Quantum algorithms

Quantum computing

G. Chênevert

M1 · S2 (2020)

Quantum algorithms

Faster computations

The whole point of quantum computing is that some quantum algorithms exhibit

quantum advantage: *i.e.* run "faster" than the best known classical algorithms

In extreme cases, this is expected to lead to

quantum supremacy: *i.e.* the ability to compute things that could never practically be achieved with classical computers.

Oct. 2019: Google claims achievement of quantum supremacy by sampling the output distribution of random quantum circuits on 53 qubits, a computation allegedly taking 10,000 years of classical computing (IBM says: 2.5 days)

Complexity of an algorithm

Classically: the complexity of an algorithm $\mathcal{A}$ is a bound on the number of computing steps needed for an input of a given size

i.e. the function

$$n \mapsto \max_{|x|=n} N_{\mathcal{A}}(x)$$

where $N_{\mathcal{A}}(x)$ denotes the number of steps used to perform $\mathcal{A}$ on x in a given computing model (usually: Turing machines)

Quantum computing model

There exist a (rather inconvenient) notion of quantum Turing machine

Most people prefer to use the (equivalent) **quantum circuit model**:

- given input x: a circuit $U_{\mathcal{A},x}$ made out of quantum gates taken from a standard universal set is prepared
- the output y is the result of the measure of $U_{\mathcal{A},x}|0\rangle$
- (then some classical post-processing is typically applied)

The **complexity** of the quantum algorithm is a bound on the number of gates needed:

$$n \mapsto \max_{|x|=n} |U_{\mathcal{A},x}|.$$

The Deutsch-Jozsa problem

Given a boolean function $f : \{0,1\}^n \to \{0,1\}$ assumed to be either

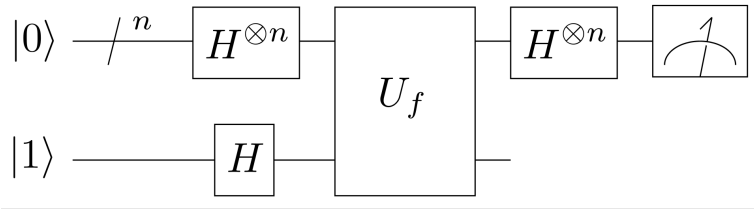
- constant: $f(x) \equiv 0$ or $f(x) \equiv 1$ ("type 0") or
- balanced: $|\{x \,|\, f(x) = 0\}| = |\{x \,|\, f(x) = 1\}| = 2^{n-1}$ ("type 1"),

problem: compute its type.

A classical algorithm needs at least $2^{n-1} + 1$ evaluations of f to decide

$\Longrightarrow$ **EXP** (EXPonential time) complexity class

The Deutsch-Jozsa algorithm



Proposition: output is $|0\rangle^{\otimes n} \iff f$ is constant

EQP (Exact Quantum Polynomial time) complexity class

$\implies$ exponential speedup !

Deutsch-Jozsa: remarks

- Here U_f is considered an **oracle** for f (black box implementation)
- Classical decision algorithm: $2^{n-1}+1$ evaluations are required to guarantee the answer... but we can get a probable answer with much less evaluations.
- With k evaluations, assuming constant and balanced functions are equiprobable:
  - if not all values are equal, f is certainly balanced
  - if all values are equal, f is constant with probability

$$\geq \frac{1}{1+2^{1-k}} \geq \frac{2}{3} \qquad \text{for } k \geq 2$$

Probabilistic algorithms

In practice: we prefer a fast algorithm with a good probability of giving a right answer to a slow algorithm that is always right!

Example: Rabin-Miller *vs* AKS primality tests

The Deutsch-Jozsa problem is in the

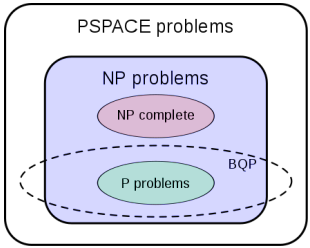
BPP (Bounded-error Probabilistic Polynomial time) complexity class

But since quantum algorithms are typically also probabilistic...

The speedup here is not so impressive after all!

Aside: the Complexity Zoo

- We know that **P** $\subseteq$ **NP** (Nondeterministic Polynomial time)
- Whether the inclusion is strict is an open question ($1,000,000)
- We also know that **P** $\subseteq$ **BPP** $\subseteq$ **BQP** (Bounded-error Quantum Polynomial time)
- Reverse inclusions *also unknown*; complexity classes are a real zoo



Quantum algorithms

Quantum advantage

Grover's algorithm

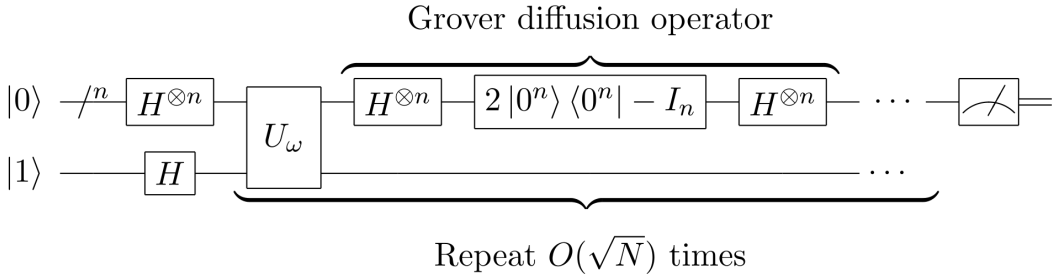
Towards Shor

Period detection

Shor's algorithm

Grover (1970)

Grover (1996)


Grover diffusion operator
$|0\rangle$
$|1\rangle$
n
$H^{\otimes n}$
H
U_ω
$H^{\otimes n}$
$2\,|0^n\rangle\langle 0^n| - I_n$
$H^{\otimes n}$
$\cdots$
$\cdots$
Repeat $O(\sqrt{N})$ times

Search problem

Suppose we have a decision function $f : X \to \{0, 1\}$ defined on a set X of size N.

The search problem defined by f is to find some $x \in X$ for which $f(x) = 1$.

Examples: database queries, factoring integers, bitcoin mining, ...

In the general (unstructured) case: a classical algorithm requires $\mathcal{O}(N)$ queries.

(Of course can do better if *e.g.* the data is sorted)

Grover's algorithm

Performs unstructured searches for arbitrary criteria in $\mathcal{O}(\sqrt{N})$ time.

$\Longrightarrow$ **quadratic speedup**

Works in two steps:

- phase inversion
- amplitude amplification

iterated a certain number of times

Phase inversion

Simplifying assumptions:

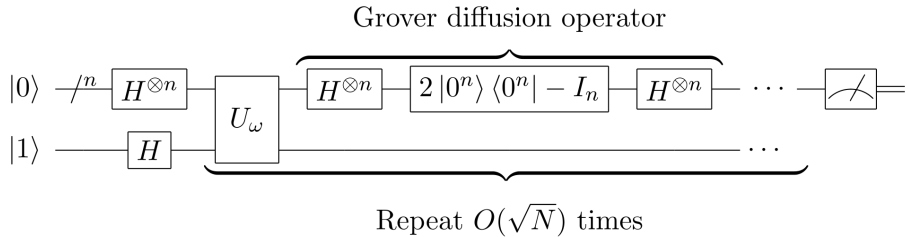
- $X = [\![0, N[\![$
- $N = 2^n$
- the equation $f(x) = 1$ admits a unique solution $\omega \in X$

So the problem is now: find $\omega \in X$ given access to a oracle for $f : [\![0, N[\![\to \{0, 1\}$

$$\text{where} \quad f(x) = \begin{cases} 1 & \text{if } x = \omega \\ 0 & \text{else.} \end{cases}$$

Phase inversion

ω is detected by inverting its phase: "$U_\omega|x\rangle = (-1)^{f(x)}|x\rangle$"



Actually

$$U_\omega\,|x\rangle \otimes |-\rangle = (-1)^{f(x)}\,|x\rangle \otimes |-\rangle$$

This is exactly what the oracle U_f does! So in fact $U_\omega = U_f$.

Amplitude amplification

The **Grover diffusion operator** G is

$$G = 2|s\rangle\langle s| - I$$

where

$$|s\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle.$$

Geometrical interpretation:

$$G|s\rangle = |s\rangle$$

$$G|\psi\rangle = -|\psi\rangle \qquad \text{when } \langle s|\psi\rangle = 0$$

$U_s = -G$ is a reflection through the hyperplane normal to $|s\rangle$

Amplitude amplification

Remark: U_ω is a reflection, too.

Actually U_ω acts on $\mathcal{V} = \mathcal{V}_N \otimes |-\rangle$ as

$$I - 2|\omega\rangle\langle\omega| = \mathrm{diag}(1, \ldots, \underbrace{-1}_{\omega}, \ldots, 1).$$

In general $I - 2|\psi\rangle\langle\psi|$ is a reflection through the hyperplane normal to $|\psi\rangle$.

GU_ω: unitary transformation of $\mathcal{V}$ that inverts every vector $|\psi\rangle$ orthogonal to both $|s\rangle$ and $|\omega\rangle$ – and acts as a rotation in the plan spanned by $|s\rangle$ and $|\omega\rangle$

Amplitude amplification

Consider unitary $|s'\rangle \sim |s\rangle - \langle\omega|s\rangle|\omega\rangle$, and write $\langle\omega|s\rangle = \frac{1}{\sqrt{N}} = \sin\frac{\theta}{2}$.

Initial state:

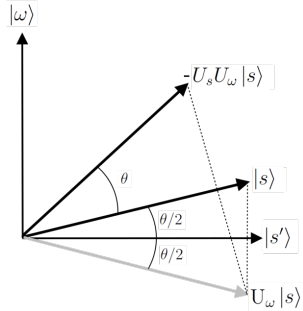
$$|\psi\rangle = |s\rangle = \cos\tfrac{\theta}{2}\,|s'\rangle + \sin\tfrac{\theta}{2}\,|\omega\rangle$$

GU_ω is a rotation of θ (exercise!), so after k iterations:

$$(GU_\omega)^k|\psi\rangle = \cos(\tfrac{\theta}{2} + k\theta)\,|s'\rangle + \sin(\tfrac{\theta}{2} + k\theta)\,|\omega\rangle$$

$$\mathbb{P}\big[\,\mathcal{M}(GU_\omega)^k|\psi\rangle = |\omega\rangle\,\big] = \sin^2(\tfrac{\theta}{2} + k\theta)$$

Optimal number of iterations



Each iteration brings the state closer to $|\omega\rangle$ by an angle of $\theta = 2 \arcsin \frac{1}{\sqrt{N}}$.

Until it starts moving away... Sage visualization

Optimal number of iterations

So, in order to maximize the probability of measuring $|\omega\rangle$, take

$$(k + \tfrac{1}{2})\theta \approx \frac{\pi}{2} \qquad \Longleftrightarrow \qquad k \approx \frac{\pi}{2\theta} - \frac{1}{2}$$

When N is large (interesting case!) we have $\theta \approx \sin\theta = \dfrac{2}{\sqrt{N}}$

so the optimal number of iterations is $k \approx \dfrac{\pi\sqrt{N}}{4}$.

Closely related to this rather surprising way to approximate π !

Implementation of G

$$G = 2|s\rangle\langle s| - I$$

- G is more easily computed if we change the basis:

$$G = H^{\otimes n} \otimes \underbrace{(2|0\rangle\langle 0| - I)}_{G_0} \otimes H^{\otimes n}$$

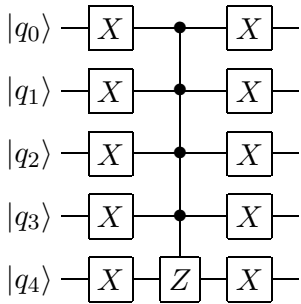
- $G_0 \sim -G_0 = U_0 = \mathrm{diag}(-1, 1, \ldots, 1)$:

$$U_0|x\rangle = \begin{cases} -|x\rangle & \text{if } x = 0 \\ |x\rangle & \text{if } x \neq 0. \end{cases}$$

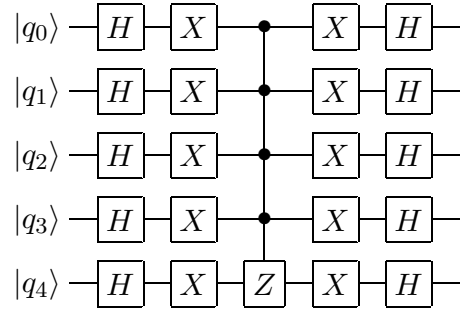
Implementation of G

Example: with $n = 4$

G_0:

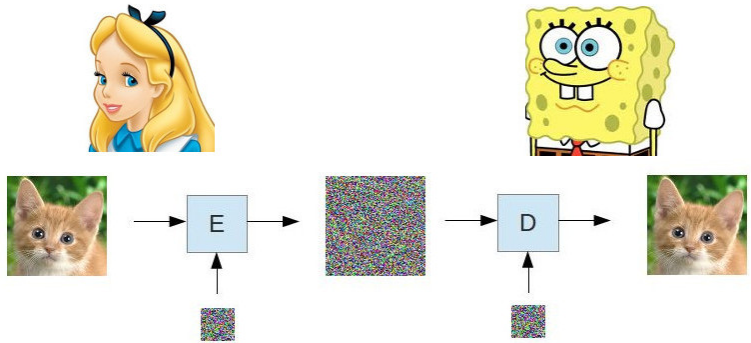


G:



Application: breaking cryptography

Alice sends secret messages to Bob:



Application: breaking cryptography

They agree on a secret n-bit encryption key k.

Alice encrypts her messages with k:

$$c = E(k, m)$$

and Bob decrypts them using the same k:

$$m = D(k, c).$$

Known plaintext attack

Imagine the attacker, Eve, knows the message m corresponding to *one* ciphertext c.

Then she can try to recover the secret key k in order to be able to decrypt *all* messages exchanged by Alice and Bob.

(A plausible scenario: this is exactly what happened with Enigma during WWII).

This is a search problem: she's looking for $k \in [\![0, 2^n[\![$ for which $D(k, c) = m$.

Brute force attack on the key

Suppose $n = 128$ (today's standard for AES).

With a classical computer: Eve will need to go through the 2^{128} possibilities: impractical for at least the next 30 years.

$\Longrightarrow$ secure communication ✓

With a quantum computer running Grover's algorithm: Eve will recover the key in $\sqrt{2^{128}} = 2^{64}$ steps: doable today using specialized hardware.

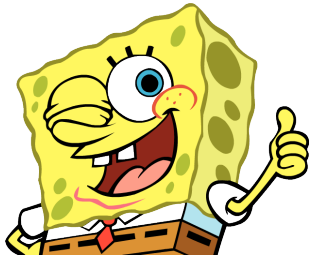
$\Longrightarrow$ no more confidentiality ✗

Solution

In case this happens:

"post-quantum symmetric cryptography": just move to 256-bit keys

No biggie!



Quantum algorithms

Quantum circuits

In the end, a quantum circuit is just a big unitary matrix.

n qubits: $2^n \times 2^n$ unitary matrix

Things we can implement using unitary matrices:

- reflections
- rotations
- ...
- **Fourier transforms**

Recall: Discrete Fourier Transform

N-point Fourier transform of a sequence $x[0], \ldots, x[N-1]$:

$$y[k] = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{-\frac{2\pi ijk}{N}} x[j]$$

Matrix formulation:

$$\mathbf{y} = \mathcal{F}\,\mathbf{x} \qquad \text{with} \qquad \mathcal{F} = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & \zeta & \zeta^2 & \ldots & \zeta^{N-1} \\ \vdots & \vdots & & & \vdots \\ 1 & \zeta^{N-1} & \zeta^{2(N-1)} & \ldots & \zeta^{(N-1)(N-1)} \end{bmatrix}$$

where ζ is *some* primitive N-th root of unity

Discrete Fourier Transform

Special case: $N = 2$

$$\mathcal{F} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = H \quad (!)$$

Inverse Fourier transform:

$$\mathcal{F}^{-1} = \mathcal{F}^{*} = \mathcal{F}^{\dagger}$$

Fourier transforms are unitary

Quantum Fourier Transform

Suppose we have a quantum state $|\psi\rangle \in \mathcal{V}_N$:

$$|\psi\rangle = \sum_{x<N} \alpha_x |x\rangle$$

Its **Fourier transform** is the state

$$\mathcal{F}|\psi\rangle = \sum_{y<N} \beta_y |y\rangle$$

defined by

$$\beta_y = \frac{1}{\sqrt{N}} \sum_{x<N} \zeta^{xy} \alpha_x.$$

Quantum Fourier Transform

In other words: from a theoretical point of view

QFT of a state = DFT of the probability amplitudes

Often written in the equivalent form:

$$\mathcal{F}\,|x\rangle = \frac{1}{\sqrt{N}} \sum_{y<N} \zeta^{xy}\,|y\rangle$$

Naive classical algorithm: $\mathcal{O}(N^2)$ operations

Cooley-Tukey (1965): *Fast Fourier Transform* $\mathcal{O}(N \log N)$ operations

Quantum Fourier Transform

Theorem

There exists a quantum circuit with $\mathcal{O}((\log N)^2)$ gates that computes the QFT.

For $N = 2^n$, a circuit with $\mathcal{O}(n^2)$

- Hadamard gates $H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
- controlled phase shifts $R_m = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{2\pi i}{2^m}} \end{bmatrix}$
- swaps

suffices.

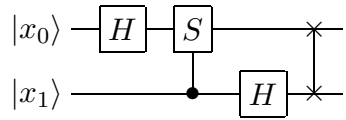
Small values of n

$n = 0$: $F = I$ ✓

$n = 1$: $F = H$ ✓

$n = 2$:

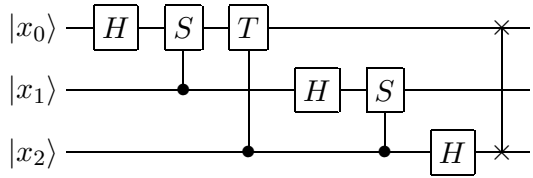
$$F = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix} =$$



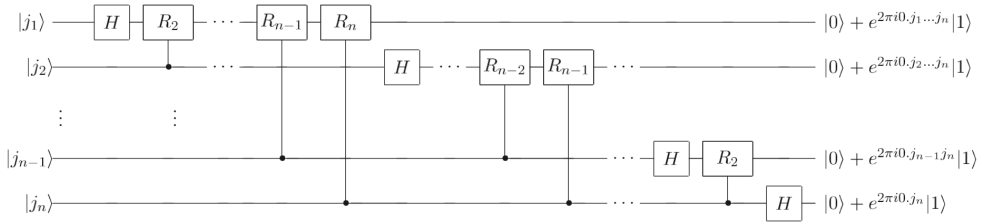
✓

Small values of n

$n = 3$:



General QFT circuit



- n Hadamard gates
- $1 + 2 + \cdots + (n-1) = \binom{n}{2}$ controlled phase shifts
- $\leq \binom{n}{2}$ swaps

Quantum algorithms

Quantum advantage

Grover's algorithm

Towards Shor

Period detection

Shor's algorithm

Recall: QFT

Fix $N = 2^n$.

The N-point (quantum) Fourier Transform

$$\mathcal{F}|x\rangle = \frac{1}{\sqrt{N}} \sum_{y<N} \zeta^{xy} |y\rangle, \qquad \zeta^N = 1 \text{ primitive}$$

is computable with a quantum circuit of size $\mathcal{O}(n^2)$.

Application: Period detection

Suppose $f : \mathbb{Z}_N \to \mathbb{Z}_N$ is p-periodic:

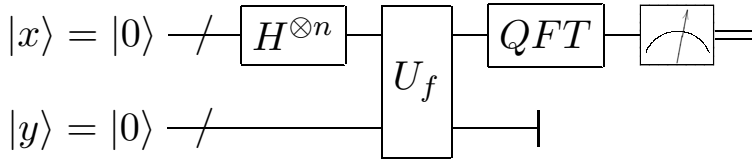
$$f(x + p) = f(x) \qquad \text{for all } x.$$

Problem: find p.

A special case of the **hidden subgroup problem**.

Idea: we can detect the period using a Fourier transform.

Quantum period detection



Theorem

If f is p-periodic, then a multiple of $\frac{N}{p}$ is measured.

Remark: for the moment we are assuming that $p \mid N$ (else replace p by $\mathrm{GCD}(p, N)$)

Proof in the $p \mid N$ case

Write $N = pq$.

Evolution of the quantum state:

$$|0\rangle \otimes |0\rangle \overset{H^{\otimes n}}{\mapsto} \frac{1}{\sqrt{N}} \sum_{x<N} |x\rangle \otimes |0\rangle \overset{U_f}{\mapsto} \frac{1}{\sqrt{N}} \sum_{x<N} |x\rangle \otimes |f(x)\rangle$$

$$\overset{\text{QFT}}{\mapsto} \frac{1}{N} \sum_{x<N} \sum_{y<N} \zeta^{xy} |y\rangle \otimes |f(x)\rangle$$

Now write $x = j + kp$, so that $f(x) = f(j)$.

Proof in the $p \mid N$ case

$$|\psi\rangle = \frac{1}{N} \sum_{j<p} \sum_{k<q} \sum_{y<N} \zeta^{(j+kp)y} |y\rangle \otimes |f(j)\rangle = \frac{1}{N} \sum_{y<N} \sum_{j<p} \zeta^{jy} \underbrace{\sum_{k<q} (\zeta^{py})^k}_{\text{0 unless } \zeta^{py}=1} |y\rangle \otimes |f(j)\rangle$$

since

$$\sum_{k<q} (\zeta^{py})^k = \begin{cases} \dfrac{1-(\zeta^{py})^q}{1-\zeta^{py}} = \dfrac{1-1}{1-\zeta^{py}} = 0 & \text{if } \zeta^{py} \neq 1 \\ \displaystyle\sum_{k<\frac{N}{p}} 1 = q & \text{if } \zeta^{py} = 1 \text{ i.e. } q \mid y \end{cases}$$

In the end we have

$$|\psi\rangle = \frac{1}{p} \sum_{r<p} \sum_{j<p} (\zeta^q)^{jr} |rq\rangle \otimes |f(j)\rangle$$

Proof in the $p \mid N$ case

$$|\psi\rangle = \frac{1}{p} \sum_{r<p} \sum_{j<p} (\zeta^q)^{jr} |rq\rangle \otimes |f(j)\rangle$$

Only values of y of the form rq have a non-zero probability of being measured:

$$\mathbb{P}\big[\, y = rq \,\big] = \frac{1}{p^2} \left\| \sum_{j<p} (\zeta^q)^{jr} |f(j)\rangle \right\|^2$$

In particular, in the special case where all values $f(j)$ are distinct:

$$\mathbb{P}\big[\, y = rq \,\big] = \frac{1}{p^2} \sum_{j<p} |(\zeta^q)^{jr}|^2 = \frac{1}{p^2} \sum_{j<p} 1 = \frac{1}{p}$$

Approximate period detection

Now suppose $f : [\![0, N[\![\to [\![0, N[\![$ is **almost** p-**periodic**:

$$f(x + p) = f(x) \qquad \text{for all } 0 \leq x < N - p.$$

Theorem

If f is almost p-periodic, then an approximate multiple of $\frac{N}{p}$ is probably measured.

And then we can (probably) recover p with classical post-processing.

Analysis in the general case

Write $N = pq + a$ with $0 \leq a < p$. Everything is the same until

$$|\psi\rangle = \frac{1}{N} \sum_{y<N} \sum_{j<p} \zeta^{jy} \sum_{k<q_j} (\zeta^{py})^k |y\rangle \otimes |f(j)\rangle$$

$$\text{with } q_j = \begin{cases} q+1 & \text{if } 0 \leq j < a \\ q & \text{if } a \leq j < p \end{cases}$$

$$\sum_{k<q_j} (\zeta^{py})^k = \frac{1-(\zeta^{py})^{q_j}}{1-\zeta^{py}} = \zeta^{\frac{py(q_j-1)}{2}} S_{q_j}\left(y\tfrac{p}{N}\right) \quad \text{where} \quad S_\alpha(x) = \begin{cases} \dfrac{\sin(\alpha\pi x)}{\sin(\pi x)} & x \notin \mathbb{Z} \\ \alpha & x \in \mathbb{Z} \end{cases}$$

Analysis in the general case

$$|\psi\rangle = \frac{1}{N} \sum_{y<N} \sum_{j<p} \zeta^{jy + \frac{py(q_j-1)}{2}} S_{q_j}\left(y\,\tfrac{p}{N}\right)\ |y\rangle \otimes |f(j)\rangle$$

To simplify: under the assumption that all the p values $f(j)$ are distinct:

$$\mathbb{P}\big[\,y\,\big] = \frac{1}{N^2}\left(a\,S_{q+1}\left(y\,\tfrac{p}{N}\right)^2 + (p-a)\,S_q\left(y\,\tfrac{p}{N}\right)^2\right)$$

cf. Sage visualization

Analysis in the general case

Proposition

The probability that $\lfloor r\frac{N}{p} \rfloor$ or $\lceil r\frac{N}{p} \rceil$ is measured is asymptotically $\geq \frac{4}{\pi^2} \approx 40\%$

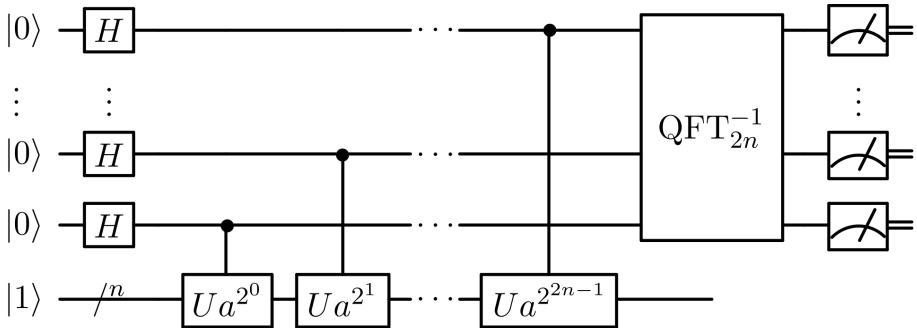
with r uniformly distributed among $[\![0, p[\![$.

Thus probability $.4\left(1 - \frac{1}{p}\right)$ of getting a "good" result y.

Fact: If $N > 2p^2$, the period p can be efficiently recovered since $\frac{r}{p}$ appears in lowest terms in the continued fraction expansion of $\frac{y}{N}$ and if p is large, r and p will most likely be coprime (probability $\approx \frac{1}{\log \log p}$ of failure).

Quantum algorithms

Shor's algorithm (1994)



Shor's algorithm

(Probably) factors an integer N with quantum complexity

$$\mathcal{O}\big((\log N)^2(\log\log N)(\log\log\log N)\big)$$

This is much better than the best currently known classical algorithm

that has complexity

$$\mathcal{O}\big(e^{1.9(\log N)^{\frac{1}{3}}(\log\log N)^{\frac{2}{3}}}\big)$$

subexponential speedup

NB: $\log N$ is the natural parameter to measure the size of an integer $N \implies$ **BQP**

Factoring vs period finding

Suppose $N = pq$ with p and q two distinct prime numbers.

Theorem (Euler)

For any integer a coprime with N, we have $a^r \equiv 1 \bmod N$ for

$$r = \varphi(N) = (p-1)(q-1).$$

In other words: r is a period for the function $f : x \mapsto a^x \bmod N$.

BPP reduction from factoring to period finding

To factor $N = pq$:

- pick a random integer $a \in [\![0, N[\![$
- with high probability $\mathrm{GCD}(a, N) = 1$
- if $x \mapsto a^x \bmod N$ has *odd* period, pick another a
- so now we have an integer a of even order $2r$: $a^{2r} \equiv 1 \bmod N$

$$N \mid a^{2r} - 1 = (a^r - 1)(a^r + 1)$$

- there is a 50% chance that $\mathrm{GCD}(N, a^r \pm 1)$ are non-trivial divisors of N

Small example: $N = 21$

Try $a = 4$:

$$4 \quad \to \quad 4^2 = 16 \quad \to \quad 4^3 \equiv 1 \qquad \text{✗}$$

Try $a = 5$:

$$5 \quad \to \quad 5^2 \equiv 4 \quad \to \quad 5^3 \equiv 20 \quad \to \quad 5^4 \equiv 16 \quad \to \quad 5^5 \equiv 17 \quad \to \quad 5^6 \equiv 1$$

but $\mathrm{GCD}(5^3 - 1, 21) = 1$, $\mathrm{GCD}(5^3 + 1, 21) = 21$ ✗

Try $a = 2$:

$$2 \quad \to \quad 2^2 = 4 \quad \to \quad 2^3 = 8 \quad \to \quad 2^4 = 16 \quad \to \quad 2^5 \equiv 11 \quad \to \quad 2^6 \equiv 1$$

and $\mathrm{GCD}(2^3 - 1, 21) = 7$, $\mathrm{GCD}(2^3 + 1, 21) = 3$ ✓

Implementation

So we need a quantum implementation U_f of the function

$$f(x) = a^x \bmod N.$$

Shor picks $Q = 2^n > N^2$ in order to be able to apply postprocessing and considers the function $f(x)$ for $x \in [\![0, Q[\![$.

If x is written in binary:

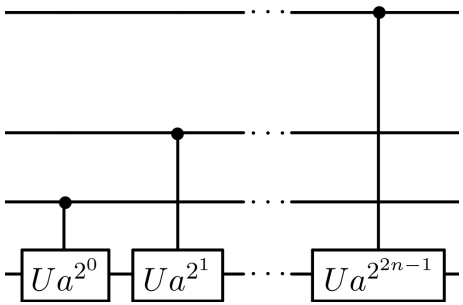
$$x = 2^{n-1} b_{n-1} + \cdots + 2^1 b_1 + 2^0 b_0$$

then

$$a^x = (a^{2^{n-1}})^{b_{n-1}} \cdots (a^{2^1})^{b_1} \cdot (a^{2^0})^{b_0}$$

Implementation

Thus we would only need to implement "multiplication by a^{2^k} mod N"



This is actually the difficult part. One approach is to translate *reversible* classical arithmetical circuits into quantum circuits + classical pre-processing

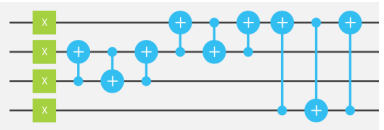
Quantum factoring: summary

To find a nontrivial factor of N:

- Pick $Q = 2^n$ large enough
- Choose a coprime with N randomly
- Implement modular exponentiation $f(x) = a^x \bmod N$ as a quantum circuit
- Apply QFT + classical post-processing to recover period R of f
- Repeat until $R = 2r$ is even and $\mathrm{GCD}(a^r - 1, N) \neq 1, N$
- Output $\mathrm{GCD}(a^r - 1, N)$

Lab 4: Factoring 15 (or pretending to) – due March 20

1. What are the *good* values of a that can be used to factor 15 ? What is the period of f in each case?
2. Verify that the following circuit acts as multiplication by 7 mod 15 on 4 qubits:



3. Can you come up with a (simple) circuit that acts as multiplication by 7^2? By 7^4?

   Controlled versions of these circuits would be needed to implement Shor's algorithm with $a = 7$ ...

4. Since arithmetic is hard: implement quantum period finding for the function $f(x) = x \bmod 2$, $x \in [\![0, 16[\![$ (5 qubits suffice). You will need to use a circuit for the 4-qubit QFT. (Bonus: what about $f(x) = x \bmod 3$?)