2^{t/2}}\left(|0\rangle + \mathrm{e}^{2i\pi 2^{t-1}\varphi}\,|1\rangle\right) \otimes \left(|0\rangle + \mathrm{e}^{2i\pi 2^{t-2}\varphi}\,|1\rangle\right) \otimes \cdots \otimes \left(|0\rangle + \mathrm{e}^{2i\pi 2^{0}\varphi}\,|1\rangle\right) \otimes |u\rangle$$

2. Apply the $\mathbf{QFT}^{-1}_{\mathbb{Z}/2^t\mathbb{Z}}$ to reach:

$$\approx \left|\lfloor 2^t\varphi \rfloor\right\rangle \otimes |u\rangle = |\varphi_1 \ldots \varphi_t\rangle \otimes |u\rangle$$

Does the first step remind you of something?

FIRST STAGE

The controlled U^{2^j}-unitary:

$$|1\rangle\,|u\rangle \longmapsto |1\rangle\,\mathbf{U}^{2^j}\,|u\rangle = \mathrm{e}^{2i\pi\varphi 2^j}\,|1\rangle\,|u\rangle$$
$$|0\rangle\,|u\rangle \longmapsto |0\rangle\,|u\rangle$$

Be careful: $\mathbf{U}^{2^j} = \underbrace{\mathbf{U}\cdots\mathbf{U}}_{2^j \text{ iterates}}$, in particular $\mathbf{U}^{2^j}\,|u\rangle \neq (\mathbf{U}\,|u\rangle)^{2^j}$

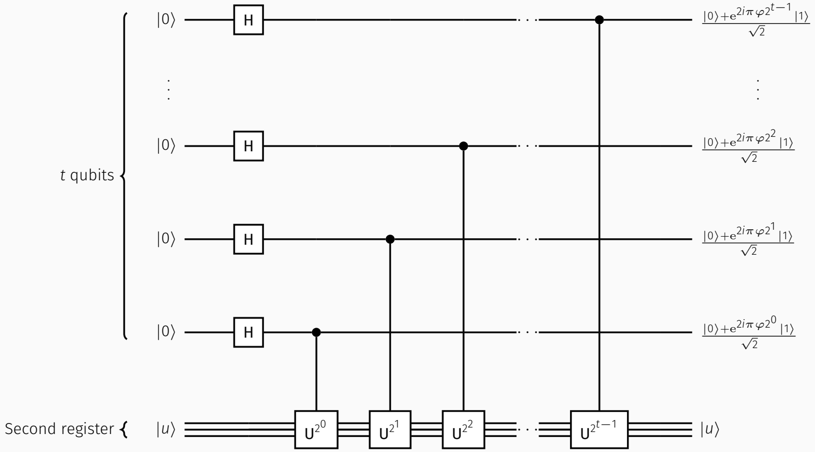
The algorithm:

1. Start with $\left|0^t\right\rangle|u\rangle$
2. Apply $\mathbf{H}^{\otimes t}\otimes\mathbf{I}$
3. For $i = 1$ to n:

   apply the controlled $\mathbf{U}^{2^j}$-gate to the i-th register.

Resulting quantum state:

$$\frac{1}{2^{t/2}}\left(|0\rangle + \mathrm{e}^{2i\pi 2^{t-1}\varphi}\,|1\rangle\right)\otimes\left(|0\rangle + \mathrm{e}^{2i\pi 2^{t-2}\varphi}\,|1\rangle\right)\otimes\cdots\otimes\left(|0\rangle + \mathrm{e}^{2i\pi 2^{0}\varphi}\,|1\rangle\right)\otimes|u\rangle$$



But what is the cost for computing $\mathbf{U}^{2^j}$? Is it 2^j?

Given an arbitrary U, computing the U^{2^j}-controlled costs $2^j \times \mathrm{Cost}(\mathsf{U})\ldots$

An example:

If $f : \{0,1\}^n \to \{0,1\}^n$ is a bijection efficiently computable, then the unitary

$$\mathsf{U} : |x\rangle \mapsto |f(x)\rangle$$

is efficiently computable. But, is

$$\mathsf{U}^{2^j} : |x\rangle \mapsto \left|f^{2^j}(x)\right\rangle \quad \left(f^{2^j} \text{ composition, not exponentiation}\right)$$

efficiently computable? It depends of the particular shape of $f\ldots$

$\longrightarrow$ Does it imply that phase estimation has an exponential cost?

Given an arbitrary U, computing the U^{2^j}-controlled costs $2^j \times \mathsf{Cost}(\mathsf{U})$ ...

An example:

If $f : \{0,1\}^n \to \{0,1\}^n$ is a bijection efficiently computable, then the unitary

$$\mathsf{U} : |\mathsf{x}\rangle \mapsto |f(\mathsf{x})\rangle$$

is efficiently computable. But, is

$$\mathsf{U}^{2^j} : |\mathsf{x}\rangle \mapsto \left| f^{2^j}(\mathsf{x}) \right\rangle \quad \left(f^{2^j} \text{ composition, not exponentiation} \right)$$

efficiently computable? It depends of the particular shape of f ...

$\longrightarrow$ Does it imply that phase estimation has an exponential cost?

No... or Yes... It depends!

As in the classical case: computing f^{2^j} is expensive $\left(2^j \times \mathsf{Cost}(f) \right)$ except for some functions...

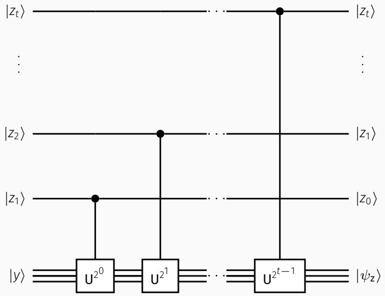
Phase estimation: be careful, in the general case

Computing U^{2^j} costs $2^j \times \mathsf{Cost}(\mathsf{U})$ unless one succeeds to use the particular shape of U ...

All the game in phase estimation lies in computing efficiently (designing an efficient circuit)

$$z_1, \ldots, z_t \in \{0,1\}^t, \quad \mathbf{V} : |z_1 \ldots z_t\rangle\, |u\rangle \mapsto |z_1 \ldots z_t\rangle\, |\psi_{\mathbf{z}}\rangle$$

where **V** is the following unitary



Phase estimation: be careful

Computing $\mathbf{U}^{2^j}$ costs $2^j \times \mathrm{Cost}(\mathbf{U})$ unless one succeeds to use the particular shape of $\mathbf{U} \ldots$

$\longrightarrow$ Let us take a look at the classical case!

What is the cost to compute x^{2^j}? Is it 2^j?

What is the cost to compute x^{2^j}? Is it 2^j?

Of course not... fast exponentiation

- Stupid algorithm: $y = 1$ and then 2^j times: $y \leftarrow yx$; output y
- Clever algorithm: if j even, $y \leftarrow 2^{2^{j/2}}$; outputs y^2; otherwise $y \leftarrow 2^{2^{(j-1)/2}}$ then outputs $2y^2$.

$\longrightarrow$ To compute $2^{2^{j/2}}$ or $2^{2^{(j-1)/2}}$: recursive call

Cost?

- Stupid algorithm: 2^j multiplications!
- Clever algorithm: $\log 2^j = j$ recursive calls and 1 or 2 multiplications for each call

$$\longrightarrow \text{It costs } j \times \underbrace{j^2}_{\text{cost of squaring}}$$

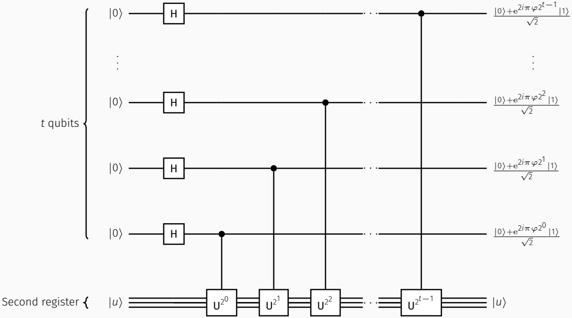
$\longrightarrow$ The "clever" algorithm is exponentially faster...

Be careful: we have used the particular shape of $x \mapsto x^{2^j}$

Usually $f^{2^j}(x) \neq f^{2^{j/2}}(x)^2$ but $f^{2^j}(x) = f^{2^{j/2}}\left(f^{2^{j/2}}(x)\right)$

$$\mathsf{U}\,|u\rangle = e^{2i\pi\varphi}\,|u\rangle \implies \mathsf{U}^{2^j}\,|u\rangle = e^{2i\pi 2^j\varphi}\,|u\rangle$$

$$\text{C-U}^{2^j}\,|0\rangle\,|u\rangle = |0\rangle\,|u\rangle \quad \text{and} \quad \text{C-U}^{2^j}\,|1\rangle\,|u\rangle = e^{2i\pi 2^j\varphi}\,|1\rangle\,|u\rangle$$



- First Step:

$$\frac{1}{\sqrt{2^t}}\left(|0\rangle + |1\rangle\right)^{\otimes t} \otimes |u\rangle$$

- Second Step:

$$\frac{1}{\sqrt{2^t}}\left(|0\rangle + e^{2i\pi 2^{t-1}\varphi}\,|1\rangle\right) \otimes \left(|0\rangle + e^{2i\pi 2^{t-2}\varphi}\,|1\rangle\right) \otimes \cdots \otimes \left(|0\rangle + e^{2i\pi 2^{0}\varphi}\,|1\rangle\right) \otimes |u\rangle$$

Suppose that

$$\varphi = 0.\varphi_1 \dots \varphi_t$$

See Lecture 5:

$$\frac{1}{2^{t/2}}\left(|0\rangle + e^{2i\pi 2^{t-1}\varphi}|1\rangle\right) \otimes \left(|0\rangle + e^{2i\pi 2^{t-2}\varphi}|1\rangle\right) \otimes \cdots \otimes \left(|0\rangle + e^{2i\pi 2^{0}\varphi}|1\rangle\right) \otimes |u\rangle$$

$$= \frac{1}{2^{t/2}}\left(|0\rangle + e^{2i\pi 0.\varphi_t}|1\rangle\right) \otimes \left(|0\rangle + e^{2i\pi 0.\varphi_{t-1}\varphi_t}|1\rangle\right) \otimes \cdots \otimes \left(|0\rangle + e^{2i\pi 0.\varphi_1\varphi_2\cdots\varphi_t}|1\rangle\right) \otimes |u\rangle$$

$$= \mathrm{QFT}_{\mathbb{Z}/2^t\mathbb{Z}}\,|\varphi_1 \dots \varphi_t\rangle$$

Applying $\mathrm{QFT}^{-1}_{\mathbb{Z}/2^t\mathbb{Z}}$ leads to:

$$|\varphi_1 \dots \varphi_t\rangle \longrightarrow \text{we have recovered } \varphi!$$

$\longrightarrow$ But what does happen if $\varphi = 0.\varphi_1 \dots \varphi_t\varphi_{t+1}\varphi_{t+2} \dots \varphi_\ell \dots$?

Important convention:

When working in $\mathbb{Z}/2^t\mathbb{Z}$ the considered Hilbert space is $\underbrace{\mathbb{C}^2 \otimes \cdots \otimes \mathbb{C}^2}_{t \text{ times}}$ and for all $x \in \mathbb{Z}/2^t\mathbb{Z}$,

$$|x\rangle \overset{\text{def}}{=} |x_1 \ldots x_t\rangle$$

where $x_1 \ldots x_t$ being the binary decomposition of x, *i.e.*, $x = \sum_{k=1}^{t} x_k 2^{t-k}$

$$\frac{1}{2^{t/2}} \left(|0\rangle + e^{2i\pi 2^{t-1}\varphi} |1\rangle \right) \otimes \left(|0\rangle + e^{2i\pi 2^{t-2}\varphi} |1\rangle \right) \otimes \cdots \otimes \left(|0\rangle + e^{2i\pi 2^{0}\varphi} |1\rangle \right) \otimes |u\rangle$$

$$= \frac{1}{2^{t/2}} \sum_{\ell=0}^{2^t-1} e^{2i\pi \ell \varphi} |\ell\rangle \otimes |u\rangle$$

Important convention:

When working in $\mathbb{Z}/2^t\mathbb{Z}$ the considered Hilbert space is $\underbrace{\mathbb{C}^2 \otimes \cdots \otimes \mathbb{C}^2}_{t \text{ times}}$ and for all $x \in \mathbb{Z}/2^t\mathbb{Z}$,

$$|x\rangle \stackrel{\text{def}}{=} |x_1 \ldots x_t\rangle$$

where $x_1 \ldots x_t$ being the binary decomposition of x, *i.e.*, $x = \sum_{k=1}^{t} x_k 2^{t-k}$

$$\frac{1}{2^{t/2}} \left(|0\rangle + \mathrm{e}^{2i\pi 2^{t-1}\varphi}\,|1\rangle\right) \otimes \left(|0\rangle + \mathrm{e}^{2i\pi 2^{t-2}\varphi}\,|1\rangle\right) \otimes \cdots \otimes \left(|0\rangle + \mathrm{e}^{2i\pi 2^{0}\varphi}\,|1\rangle\right) \otimes |u\rangle$$

$$= \frac{1}{2^{t/2}} \sum_{\ell=0}^{2^t-1} \mathrm{e}^{2i\pi\ell\varphi}\,|\ell\rangle \otimes |u\rangle$$

Applying $\mathrm{QFT}^{-1}_{\mathbb{Z}/2^t\mathbb{Z}}$ leads to:

$$\mathrm{QFT}^{-1}_{\mathbb{Z}/2^t\mathbb{Z}} \otimes \mathrm{I} \left(\frac{1}{2^{t/2}} \sum_{\ell=0}^{2^t-1} \mathrm{e}^{2i\pi\ell\varphi}\,|\ell\rangle \otimes |u\rangle\right) = \frac{1}{2^t} \sum_{k,\ell=0}^{2^t} \mathrm{e}^{2i\pi\ell(\varphi - \frac{k}{2^t})}\,|k\rangle \otimes |u\rangle$$

Best approximation of φ for the first t bits:

Le $b \in [\![0, 2^t - 1]\!]$ be such that $b/2^t = 0.b_1 \dots b_t$ and

$$0 \leq \varphi - \frac{b}{2^t} \leq 2^{-t} \quad : b/2^t \text{ best } t \text{ bits approximation of } \varphi$$

Up to now we have the following quantum state:

$$\frac{1}{2^t} \sum_{k,\ell=0}^{2^t-1} \mathrm{e}^{2i\pi\ell(\varphi - \frac{k}{2^t})} |k\rangle |u\rangle$$

Let α_j be the amplitude of $(b + j \bmod 2^t)$ in the first register:

$$|\alpha_j|^2 = \frac{1}{2^{2t}} \left| \sum_{\ell=0}^{2^t-1} \left(\mathrm{e}^{2i\pi\left(\varphi - \frac{b+j}{2^t}\right)} \right)^k \right|^2$$

Measure (see Exercise Session):

Let $m \in \{0,1\}^t$ be the outcome after measuring the first register in the computational basis (defining an integer in $[\![0, 2^t - 1]\!]$). We have

$$\mathbb{P}\left(|m - b| > \alpha\right) \leq \frac{1}{2(\alpha - 1)}$$

Best approximation of φ for the first t bits:

Le $b \in [\![0, 2^t - 1]\!]$ be such that $b/2^t = 0.b_1 \dots b_t$ and

$$0 \leq \varphi - \frac{b}{2^t} \leq 2^{-t} \quad : b/2^t \text{ best } t \text{ bits approximation of } \varphi$$

Measure:

Let m be the outcome after measuring the first register in the computational basis. We have

$$\mathbb{P}\left(|b - m| > \alpha\right) \leq \frac{1}{2(\alpha - 1)}$$

$\longrightarrow$ Determining φ with n bits of accuracy thanks to the output of the measure m ($t > n$):

$$\left|\frac{b}{2^t} - \frac{m}{2^t}\right| < 2^{-n}$$

Best approximation of φ for the first t bits:

Le $b \in [\![0, 2^t - 1]\!]$ be such that $b/2^t = 0.b_1 \dots b_t$ and

$$0 \leq \varphi - \frac{b}{2^t} \leq 2^{-t} \quad : b/2^t \text{ best } t \text{ bits approximation of } \varphi$$

Measure:

Let m be the outcome after measuring the first register in the computational basis. We have

$$\mathbb{P}\left(|b - m| > \alpha\right) \leq \frac{1}{2(\alpha - 1)}$$

$\longrightarrow$ Determining φ with n bits of accuracy thanks to the output of the measure m ($t > n$):

$$\left| \frac{b}{2^t} - \frac{m}{2^t} \right| < 2^{-n}$$

$\longrightarrow$ Therefore: choosing $\alpha = 2^{t-n} - 1$ in the above probability...

But to reach a probability of success $\geq 1 - \varepsilon$:

$$\frac{1}{2(\alpha - 1)} = \frac{1}{2(2^{t-n} - 2)} \leq \varepsilon \iff t = n + \left\lceil \log\left(2 + \frac{1}{2\varepsilon}\right) \right\rceil$$

Phase estimation:

- **Input:** a unitary **U** with eigenstate $|u\rangle$:
$$\mathbf{U}\,|u\rangle = \mathrm{e}^{2i\pi\varphi}\,|u\rangle$$
- **Output:** $\varphi \in [0,1)$, *i.e.*, the knowledge of the associate eigenvalue of $|u\rangle$

Proposition:

The phase estimation (before the last step measuring in the computational basis) computes,

$$\left|0^t\right\rangle |u\rangle \mapsto |\psi_u\rangle\,|u\rangle$$

such that $|\psi_u\rangle$ is an approximation of φ, *i.e.* when measuring the first register we obtain $\widetilde{\varphi} \in \{0,1\}^t$ admitting the same first n bits than φ with probability $\geq 1-\varepsilon$ if t is chosen as

$$t = n + \left\lceil \log\left(2 + \frac{1}{2\varepsilon}\right)\right\rceil$$

Furthermore, the algorithm uses $O(t^2)$ elementary gates and calls to controlled-$\mathbf{U}^{2^j}$ of $0 \leq j < t$

- Be careful: we need to compute $\left(\mathsf{U}^{2^j}\right)_{0\leq j\leq t}$ which has a cost $\geq 2^t$ unless one uses the particular shape of U. . .
- Accuracy with a probability exponentially close to 1 at the cost of a "constant" overhead: n bits of φ with probability $1 - \mathrm{e}^{-Cn}$ but with $t = O(n)$
- Be careful: to run phase estimation we also need to be able to compute the eigenvector $|u\rangle$ . . .

APPLICATION 1: QFT OVER $\mathbb{Z}/N\mathbb{Z}$

Recall that characters of

- $\mathbb{F}_2^n$: $\chi_{\mathsf{x}}(\mathsf{y}) = (-1)^{\mathsf{x}\cdot\mathsf{y}}$
- $\mathbb{Z}/2^n\mathbb{Z}$: $\chi_x(y) = \mathrm{e}^{\frac{2i\pi xy}{2^n}}$
- $\mathbb{Z}/N\mathbb{Z}$: $\chi_x(y) = \mathrm{e}^{\frac{2i\pi xy}{N}}$

Lecture 5: computing **efficiently** $\left(O(n) \text{ and } O(n^2)\right)$

$$\mathsf{QFT}_{\mathbb{F}_2^n} = \mathsf{H}^{\otimes n} : |\mathsf{x}\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{\mathsf{y}\in\{0,1\}^n} (-1)^{\mathsf{x}\cdot\mathsf{y}} |\mathsf{y}\rangle \quad \text{and} \quad \mathsf{QFT}_{\mathbb{Z}/2^n\mathbb{Z}} : |x\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{y\in\mathbb{Z}/2^n\mathbb{Z}} \mathrm{e}^{\frac{2i\pi xy}{2^n}} |y\rangle$$

Aim: computing efficiently $\mathsf{QFT}_{\mathbb{Z}/N\mathbb{Z}}$ (when N not a power of 2)

$$\mathsf{QFT}_{\mathbb{Z}/N\mathbb{Z}} : |x\rangle \longmapsto \frac{1}{\sqrt{N}} \sum_{y\in\mathbb{Z}/N\mathbb{Z}} \mathrm{e}^{\frac{2i\pi xy}{N}} |y\rangle$$

Computing $\mathbf{QFT}_{\mathbb{Z}/N\mathbb{Z}}$: use phase estimation!

$$\mathrm{U}_1\left(|k\rangle\,|0\rangle\right) \mapsto |k\rangle\,\mathrm{QFT}_{\mathbb{Z}/N\mathbb{Z}}\,|k\rangle \quad \text{and} \quad \mathrm{U}_2\left(\mathrm{QFT}_{\mathbb{Z}/N\mathbb{Z}}\,|k\rangle\,|0\rangle\right) \mapsto \mathrm{QFT}_{\mathbb{Z}/N\mathbb{Z}}\,|k\rangle\,|k\rangle$$

$\longrightarrow$ These two unitaries are enough to compute $\mathrm{QFT}_{\mathbb{Z}/N\mathbb{Z}}\,|k\rangle$!

We can perform $\mathrm{QFT}_{\mathbb{Z}/N\mathbb{Z}}$ as:

$$|k\rangle\,|0\rangle \xrightarrow{\mathrm{U}_1} |k\rangle\,\mathrm{QFT}_{\mathbb{Z}/N\mathbb{Z}}\,|k\rangle \xrightarrow{\mathrm{SWAP}} \mathrm{QFT}_{\mathbb{Z}/N\mathbb{Z}}\,|k\rangle\,|k\rangle \xrightarrow{\mathrm{U}_2^{-1}} \mathrm{QFT}_{\mathbb{Z}/N\mathbb{Z}}\,|k\rangle\,|0\rangle$$

$$\mathsf{U}_1\left(|k\rangle\,|0\rangle\right) \mapsto |k\rangle\,\mathsf{QFT}_{\mathbb{Z}/N\mathbb{Z}}\,|k\rangle \quad \text{and} \quad \mathsf{U}_2\left(\mathsf{QFT}_{\mathbb{Z}/N\mathbb{Z}}\,|k\rangle\,|0\rangle\right) \mapsto \mathsf{QFT}_{\mathbb{Z}/N\mathbb{Z}}\,|k\rangle\,|k\rangle$$

Be careful: $|k\rangle$ here is s.t $k \in \mathbb{Z}/N\mathbb{Z}$ and N may not be a power of two... In particular $|k\rangle$ cannot be written as $|0010\ldots1\rangle$

$(|k\rangle)_{k\in\mathbb{Z}/N\mathbb{Z}}$ is an orthonormal basis of an Hilbert space of dimension N

$\longrightarrow$ This quantum space is called the space of qudits!

Two possibilities to perform computation with qudits: (*i*) encode qudits in qubits or (*ii*) implement your quantum device directly with Hilbert spaces of dimension > 2

It is the same issue with classical computer! How to implement trits, namely $\mathbb{Z}/3\mathbb{Z}$?

To build the unitary $|k\rangle\,|0\rangle \mapsto |k\rangle\,\mathbf{QFT}_{\mathbb{Z}/N\mathbb{Z}}\,|k\rangle$ (admitting we can perform efficiently the different unitaries over qudits)

1. Start from $|k\rangle\,|0\rangle\,|0\rangle$
2. Apply the "uniform superposition" over the second register
$$|k\rangle \frac{1}{\sqrt{N}} \sum_{j\in\mathbb{Z}/N\mathbb{Z}} |j\rangle\,|0\rangle$$
3. Apply the multiplication operator $\Big(|x\rangle\,|y\rangle\,|0\rangle \mapsto |x\rangle\,|y\rangle\,|xy \bmod N\rangle\Big)$
$$|k\rangle \frac{1}{\sqrt{N}} \sum_{j\in\mathbb{Z}/N\mathbb{Z}} |j\rangle\,|kj \bmod N\rangle$$
4. Apply the "phase flip in $\mathbb{Z}/N\mathbb{Z}$" $\Big(|x\rangle \mapsto \mathrm{e}^{2i\pi\frac{x}{N}}\,|x\rangle\Big)$ on the third register
$$|k\rangle \frac{1}{\sqrt{N}} \sum_{j\in\mathbb{Z}/N\mathbb{Z}} \mathrm{e}^{2i\pi\frac{kj}{N}}\,|j\rangle\,|kj \bmod N\rangle$$
5. Apply the inverse of the multiplication operation:
$$|k\rangle \frac{1}{\sqrt{N}} \sum_{j\in\mathbb{Z}/N\mathbb{Z}} \mathrm{e}^{2i\pi\frac{kj}{N}}\,|j\rangle\,|0\rangle = |k\rangle\,\mathbf{QFT}_{\mathbb{Z}/N\mathbb{Z}}\,|k\rangle\,|0\rangle$$

COMPUTING THE SECOND UNITARY U_2: USE PHASE AMPLIFICATION

$$\mathsf{U} : |k\rangle \mapsto |k + 1 \bmod N\rangle$$

$\longrightarrow \mathsf{U}^{2^j} : |k\rangle \mapsto \left|k + 2^j \bmod N\right\rangle$ can be built in time $O(\log N)$

$\left(x \mapsto x + 2^j \bmod N \text{ can be classically computed in time } O(\log N) \right)$

We have the following computation:

$$\begin{aligned} \mathsf{U}\left(\mathsf{QFT}_{\mathbb{Z}/N\mathbb{Z}}\,|k\rangle\right) &= \frac{1}{\sqrt{N}} \sum_{y \in \mathbb{Z}/N\mathbb{Z}} \mathrm{e}^{\frac{2i\pi ky}{N}} \mathsf{U}|y\rangle \\ &= \frac{1}{\sqrt{N}} \sum_{y \in \mathbb{Z}/N\mathbb{Z}} \mathrm{e}^{\frac{2i\pi ky}{N}} \,|y+1\rangle \\ &= \mathrm{e}^{\frac{-2i\pi k}{N}} \frac{1}{\sqrt{N}} \sum_{y \in \mathbb{Z}/N\mathbb{Z}} \mathrm{e}^{\frac{2i\pi ky}{N}} \,|y\rangle \\ &= \mathrm{e}^{2i\pi\,\frac{N-k}{N}} \;\mathsf{QFT}_{\mathbb{Z}/N\mathbb{Z}}\,|k\rangle \end{aligned}$$

$\longrightarrow \mathsf{QFT}_{\mathbb{Z}/N\mathbb{Z}}\,|k\rangle$ is an eigenvector of U with eigenvalue $\mathrm{e}^{2i\pi\varphi}$ where

$$\varphi \stackrel{\text{def}}{=} \frac{N-k}{N}$$

The translation operator **U** *is diagonal in the Fourier basis*

$\mathbf{QFT}_{\mathbb{Z}/N\mathbb{Z}} |k\rangle$ is an eigenvector of **U** with eigenvalue $\mathrm{e}^{2i\pi\varphi}$ where

$$\varphi \stackrel{\text{def}}{=} \frac{N-k}{N}$$

Applying phase estimation with $n = \lceil \log N \rceil$ (bits of precision) enables to compute:

$$\mathbf{QFT}_{\mathbb{Z}/N\mathbb{Z}} |k\rangle |0\rangle \longmapsto \mathbf{QFT}_{\mathbb{Z}/N\mathbb{Z}} |k\rangle |N-k\rangle$$

$\longrightarrow$ Be careful: phase estimation gives only an approximation of the transform!

Therefore: after applying the unitary $|x\rangle \mapsto |N-x\rangle$ we obtain an approximation of

$$\mathbf{QFT}_{\mathbb{Z}/N\mathbb{Z}} |k\rangle |0\rangle \longmapsto \mathbf{QFT}_{\mathbb{Z}/N\mathbb{Z}} |k\rangle |k\rangle$$

Cost:

Given $t = O(\log N)$, we have a cost of $O(t^2)$ plus the cost to run $\mathbf{U}^{2^j} : |k\rangle \mapsto \left| k + 2^j \bmod N \right\rangle$ for $0 \leq j < t$ which can be done in time $O(t^3)$ (clever combination of the $\mathbf{U}^{2^j}$-controlled)

$\longrightarrow$ Final cost to compute $\mathbf{QFT}_{\mathbb{Z}/N\mathbb{Z}}$: $O\left(\log^3 N\right)$

WHAT ABOUT THE GENERAL CASE?

Is it possible to efficiently build $\mathbf{QFT}_G$ where G is any arbitrary finite abelian group?

WHAT ABOUT THE GENERAL CASE?

Is it possible to efficiently build **QFT**$_G$ where G is any arbitrary finite abelian group?

$\longrightarrow$ Yes!

How to proceed (rough explanation)?

Any finite abelian group G of size N is isomorphic to the product of cyclic groups:

$$\mathbb{Z}/n_1\mathbb{Z} \times \cdots \times \mathbb{Z}/n_k\mathbb{Z}$$

Then (admitted),

$$\mathbf{QFT}_G \text{ can be written as } \mathbf{QFT}_{\mathbb{Z}/n_1\mathbb{Z}} \otimes \cdots \otimes \mathbf{QFT}_{\mathbb{Z}/n_k\mathbb{Z}}$$

$\longrightarrow$ We deduce that **QFT**$_G$ can be computed in time $O\left(\log^3 \sharp G\right)$

Be careful, given a finite Abelian group it is classically hard to compute its decomposition as cyclic groups... Quantum case: end of the lecture

APPLICATION 2: ORDER FINDING

Order finding problem:

- **Input:** integers x, N where $\gcd(x, N) = 1$
- **Output:** least positive integer r such that $x^r = 1 \bmod N$

Solving the factorization reduces to this problem

$$\left(\text{solving order finding} \implies \text{solving factorization}\right)$$

Proposition:

We can quantumly determine the order r (with high probability) in time

$$O\left(\log^3 N\right)$$

$\longrightarrow$ Best classical algorithms are sub-exponential in N:

$$\exp\left((c + o(1)) \log^{\alpha}(N) \log^{1-\alpha}(\log N)\right)$$

where c, α are constants

Suppose that we work in the space of L qubits

Given $y \in [\![0, 2^L - 1]\!]$, we will naturally identify $|y\rangle$ to $|y_1 \dots y_L\rangle$

where $y_1 \dots y_L$ **binary decomposition of y**

For instance:

Given $3, 5 \in [\![0, 2^3 - 1]\!]$,

$$|3\rangle = |011\rangle \quad \text{and } |5\rangle = |101\rangle$$

x integer : $\mathbf{gcd}(x, N) = 1$ and r its order, smallest positive integer such that $x^r = 1 \textbf{ mod } N$

Phase estimation applied to the following unitary and eigenvector:

Let,

$$L \stackrel{\text{def}}{=} \lceil \log N \rceil \quad \text{(work in the space of } L\text{-qubits)}$$

$$\forall y \in \{0,1\}^L = [\![0, 2^L - 1]\!], \quad \mathbf{U}\,|y\rangle \stackrel{\text{def}}{=} \begin{cases} |xy \textbf{ mod } N\rangle & \text{if } 0 \leq y \leq N-1 \\ |y\rangle & \text{otherwise } \left(N - 1 < y < 2^{\lceil \log N \rceil}\right). \end{cases}$$

$$\forall s \in [\![0, r]\!], \quad |u_s\rangle \stackrel{\text{def}}{=} \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \mathrm{e}^{-\frac{2i\pi sk}{r}} \left|x^k \textbf{ mod } N\right\rangle \text{ eigenvector of } \mathbf{U} \text{ with eigenvalue } \mathrm{e}^{2i\pi\frac{s}{r}}$$

$\longrightarrow$ We work here in the space of qubits (natural trick, identity if integers $\geq N-1$)

$$\begin{aligned} \mathbf{U}\,|u_s\rangle &= \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \mathrm{e}^{-\frac{2i\pi sk}{r}}\, \mathbf{U} \left|x^k \textbf{ mod } N\right\rangle \\ &= \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \mathrm{e}^{-\frac{2i\pi sk}{r}} \left|x^{k+1} \textbf{ mod } N\right\rangle \\ &= \mathrm{e}^{2i\pi\frac{s}{r}}\, |u_s\rangle \end{aligned}$$

$\longrightarrow$ Be careful: in the last equality we used: x has order r modulo N, thus $x^r = 1 \textbf{ mod } N$

For the eigenvalue $\frac{s}{r}$: we work in $\mathbb{Z}/N\mathbb{Z}$ and $L = \lceil \log N \rceil$.

To perform efficiently phase estimation, two issues:

- How to compute efficiently the $\mathbf{U}^{2^j}$'s?
- How to compute the eigenvector $|u_s\rangle$?

$\longrightarrow$ We will be able to recover approximations of $\frac{s}{r}$, **not** r...

For the eigenvalue $\frac{s}{r}$: we work in $\mathbb{Z}/N\mathbb{Z}$ and $L = \lceil \log N \rceil$.

To perform efficiently phase estimation, two issues:

- How to compute efficiently the $\mathbf{U}^{2^j}$'s?
- How to compute the eigenvector $|u_s\rangle$?

$\longrightarrow$ We will be able to recover approximations of $\frac{s}{r}$, **not** r...

Be patient!

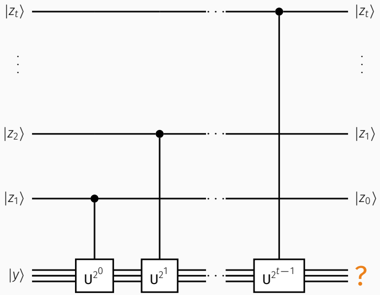
Parameter of phase estimation:

We will determine the first $2L+1$ bits of $\frac{s}{r}$ with probability $1-\varepsilon$

$\longrightarrow$ Choose in phase estimation $t = 2L + 1 + \left\lceil \log\left(2 + \frac{1}{2\varepsilon}\right)\right\rceil$

In particular: $t = O(L)$ even if $\varepsilon = \mathrm{e}^{-CL}$ with $C > 0$ (constant)

$$\mathsf{U}\,|y\rangle = |xy \bmod N\rangle \quad (0 \leq y \leq N-1)$$



The above circuit (used in the phase estimate) performs the following computation:

$$\begin{aligned} |z_t \ldots z_1\rangle\,|y\rangle &\longrightarrow |z\rangle\,\mathsf{U}^{z_t 2^{t-1}} \cdots \mathsf{U}^{z_1 2^0}\,|y\rangle \\ &= |z\rangle \left| x^{z_t 2^{t-1}} \times \cdots \times x^{z_1 2^0} y \bmod N \right\rangle \\ &= |z\rangle\,|yx^z \bmod N\rangle \end{aligned}$$

$\longrightarrow$ To perform efficiently phase estimation: compute $|z\rangle\,|y\rangle \mapsto |z\rangle\,|yx^z \bmod N\rangle$ efficiently

(modular exponentiation)

Aim: computing efficiently

$$|z\rangle\,|y\rangle \mapsto |z\rangle\,\left|yx^z \bmod N\right\rangle$$

1. Let $\mathrm{U_{EM}} : |z\rangle\,|y\rangle \mapsto |z\rangle\,|y \oplus (x^z \bmod N)\rangle$ (be careful $z \mapsto x^z \bmod N$ not bijective)

$$|z\rangle\,|y\rangle\,|0\rangle \xrightarrow{\mathrm{U_{EM}}} |z\rangle\,|y\rangle\,\left|x^z \bmod N\right\rangle \xrightarrow{\text{mult}} |z\rangle\,\left|yx^z \bmod N\right\rangle\,\left|x^z \bmod N\right\rangle \xrightarrow{\mathrm{U_{EM}^{-1}}} |z\rangle\,\left|yx^z \bmod N\right\rangle\,|0\rangle$$

Aim: computing efficiently

$$|z\rangle\,|y\rangle \mapsto |z\rangle\,\left|yx^z \bmod N\right\rangle$$

1. Let $U_{EM} : |z\rangle\,|y\rangle \mapsto |z\rangle\,|y \oplus (x^z \bmod N)\rangle$ (be careful $z \mapsto x^z \bmod N$ not bijective)

$$|z\rangle\,|y\rangle\,|0\rangle \xrightarrow{U_{EM}} |z\rangle\,|y\rangle\,\left|x^z \bmod N\right\rangle \xrightarrow{\text{mult}} |z\rangle\,\left|yx^z \bmod N\right\rangle\left|x^z \bmod N\right\rangle \xrightarrow{U_{EM}^{-1}} |z\rangle\,\left|yx^z \bmod N\right\rangle|0\rangle$$

2. Computing efficiently U_{EM}: classically

$$x \mapsto x^z \bmod N$$

can by computed in $O(\log z) = O(\log t) = O(\log N)$ squaring, therefore $O\left(\log^3 N\right)$ operations

Aim: computing efficiently

$$|z\rangle\,|y\rangle \mapsto |z\rangle\,\left|yx^z \bmod N\right\rangle$$

1. Let $U_{EM} : |z\rangle\,|y\rangle \mapsto |z\rangle\,|y \oplus (x^z \bmod N)\rangle$ (be careful $z \mapsto x^z \bmod N$ not bijective)

$$|z\rangle\,|y\rangle\,|0\rangle \xrightarrow{U_{EM}} |z\rangle\,|y\rangle\,\left|x^z \bmod N\right\rangle \xrightarrow{\text{mult}} |z\rangle\,\left|yx^z \bmod N\right\rangle\left|x^z \bmod N\right\rangle \xrightarrow{U_{EM}^{-1}} |z\rangle\,\left|yx^z \bmod N\right\rangle|0\rangle$$

2. Computing efficiently U_{EM}: classically

$$x \mapsto x^z \bmod N$$

can by computed in $O\left(\log z\right) = O(\log t) = O\left(\log N\right)$ squaring, therefore $O\left(\log^3 N\right)$ operations

Conclusion: using phase estimation

We determine the first $2L+1$ bits of $\frac{s}{r}$ with probability $1 - e^{-CL}$ in time $O(L^3)$ where $L = \lceil \log N \rceil$

Aim: computing

$$|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \mathrm{e}^{-\frac{2i\pi sk}{r}} \left| x^k \bmod N \right\rangle$$

But we do not know $r \dots$ Our aim is to find it!

The trick:

$$\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |u_s\rangle = |1\rangle$$

$\longrightarrow$ Plugging $|1\rangle$ in the phase estimation algorithm will give the first $2L+1$ bits of $\frac{s}{r}$ for some (uniform and unknown) $s \in [\![0, r-1]\!]$ with probability $1-\varepsilon$

Exercise session:

Proof of this statement

Up to now we have recovered (with high probability) in quantum time $O(L^3)$ the first $2L+1$ bits of $\frac{s}{r}$ where $0 \leq s < r \in [\![1, N-1]\!]$

$\longrightarrow$ It does not give r, even $\frac{s}{r}$ . . .

Theorem (admitted) about continued fractions:

Let $\widetilde{\varphi}$ be a rational given as input, let s and r be L bits integers such that

$$\left|\frac{s}{r} - \widetilde{\varphi}\right| < \frac{1}{2r^2}$$

Then, there exists an algorithm (using "continued fractions") that outputs (s', r') which verifies

$$\mathbf{gcd}(s', r') = 1 \quad \text{and} \quad \tfrac{s'}{r'} = \tfrac{s}{r}$$

using $O(L^3)$ classical operations

In our case:

With probability $1 - \varepsilon$: phase estimation outputs $\widetilde{\varphi}$ an approximation of $\frac{s}{r}$ accurate to $2L + 1$ bits, therefore:

$$\left|\frac{s}{r} - \widetilde{\varphi}\right| \leq \frac{1}{2^{2L+1}} \leq \frac{1}{2r^2} \quad (\text{as } r \leq N - 1)$$

$\longrightarrow$ In time $O(L^3)$ we compute s', r' co-prime such that $\frac{s'}{r'} = \frac{s}{r}$

s', r' are co-prime such that $\frac{s'}{r'} = \frac{s}{r}$

$\longrightarrow$ If $\mathbf{gcd}(s, r) > 1$ then $r' \neq r$, only $r' \mid r \dots$

A solution (but inefficient. . .):

The number of prime numbers $< r$ is $\approx r/\log(r)$

$$\longrightarrow \mathbb{P}(\mathbf{gcd}(s, r) = 1) \approx \log(r)/r \text{ as } s \text{ is uniformly picked in } [\![0, r-1]\!]$$

Therefore we need to repeat $\approx r = O(L)$ number of times the algorithm before reaching $\mathbf{gcd}(s, r) = 1$. It will increase the cost from $O(L^3)$ to $O(L^4)$.

Fundamental remark:

$$\frac{s_1'}{r_1'} = \frac{s_1}{r} \implies r\, s_1' = s_1\, r_1' \quad \text{and} \quad \frac{s_2'}{r_2'} = \frac{s_2}{r} \implies r\, s_2' = s_2\, r_2'$$

We have $\mathbf{gcd}(s_1', r_1') = \mathbf{gcd}(s_2', r_2') = 1$, supposing that $\mathbf{gcd}(s_1', s_2') = 1$ implies that $r = \mathbf{lcm}(r_1', r_2')$

$\longrightarrow$ Therefore: obtaining two estimations (s_1', r_1') and (s_2', r_2') and supposing that $\mathbf{gcd}(s_1', s_2') = 1$ we can recover $r = \mathbf{lcm}(r_1', r_2')$.

What is the probability that $\mathbf{gcd}(s_1', s_2') = 1$ given that s_1' and that s_2' are uniformly distributed in $[\![0, r-1]\!]$?

Fundamental remark:

$$\frac{s_1'}{r_1'} = \frac{s_1}{r} \Longrightarrow r\, s_1' = s_1\, r_1' \quad \text{and} \quad \frac{s_2'}{r_2'} = \frac{s_2}{r} \Longrightarrow r\, s_2' = s_2\, r_2'$$

We have $\mathbf{gcd}(s_1', r_1') = \mathbf{gcd}(s_2', r_2') = 1$, supposing that $\mathbf{gcd}(s_1', s_2') = 1$ implies that $r = \mathbf{lcm}(r_1', r_2')$

$\longrightarrow$ Therefore: obtaining two estimations (s_1', r_1') and (s_2', r_2') and supposing that $\mathbf{gcd}(s_1', s_2') = 1$ we can recover $r = \mathbf{lcm}(r_1', r_2')$.

What is the probability that $\mathbf{gcd}(s_1', s_2') = 1$ given that s_1' and that s_2' are uniformly distributed in $[\![0, r-1]\!]$?

$\longrightarrow$ It is $\geq \frac{1}{4}$ (see exercise session)

In conclusion:

Repeating the algorithm a constant number of times enables to recover r with probability exponentially close to one (times $(1-\varepsilon)$)!

To compute the order r of $x \bmod N$ (where $\gcd(x, N) = 1$) we first run a constant number of times the phase estimation with $t = 2\lceil \log N \rceil + 1 + \log\left(2 + \frac{1}{2\varepsilon}\right)$. It has been necessary to compute:

- $\mathrm{QFT}_{\mathbb{Z}/2^t\mathbb{Z}}$: done in time $O(t^2) = O(\log^2 N)$
- modular exponentiation $|z\rangle |y\rangle \mapsto |z\rangle |yx^z \bmod N\rangle$: done in time $O(\log^3 N)$

It outputs a $2\lceil \log N \rceil + 1$ approximation of some $\frac{s}{r}$ where $s \in [\![0, r-1]\!]$ is uniform and unknown

To compute the order r of $x \bmod N$ (where $\mathbf{gcd}(x, N) = 1$) we first run a constant number of times the phase estimation with $t = 2\lceil \log N \rceil + 1 + \log\left(2 + \frac{1}{2\varepsilon}\right)$. It has been necessary to compute:

- $\mathrm{QFT}_{\mathbb{Z}/2^t\mathbb{Z}}$: done in time $O(t^2) = O(\log^2 N)$
- modular exponentiation $|z\rangle |y\rangle \mapsto |z\rangle |yx^z \bmod N\rangle$: done in time $O(\log^3 N)$

It outputs a $2\lceil \log N \rceil + 1$ approximation of some $\frac{s}{r}$ where $s \in [\![0, r-1]\!]$ is uniform and unknown

Then after collecting some approximations of $\frac{s_i}{r}$, apply continued fraction algorithm to obtain (s_i', r_i') in time $O\left(\log^3 N\right)$ with $\frac{s_i'}{r_i'} = \frac{s_i}{r}$. It enables to get r by computing some $\mathbf{lcm}(r_i', r_j')$.

To compute the order r of $x \textbf{ mod } N$ (where $\textbf{gcd}(x, N) = 1$) we first run a constant number of times the phase estimation with $t = 2\lceil \textbf{log}\, N \rceil + 1 + \textbf{log}\left(2 + \frac{1}{2\varepsilon}\right)$. It has been necessary to compute:

- $\textbf{QFT}_{\mathbb{Z}/2^t\mathbb{Z}}$: done in time $O(t^2) = O(\textbf{log}^2 N)$
- modular exponentiation $|z\rangle\,|y\rangle \mapsto |z\rangle\,|yx^z \textbf{ mod } N\rangle$: done in time $O(\textbf{log}^3 N)$

It outputs a $2\lceil \textbf{log}\, N \rceil + 1$ approximation of some $\frac{s}{r}$ where $s \in [\![0, r-1]\!]$ is uniform and unknown

Then after collecting some approximations of $\frac{s_i}{r}$, apply continued fraction algorithm to obtain (s_i', r_i') in time $O\left(\textbf{log}^3 N\right)$ with $\frac{s_i'}{r_i'} = \frac{s_i}{r}$. It enables to get r by computing some $\textbf{lcm}(r_i', r_j')$.

$\longrightarrow$ This procedure works with probability $(1 - \mathrm{e}^{-C \log N})(1 - \varepsilon)$ for constant $C > 0$ depending on the number of repetitions

Final cost:

$$O\left(\textbf{log}^3 N\right)$$

$\longrightarrow$ This could be done in time $O(\textbf{log}^2(N)\textbf{poly}(\textbf{loglog}N))$

Order finding algorithm is efficient because we know quantumly how to perform classical computations **and the quantum Fourier transform over $\mathbb{Z}/2^t\mathbb{Z}$**

SHOR'S ALGORITHM

Factoring problem:

- **Input:** an integer N
- **Output:** a non-trivial factor of N

$\longrightarrow$ Security of public-key encryption scheme RSA relies on the hardness of this problem...

Classically best algorithms have a complexity:

$$\exp\left((c+o(1))\log^{\alpha}(N)\log^{1-\alpha}(\log N)\right)$$

Shor's algorithm is basically applying order finding for some random $x \in [\![0, N-1]\!] \ldots$

But why?

Theorem 1:

Suppose N is a L bits not prime integer and $1 \leq y \leq$ be a non-trivial integer such that

$$y^2 = 1 \textbf{ mod } N$$

Then, at least $\textbf{gcd}(y-1, N)$ or $\textbf{gcd}(y+1, N)$ is a non-trivial factor of N that can be computed in time $O(L^3)$

Theorem 2:

Suppose that $N = p_1^{\alpha_1} \cdots p_m^{\alpha_m}$ where the p_i's are different primes. Let x be an integer chosen uniformly at random, subject to the requirements that $1 \leq x \leq N-1$ and $\textbf{gcd}(x, N) = 1$. Let r be the order of x. Then,

$$\mathbb{P}\left(r \text{ is even} \quad \text{and} \quad x^{r/2} \neq -1 \textbf{ mod } N\right) \geq 1 - \frac{1}{2^m}$$

$\longrightarrow$ Let x be picked according to Theorem 2, then with (at least) a constant probability

$x^{r/2}$ is a solution $\neq \pm 1$ of $\left(X^2 = 1 \textbf{ mod } N\right)$

According to Theorem 1: $\textbf{gcd}(x^{r/2} - 1, N)$ or $\textbf{gcd}(x^{r/2} + 1, N)$ is a $\neq \pm 1$ factor of N

Theorem 1:

Suppose N is a L bits not prime integer and $1 \leq y \leq$ be a non-trivial integer such that

$$y^2 = 1 \ \mathbf{mod} \ N$$

Then, at least $\mathbf{gcd}(y-1, N)$ or $\mathbf{gcd}(y+1, N)$ is a non-trivial factor of N that can be computed in time $O(L^3)$

Theorem 2:

Suppose that $N = p_1^{\alpha_1} \cdots p_m^{\alpha_m}$ where the p_i's are different primes. Let x be an integer chosen uniformly at random, subject to the requirements that $1 \leq x \leq N-1$ and $\mathbf{gcd}(x, N) = 1$. Let r be the order of x. Then,

$$\mathbb{P}\left(r \text{ is even} \quad \text{and} \quad x^{r/2} \neq -1 \ \mathbf{mod} \ N\right) \geq 1 - \frac{1}{2^m}$$

$\longrightarrow$ Let x be picked according to Theorem 2, then with (at least) a constant probability

$$x^{r/2} \text{ is a solution} \neq \pm 1 \text{ of } \left(X^2 = 1 \ \mathbf{mod} \ N\right)$$

According to Theorem 1: $\mathbf{gcd}(x^{r/2} - 1, N)$ or $\mathbf{gcd}(x^{r/2} + 1, N)$ is a $\neq \pm 1$ factor of N

Given x, we just need to compute its order to find a non-trivial factor!

1. Pick x uniformly at random in $[\![1, N]\!]$
2. Compute $d = \textbf{gcd}(x, N)$. If $d > 1$, output d
3. Use the quantum order-finding subroutine to find the order r of x **mod** N
4. If r is even and $x^{r/2} \neq -1 \textbf{ mod } N$ then compute $\textbf{gcd}(x^{r/2} - 1, N)$ or $\textbf{gcd}(x^{r/2} + 1, N)$ and test if one of these is a non-trivial factor of N. Otherwise go back to Step 1.

By using the law of total probability:

$$\begin{aligned}\mathbb{P}\,(\text{success}) &\geq \mathbb{P}\,(\text{success} \mid \text{ Step 3 succeeds})\,\mathbb{P}\,(\text{ Step 3 succeeds})\\ &= \mathbb{P}\left(r \text{ is even and } x^{r/2} \neq -1 \textbf{ mod } N\right)\mathbb{P}\,(\text{ order finding succeeds})\\ &\geq (1 - \mathrm{e}^{-C\log N})(1-\varepsilon)\left(1 - \frac{1}{2^m}\right) \quad (m \text{ number of prime factors of } N)\end{aligned}$$

$\longrightarrow$ Repeating the algorithm a constant amount of times gives a non-trivial factor

Final cost:

$$O\left(\log^3 N\right) \text{ cost of phase estimation} + \text{Step 4}$$

HIDDEN SUBGROUP PROBLEM

Shor's algorithm relies on the order-finding which itself crucially used $\mathsf{QFT}_{\mathbb{Z}/2^t\mathbb{Z}}$

$\Big($in the phase estimation$\Big)$

$\longrightarrow$ It turns out that what we did is extremely "general"

Techniques we have presented enable to compute the "period" of a wide class of functions. . .

- What do we mean by "general"?
- Computing the "period" of which class of functions and does it imply some interesting statements?

$\longrightarrow$ Hidden Subgroup Problem!

Hidden Subgroup Problem (HSP):

- **Input:** a function $f : G \to S$ where **G is a known group**[a] and S is a finite set
- **Promise:** f satisfies

$$f(x) = f(y) \text{ if and only if } y \in xH \qquad \text{i.e., } y = xh \text{ for some } h \in H \tag{1}$$

  for an **unknown subgroup $H \subseteq G$**
- **Output:** H

[a] see later for a precise definition

$\longrightarrow$ We say that f hides the subgroup H

Cosets:

The set:

$$xH \stackrel{\text{def}}{=} \{xh : h \in H\}$$

is called a **left-coset** of H.

$\longrightarrow$ A function f that hides H is constant on each left-coset of H and distinct on different left cosets

HSP may be seen as a purely abstract problem. . . But no!

Here are particular instantiations of HSP

- Simon's problem:

$$G = \mathbb{F}_2^n, \quad H = \{0, \mathbf{s}\} \quad \text{and} \quad f \quad \text{being the input in Simon's problem}$$

- Order finding:

$$G = \mathbb{Z}/\Phi(N)\mathbb{Z} \text{ (}\Phi \text{ be the Euler function)}, \quad H = \{rx : x \in \mathbb{Z}/\Phi(N)\mathbb{Z}\} \quad \text{and } f(a) = x^a \bmod N$$

- Discrete logarithm problem: see exercise session!
- etc. . .

Be careful:

In Shor's algorithm, when using a solver to order finding we don't know $\Phi(N)$ and therefore we don't know G . . .

HOW TO SOLVE IT IN THE ABELIAN CASE: QFT! (KITAEV, INSPIRED BY SHOR)

We suppose that G is Abelian (we note G in additive notation)

$f : G \longrightarrow S$ that hides some subgroup H

1. Start with $|0\rangle\,|0\rangle$, where the two registers have dimensions $\sharp G$ and $\sharp S$, respectively
2. Create a uniform superposition over G in the first register: $\frac{1}{\sqrt{\sharp G}} \sum_{g \in G} |g\rangle\,|0\rangle$
3. Compute f in superposition: $\frac{1}{\sqrt{\sharp G}} \sum_{g \in G} |g\rangle\,|f(g)\rangle$
4. Measure the second register. This yields some value $s \in G$. The second register collapses to (using the promise over f)
$$\frac{1}{\sqrt{\sharp H}} \sum_{h \in H} |s + h\rangle$$
5. Apply QFT_G giving: $\frac{1}{\sqrt{\sharp H}} \sum_{h \in H} |\chi_{s+h}\rangle$ for some quantum state $|\chi_{s+h}\rangle$
6. Measure and output the resulting $g \in G$

WHY DOES IT WORK?

G is Abelian, be $(\chi_g)_{g \in G}$ be its characters

$$
\begin{aligned}
|\chi_{s+h}\rangle &= \mathrm{QFT}_G \sum_{h \in H} |s+h\rangle \\
&= \frac{1}{\sqrt{G}} \sum_{h \in H} \mathrm{QFT}_G |s+h\rangle \\
&= \frac{1}{\sqrt{G}} \sum_{h \in H} \sum_{g \in G} \chi_g(s+h) |g\rangle \\
&= \frac{1}{\sqrt{G}} \sum_{g \in G} \left(\sum_{h \in H} \chi_g(h) \right) \chi_g(s) |g\rangle \quad \text{from Lecture 5: } \sum_{h \in H} \chi_g(h) = \begin{cases} \sharp H & \text{if } g \in H^\perp \\ 0 & \text{otherwise.} \end{cases} \\
&= \frac{1}{\sqrt{G}} \sum_{g \in H^\perp} \sharp H \, \chi_g(s) |g\rangle
\end{aligned}
$$

$\longrightarrow$ The quantum step before measurement is: $\sqrt{\frac{\sharp H}{\sharp G}} \sum_{g \in H^\perp} \chi_g(s) |g\rangle$

The quantum step before measurement is:

$$\sqrt{\frac{\sharp H}{\sharp G}} \sum_{g \in H^{\perp}} \chi_g(s)\,|g\rangle \quad \text{where } H^{\perp} = \{g \in G : \forall h \in H, \chi_g(h) = 1\}$$

$\longrightarrow$ Measuring gives a uniform $g \in H^{\perp}$ giving some information about $H \ldots$

repeating a **poly(log** $\sharp G$**)** times enables to recover H with high probability!

- For a rigorous proof of this statement: see Chapter 6 in the lecture notes by Andrew Childs

An example: Simon's problem

$$G = \mathbb{F}_2^n, \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{F}_2^n,\ \chi_{\mathbf{x}}(\mathbf{y}) = (-1)^{\mathbf{x} \cdot \mathbf{y}} \quad \text{and } H = \{\mathbf{0}, \mathbf{s}\}$$

$$\longrightarrow H^{\perp} = \left\{\mathbf{x} \in \mathbb{F}_2^n : \mathbf{x} \cdot \mathbf{s} = 0\right\}$$

In other words, we recover Simon's algorithm...

G is an Abelian group

Recall that we compute QFT_G as $\mathsf{QFT}_{\mathbb{Z}/n_1\mathbb{Z}} \otimes \cdots \otimes \mathsf{QFT}_{\mathbb{Z}/n_k\mathbb{Z}}$ where we used the isomorphism:

$$G \cong \mathbb{Z}/n_1\mathbb{Z} \times \cdots \times \mathbb{Z}/n_k\mathbb{Z} \tag{2}$$

But is it easy to compute this isomorphism/decomposition even if we “know” G?

G is an Abelian group

Recall that we compute $\mathbf{QFT}_G$ as $\mathbf{QFT}_{\mathbb{Z}/n_1\mathbb{Z}} \otimes \cdots \otimes \mathbf{QFT}_{\mathbb{Z}/n_k\mathbb{Z}}$ where we used the isomorphism:

$$G \cong \mathbb{Z}/n_1\mathbb{Z} \times \cdots \times \mathbb{Z}/n_k\mathbb{Z} \tag{2}$$

But is it easy to compute this isomorphism/decomposition even if we “know” G?

$\longrightarrow$ Yes! At least quantumly for a “good” definition of knowing G . . .

Quantum decomposition of Abelian groups:

Suppose we have (*i*) a unique encoding of each element of G, (*ii*) the ability to perform group operations on these elements, and (*iii*) a generating set for G.
Then, there exists an efficient quantum algorithm that decomposes G, namely outputs the isomorphism given in Equation (2)

$\longrightarrow$ See Chapter 6 in the lecture notes by Andrew Childs

To solve HSP we crucially used that we restrict ourself to the Abelian case...

$\big($in the Abelian case, $H^\perp$ gives linear relations enabling to recover $H\big)$

$\longrightarrow$ And the non-Abelian case?

No efficient algorithm is known for the non-Abelian case
(even if nothing indicates that it is impossible)...

$\longrightarrow$ Finding such an algorithm would have a huge impact in theoretical computer science, (post-quantum) cryptography...

If you are interested by this topic:

- Nice reading about Fourier transform (classical & quantum) over non-Abelian group: Chapter 11 in the lectures by Andrew Child `https://www.cs.umd.edu/~amchilds/qa/`
- The hidden nonabelian subgroup problem and the Kuperberg algorithm, see Chapters 11-13 in the lectures by Andrew Child `https://www.cs.umd.edu/~amchilds/qa/`

EXERCISE SESSION