Quantum computing

Fault-tolerant quantum computing

Federico Meloni (DESY)
BND Graduate School 2023

10/08/2023

Who am I?

Data analysis

Reconstruction techniques

Detector instrumentation

2011 - 2014 PhD studies
University of Milano

2018 - current Staff scientist
DESY

Time

2014 - 2017 Postdoctoral researcher
2017 - 2018 "Oberassistent"
University of Bern

1 year

Today

Software and computing

Phenomenology

Education

Introduction

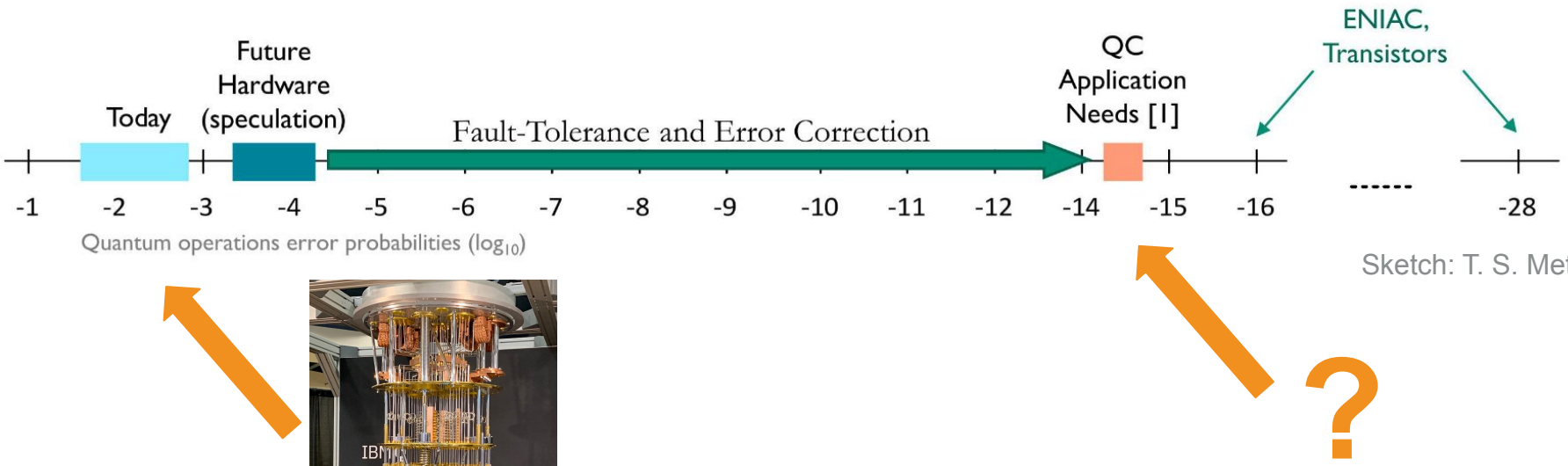
A useful Quantum Computer needs to be:

- **Universal and General Purpose**: not limited to a single class of problems
- **Accurate**: probability of error on the output can be arbitrarily small
- **Scalable**: resource requirements do not grow exponentially in the size of target error probability of the computation

Building Quantum Computers is challenging:

- Quantum information is inherently fragile
- Not a pure engineering problem: understanding the underlying physics still matters a lot!
- Current devices are noisy. We don't expect quantum devices to be as good as classical transistors for information processing

Where are we?



Fault-Tolerant quantum computers aim to achieve a useful quantum computer given imperfect devices underneath

- Using encoded quantum data will require O(1000) more physical qubits
- The small devices available today serve as demonstrators for theoretical concepts (e.g., error correction) applicable to more reliable platforms

Fault-tolerant classical computing

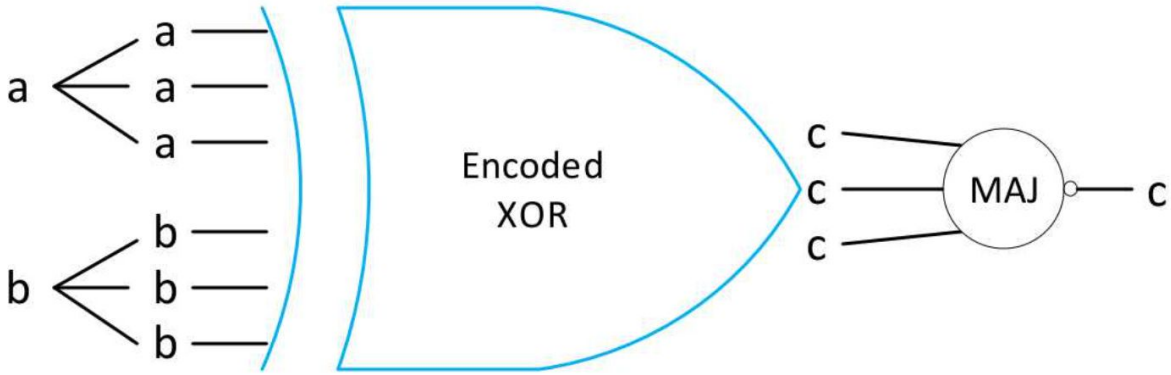
A fault-tolerant computing protocol maintains general purpose computations efficiently in the presence of faults during the computation

Computational Model: circuits in which **each gate has exactly one output**

Noise Model: ideal gates followed by a **bit flip with probability p**

Goal: **approximate the ideal circuit** to precision ε using faulty gates

Approach: **encode the data and process it with encoded gates** which suppress the spread of errors



The classical threshold theorem

A g-gate ideal circuit can be simulated to precision ε by an O(g log (g/ε))-gate faulty circuit

von Neumann,

Automata Studies. (AM-34), 1956

- As long as gate error $p < p_c$, the accuracy threshold for classical computation

The quantum threshold theorem

A quantum circuit is fault-tolerant against t failures if failures in t elements results in at most t errors per code block (group of qubits corrected together)

There exists a physical error probability p_c below which an arbitrary quantum computation can be performed efficiently

- 2-input gate accuracy threshold [0802.1464]
- At k levels of encoding, the effective error rate P_L scales as $p_c\,(p/p_c)^{2k}$. For a computation of length N, we need log (log N) levels of encoding

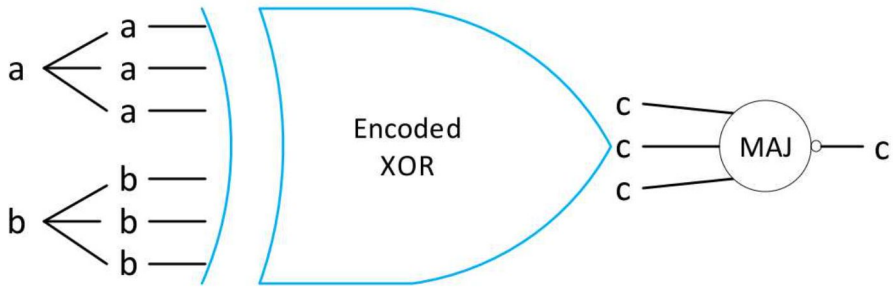
How can we achieve fault tolerant QC?

A series of problems seem to prevent the possibility of fault-tolerant quantum computing:

- The no-cloning theorem
- The collapse of the state after measurement
- Unitary operations are continuous (not discrete)

Despite these problems, **quantum error correction is possible** (I'll give just one example, you'll see more details in Jeanette's lecture tomorrow)

- We can use methods from classical error correction



Classical bit flips

We can use redundancy to code the information

$$0 \rightarrow 000$$

$$1 \rightarrow 111$$

We use majority voting to “correct” errors

$$000, 001, 010, 100 \rightarrow 000$$

$$111, 110, 101, 011 \rightarrow 111$$

In this way, we can correct errors that affect only a single bit

Quantum bit flips

We can extend the previous idea to the quantum domain

We use three qubits to code one

$$|0\rangle \rightarrow |000\rangle$$

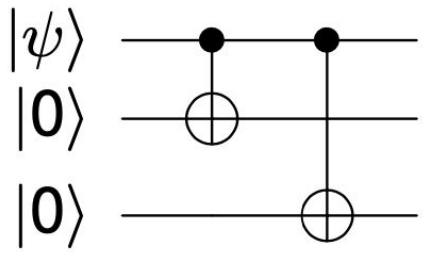
$$|1\rangle \rightarrow |111\rangle$$

By linearity

$$\alpha\,|0\rangle + \beta\,|1\rangle \rightarrow \alpha\,|000\rangle + \beta\,|111\rangle$$

It does NOT violate the no-cloning theorem

The circuit for encoding is simple



Detecting and correcting bit flips

We can detect qubit flips without measuring them by using additional qubits

$$|000\rangle, |111\rangle \rightarrow |00\rangle \qquad |001\rangle, |110\rangle \rightarrow |01\rangle$$

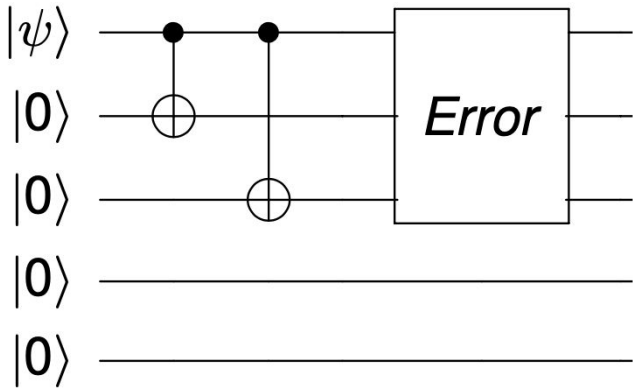
$$|010\rangle, |101\rangle \rightarrow |10\rangle \qquad |100\rangle, |011\rangle \rightarrow |11\rangle$$

Detecting and correcting bit flips

We can detect qubit flips without measuring them by using additional qubits

$$|000\rangle, |111\rangle \rightarrow |00\rangle \qquad |001\rangle, |110\rangle \rightarrow |01\rangle$$

$$|010\rangle, |101\rangle \rightarrow |10\rangle \qquad |100\rangle, |011\rangle \rightarrow |11\rangle$$

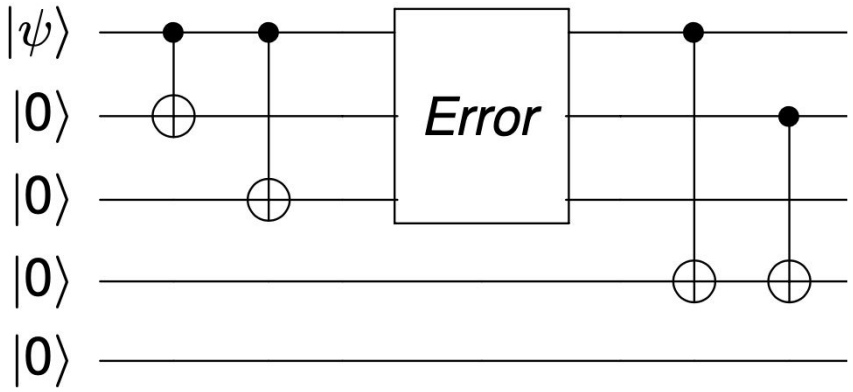


Detecting and correcting bit flips

We can detect qubit flips without measuring them by using additional qubits

$$|000\rangle, |111\rangle \rightarrow |00\rangle \qquad |001\rangle, |110\rangle \rightarrow |01\rangle$$

$$|010\rangle, |101\rangle \rightarrow |10\rangle \qquad |100\rangle, |011\rangle \rightarrow |11\rangle$$

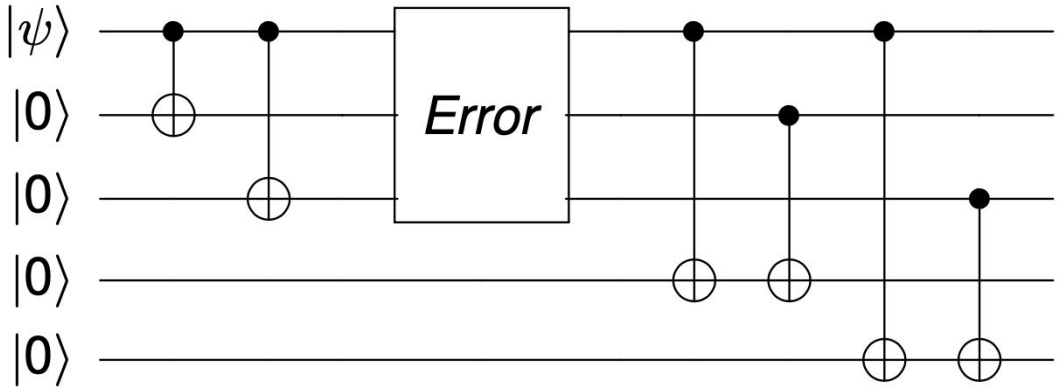


Detecting and correcting bit flips

We can detect qubit flips without measuring them by using additional qubits

$$|000\rangle, |111\rangle \rightarrow |00\rangle \qquad |001\rangle, |110\rangle \rightarrow |01\rangle$$

$$|010\rangle, |101\rangle \rightarrow |10\rangle \qquad |100\rangle, |011\rangle \rightarrow |11\rangle$$

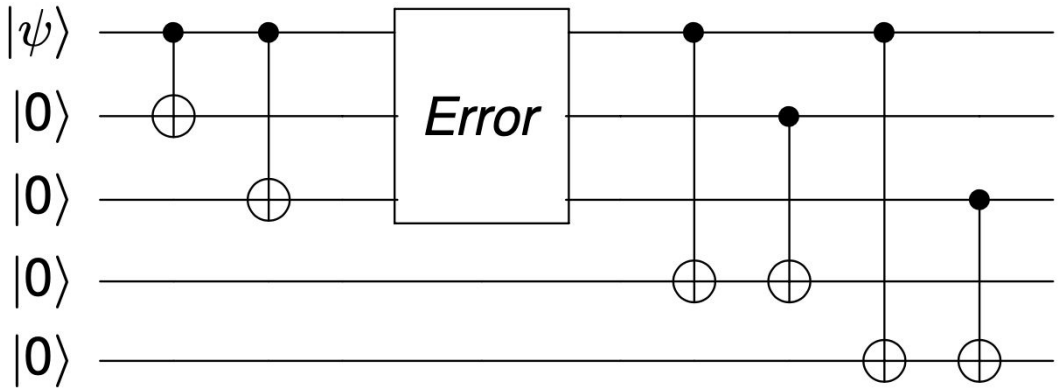


Detecting and correcting bit flips

We can detect qubit flips without measuring them by using additional qubits

$$|000\rangle, |111\rangle \rightarrow |00\rangle \qquad |001\rangle, |110\rangle \rightarrow |01\rangle$$

$$|010\rangle, |101\rangle \rightarrow |10\rangle \qquad |100\rangle, |011\rangle \rightarrow |11\rangle$$



We can measure the additional qubits and apply an error correction operation:

|00> = all good |01> = invert the third qubit

|10> = invert the second qubit |11> = invert the first qubit

Quantum Fourier Transform

QFT != Quantum Field Theory

Now, let's assume we had an ideal quantum computer. What could we do with it?

The Quantum Fourier Transform (QFT) is widely used in quantum computing

A general quantum state $|\varphi\rangle$ on n qubits can be written

$$|\varphi\rangle = \sum_{j=0}^{2^n-1} a_j |j\rangle_n = \sum_{j=0}^{N-1} a_j |j\rangle_n \qquad \text{for } N = 2^n$$

There are N amplitudes a_j corresponding to the N standard basis kets $|j\rangle$

For a fixed $|\varphi\rangle$, we get a complex-valued function where $a(j)=a_j$, with $1 = \sum_{j=0}^{N-1} |a_j|^2$

The quantum Fourier Transform of $|\varphi\rangle$ is

$$\mathbf{QFT}_n : |\varphi\rangle = \sum_{j=0}^{N-1} a_j |j\rangle_n \rightarrow \sum_{j=0}^{N-1} b_j |j\rangle_n \quad \text{with} \quad b_j = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_k \omega^{jk} \quad \text{and} \quad \omega = e^{\frac{2\pi i}{N}}$$

In matrix form

$$\mathbf{QFT}_n = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \cdots & \omega^{2(N-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \cdots & \omega^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \omega^{3(N-1)} & \cdots & \omega^{(N-1)(N-1)} \end{bmatrix}$$

In matrix form (after some massaging)

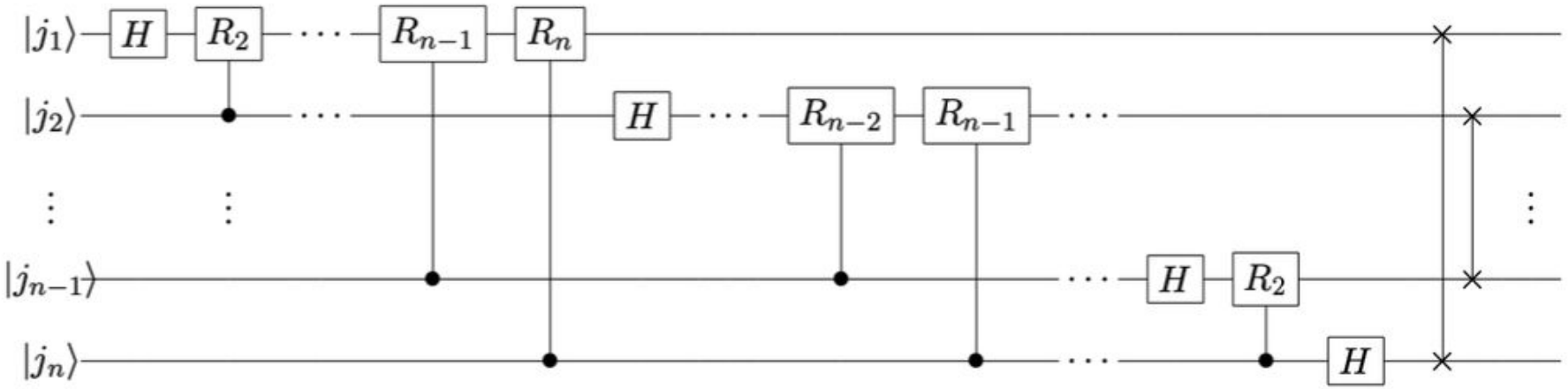
$$\mathbf{QFT}_n = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \cdots & \omega^{N-2} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \cdots & \omega^{N-3} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{N-2} & \omega^{N-3} & \cdots & \omega \end{bmatrix}$$

Interesting fact: for n = 1

$$\mathrm{QFT}_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & (-1)^1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \mathrm{H}$$

but this is **not true in general**!

QFT circuit



The circuit in the figure implements the QFT

- The R gates in the circuit are what we call $R_Z(2\pi/2^k)$
- The number of gates is **quadratic in m**, an exponential speed-up over the classical case (FFT)

Two properties of the QFT will be very useful in a moment:

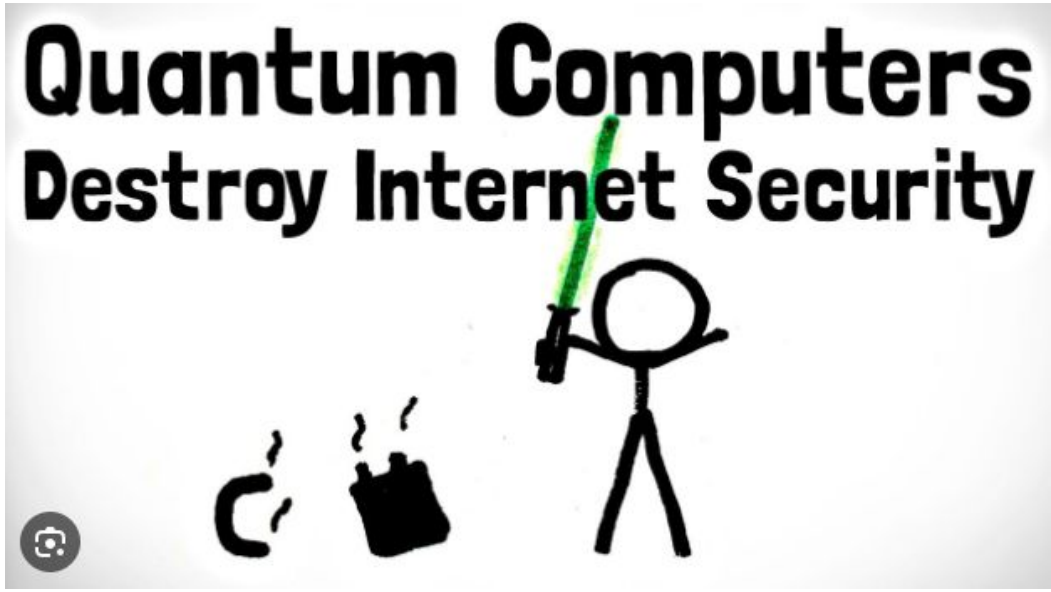
- Shift-invariance (up to an unobservable phase)
- QFT transforms sequences with period r into sequences with period M/r (where $M = 2^m$)

Shor’s algorithm

Introduction

Shor's algorithm is, probably, the most famous quantum algorithm

- It finds a factor of a n-bit integer in time $O(n^2(\log n)(\log\log n))$
- The best classical algorithm that we know of for the same task needs time $O(e^{cn/3\ (\log n)2/3})$
- Dramatic consequences for current cryptography (RSA)



Drawing taken from: @MinutePhysics

How does RSA work?

RSA is a security protocol to allow others to send you secure communications.

- You publish a public key used to encrypt these messages intended for you. Anyone who has access to the key can use it.
- There is an additional key, your private key. You and only you have it. With it you can decrypt and read the encrypted messages.

Public key: a pair of numbers (e, n), with n an integer (product of two primes)

Private key: a pair of numbers (d, n), with the same n as the public key

$$\left(m^e\right)^d \equiv m(\mathrm{mod\ n})$$

Knowing e and n, or even m, it can be extremely difficult to find d

Example (from wikipedia)

1. Choose two distinct prime numbers (p=61 and q=53), n = pq = 3233.

Example (from wikipedia)

1. Choose two distinct prime numbers (p=61 and q=53), n = pq = 3233.
2. Compute λ(n) = lcm(p − 1, q − 1) = lcm(60, 52) = 780

Example (from wikipedia)

1. Choose two distinct prime numbers (p=61 and q=53), n = pq = 3233.
2. Compute λ(n) = lcm(p − 1, q − 1) = lcm(60, 52) = 780
3. Choose any number 1 < e < 780 that is coprime to 780.
   If we choose a prime number, we only have to check it is not a divisor of 780.
   Let e=17.

Example (from wikipedia)

1. Choose two distinct prime numbers (p=61 and q=53), n = pq = 3233.
2. Compute λ(n) = lcm(p − 1, q − 1) = lcm(60, 52) = 780
3. Choose any number 1 < e < 780 that is coprime to 780.
   If we choose a prime number, we only have to check it is not a divisor of 780.
   Let e=17.
4. Compute d, the modular multiplicative inverse of e (mod λ(n)), d = 413

$$1 = (17 \times 413) \bmod 780$$

Example (from wikipedia)

1. Choose two distinct prime numbers (p=61 and q=53), n = pq = 3233.
2. Compute λ(n) = lcm(p − 1, q − 1) = lcm(60, 52) = 780
3. Choose any number 1 < e < 780 that is coprime to 780.
   If we choose a prime number, we only have to check it is not a divisor of 780.
   Let e=17.
4. Compute d, the modular multiplicative inverse of e (mod λ(n)), d = 413

$$1 = (17 \times 413) \bmod 780$$

The public key is (n = 3233, e = 17). The encryption function is

$$c(m) = m^e \bmod n = m^{17} \bmod 3233$$

The private key is (n = 3233, d = 413). The decryption function is

$$m(c) = c^d \bmod n = c^{413} \bmod 3233$$

Example (from wikipedia)

1. Choose two distinct prime numbers (p=61 and q=53), n = pq = 3233.
2. Compute λ(n) = lcm(p − 1, q − 1) = lcm(60, 52) = 780
3. Choose any number 1 < e < 780 that is coprime to 780.
   If we choose a prime number, we only have to check it is not a divisor of 780.
   Let e=17.
4. Compute d, the modular multiplicative inverse of e (mod λ(n)), d = 413

$$1 = (17 \times 413) \bmod 780$$

The public key is (n = 3233, e = 17). The encryption function is

$$c(m) = m^e \bmod n = m^{17} \bmod 3233$$

The private key is (n = 3233, d = 413). The decryption function is

$$m(c) = c^d \bmod n = c^{413} \bmod 3233$$

For the message m = 65: $c(m) = 65^{17} \bmod 3233 = 2790$

$m(c) = 2790^{413} \bmod 3233 = 65$

What if we could compute your private key from the public key?

The algorithm

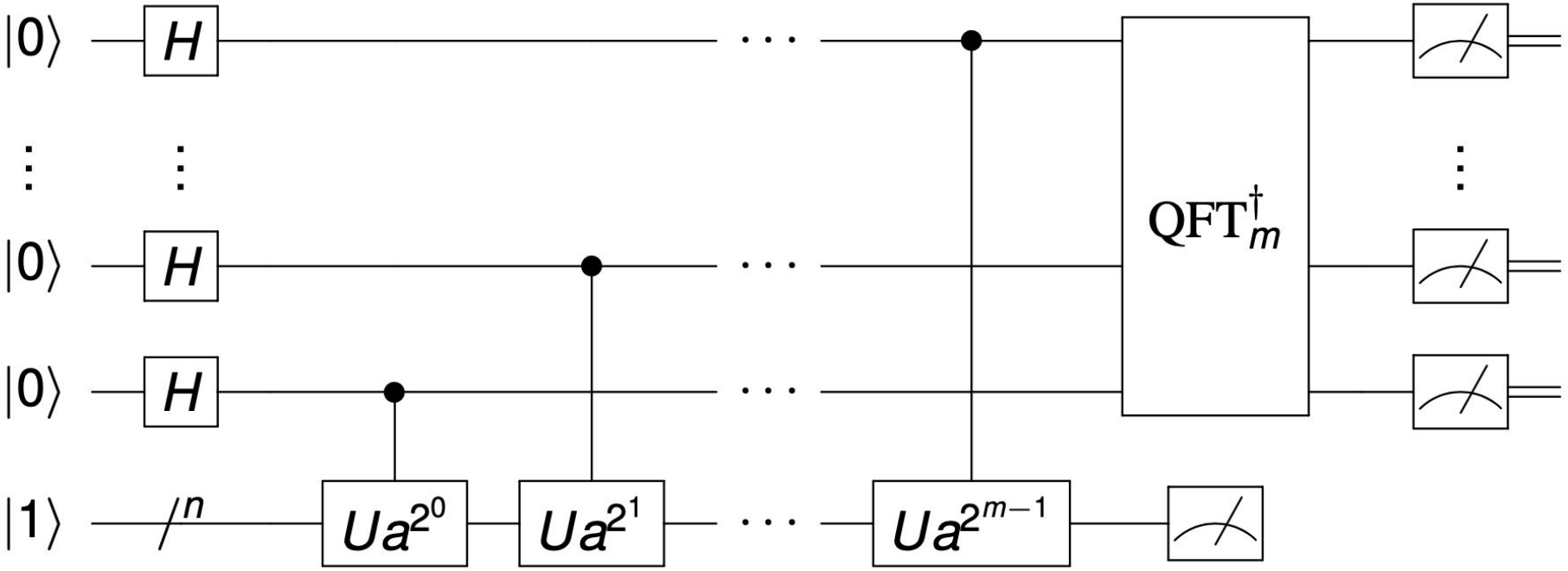
1. Given N, check that N is not a prime or power of a prime. If it is, stop.
2. Choose $1 < a < N$ at random
3. If $b = \gcd(a,N) > 1$, output b and stop
4. Find the order of a mod N ($r > 0$ such that $a^r \equiv 1 \bmod N$)
5. If r is odd, go to 2
6. Compute

$$x = a^{r/2} + 1 \bmod N \qquad y = a^{r/2} - 1 \bmod N$$

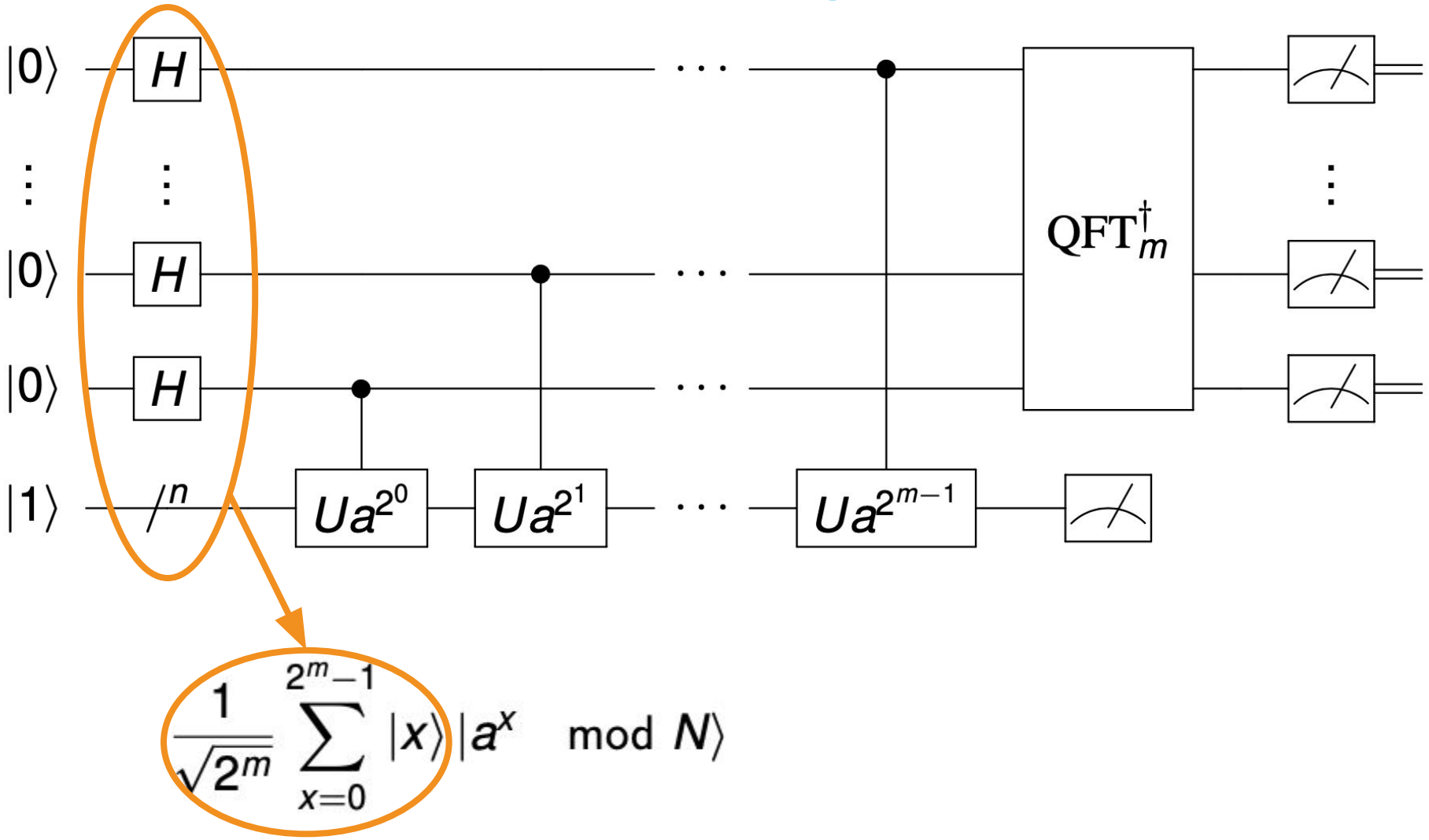
7. If $x = 0$, go to 2. If $y = 0$, take $r = r/2$ and go to 5.
8. Compute $p = \gcd(x,N)$ and $q = \gcd(y,N)$. At least one of them will be a non-trivial factor of N

Every step, apart from 4 can be carried out efficiently on a classical computer. For step 4, there exists a circuit with a number of gates which is polynomial in n (the number of bits of N)

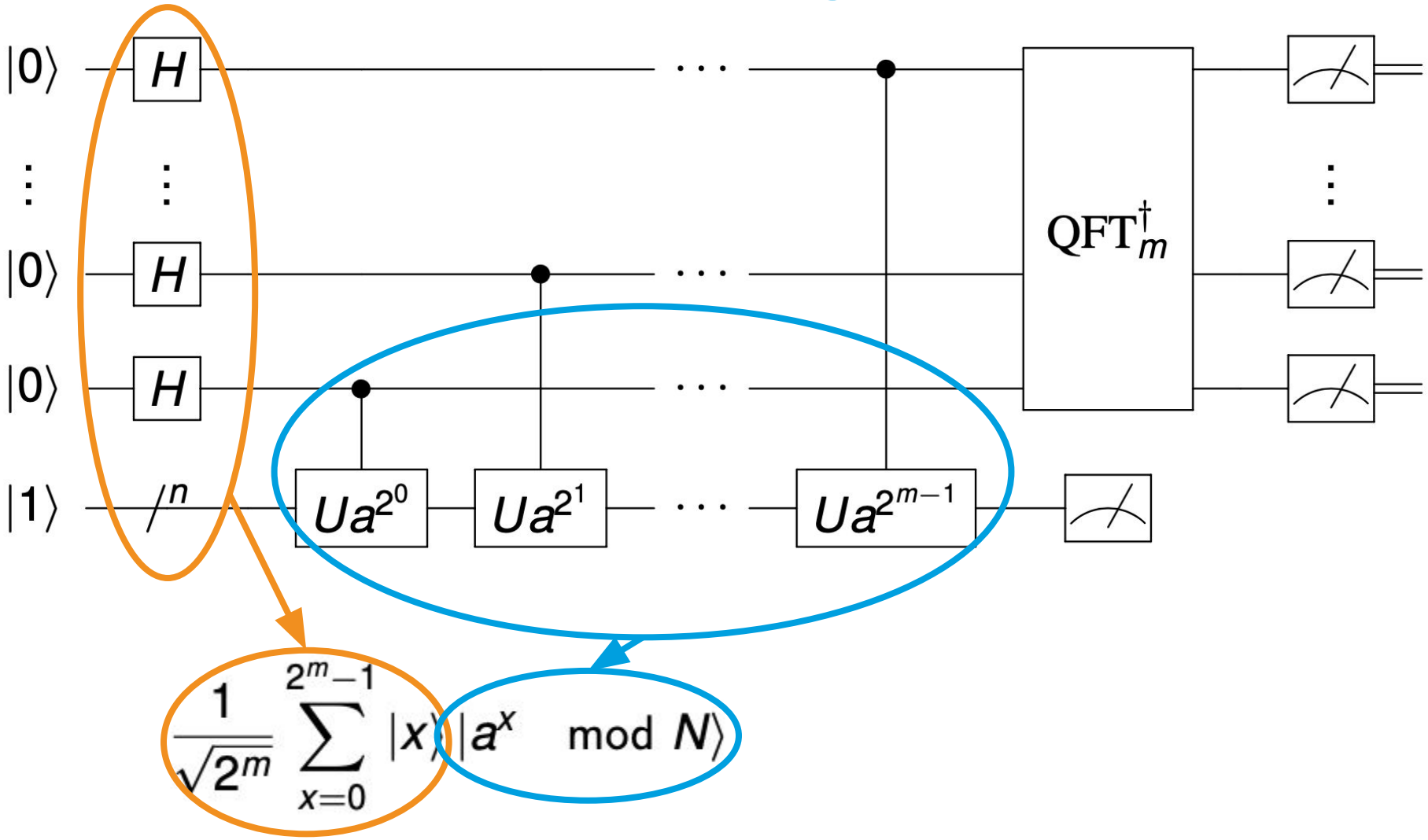
The quantum order-finding routine



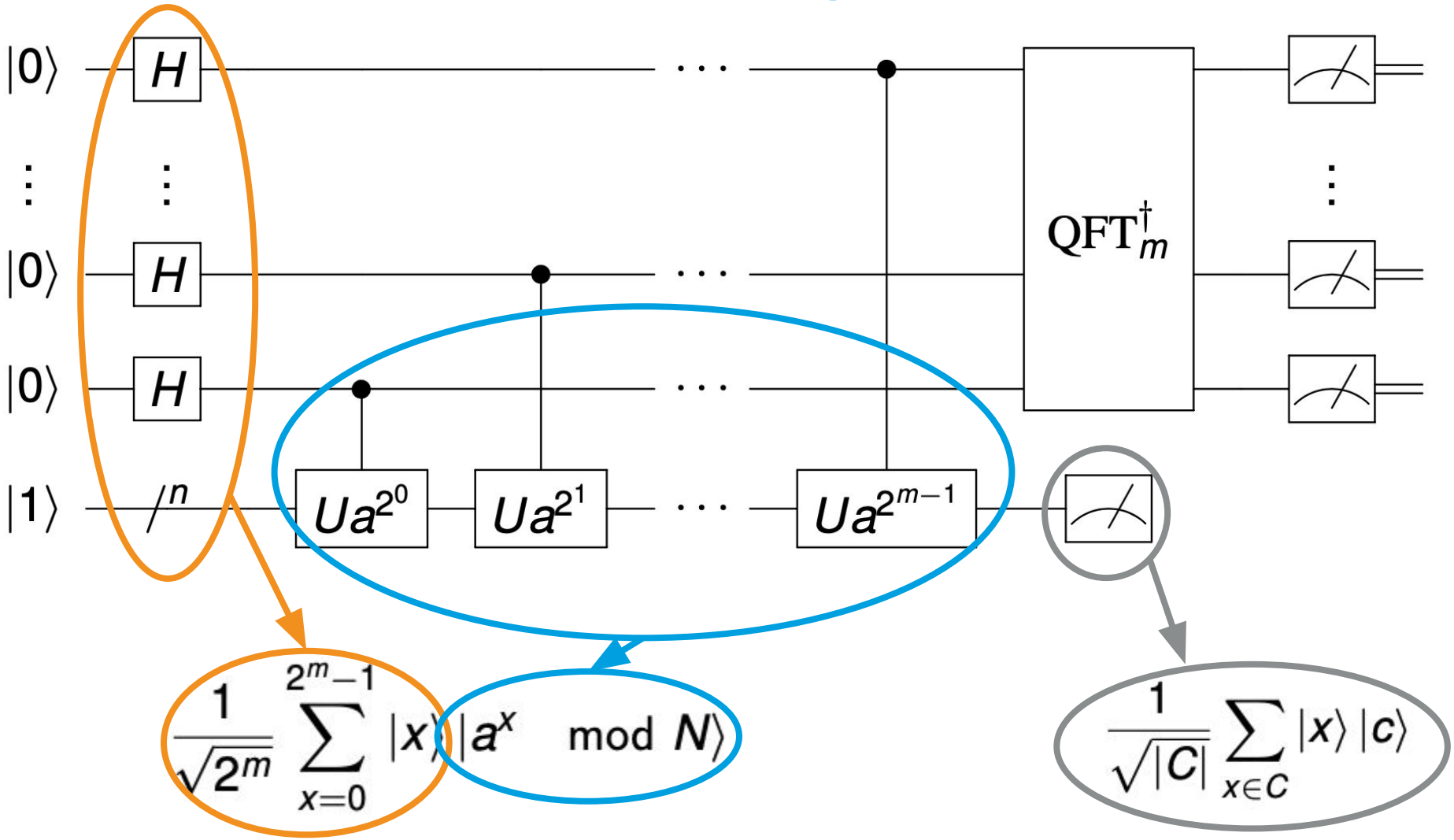
The quantum order-finding routine



The quantum order-finding routine



The quantum order-finding routine



Example

If a = 2, N = 5, m = 4, we would have

$$1/4\ (|0\rangle\ |1\rangle + |1\rangle\ |2\rangle + |2\rangle\ |4\rangle + |3\rangle\ |3\rangle + |4\rangle\ |1\rangle + \ldots + |15\rangle\ |3\rangle)$$

and when we measure we could obtain, for instance

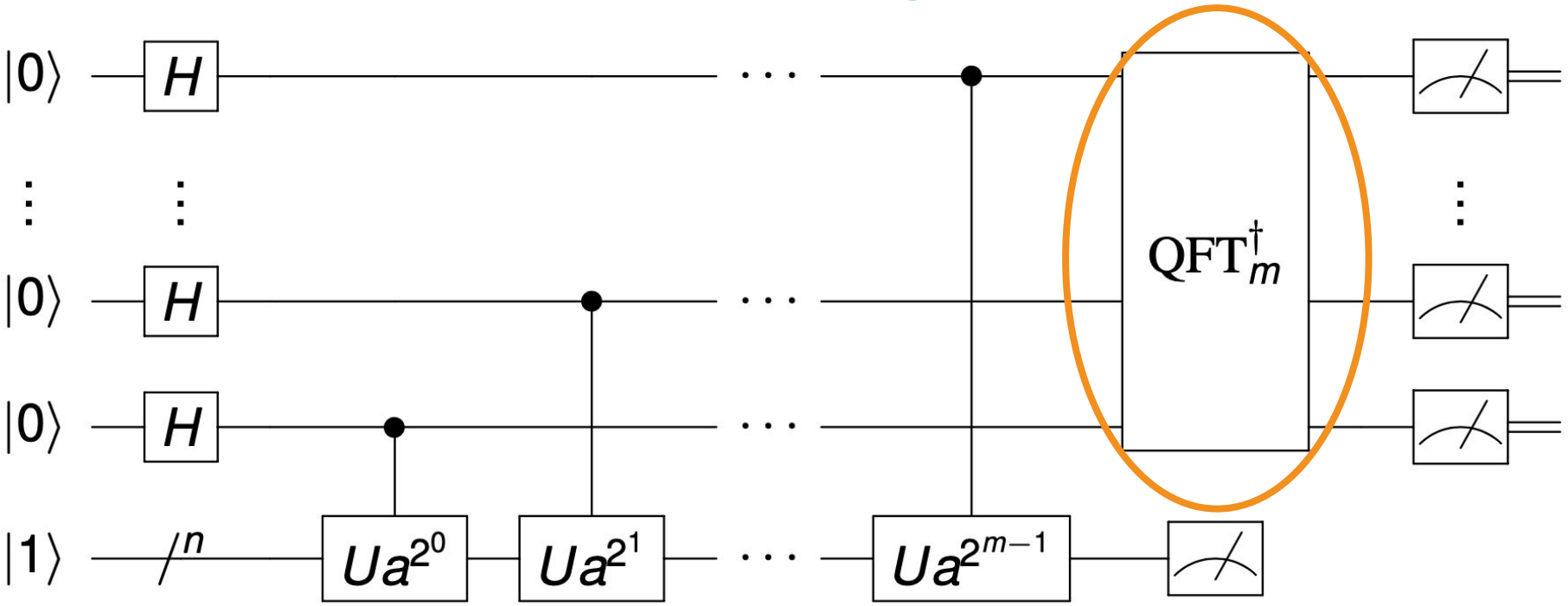
$$1/2\ (|1\rangle\ |2\rangle + |5\rangle\ |2\rangle + |9\rangle\ |2\rangle + |13\rangle\ |2\rangle)$$

Note that the values of the first register are exactly 4 units apart and that

$$2^4 = 1 \bmod 5$$

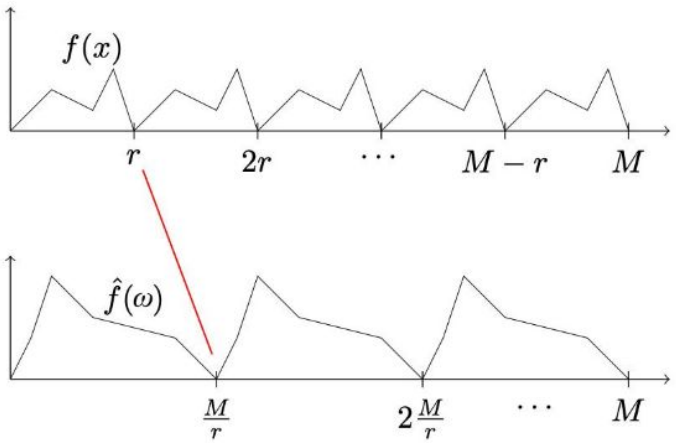
In general, we will obtain values that are r units apart, where $a^r = 1 \bmod N$

The quantum order-finding routine



Properties of the QFT:

- Shift-invariance
- QFT transforms sequences with period r into sequences with period M/r (where $M = 2^m$)



Quantum phase estimation

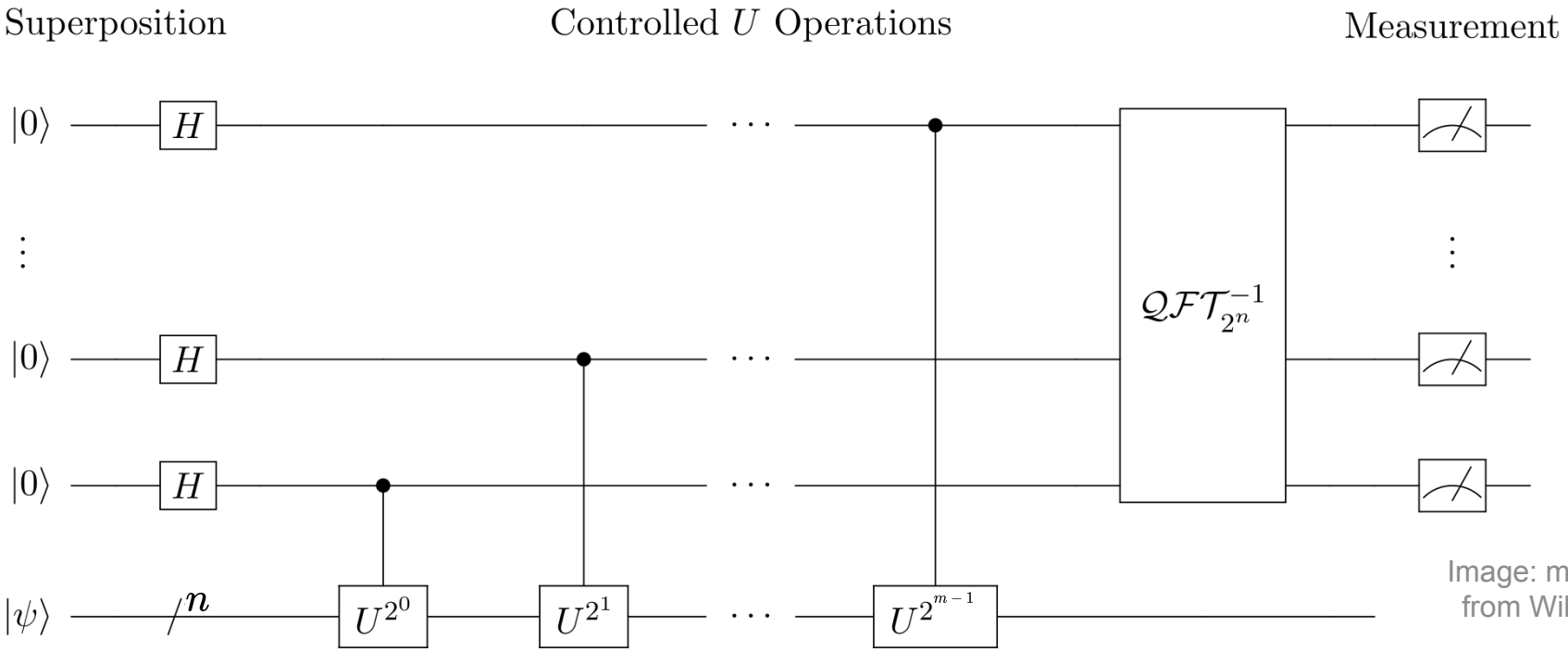


Image: modified from Wikipedia

Suppose we are given a unitary operation U and one of its eigenvectors $|\psi\rangle$

- We know that there exists $\theta \in [0, 1)$ such that $U|\psi\rangle = e^{2\pi i\theta}$
- We can estimate θ with the circuit shown above

The circuit before the QFT will prepare $\frac{1}{\sqrt{2^m}} \sum_{k=0}^{2^m-1} e^{2\pi i\theta k} \left| k \right\rangle$

By using the inverse QFT we can measure $j \approx 2^m\theta$

Shor's algorithm case

The circuit used in Shor's algorithm is a case of quantum phase estimation

- The (unitary) operation of modular multiplication by a has eigenvalues

$$e^{2\pi i \frac{k}{r}} \qquad k = 0, \ldots, r-1$$

  where r is the period of a

- It is not easy to prepare one of the eigenvectors $|\psi_k\rangle$ of the unitary operation
- But we use the fact that $|1\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} |\psi_k\rangle$

- We will get a random integer of the form k/r 2^{2m} for random k = 0, 1, …, r-1
- We can then re-run the quantum subroutine several times to extract r
- **Internet security is now destroyed**

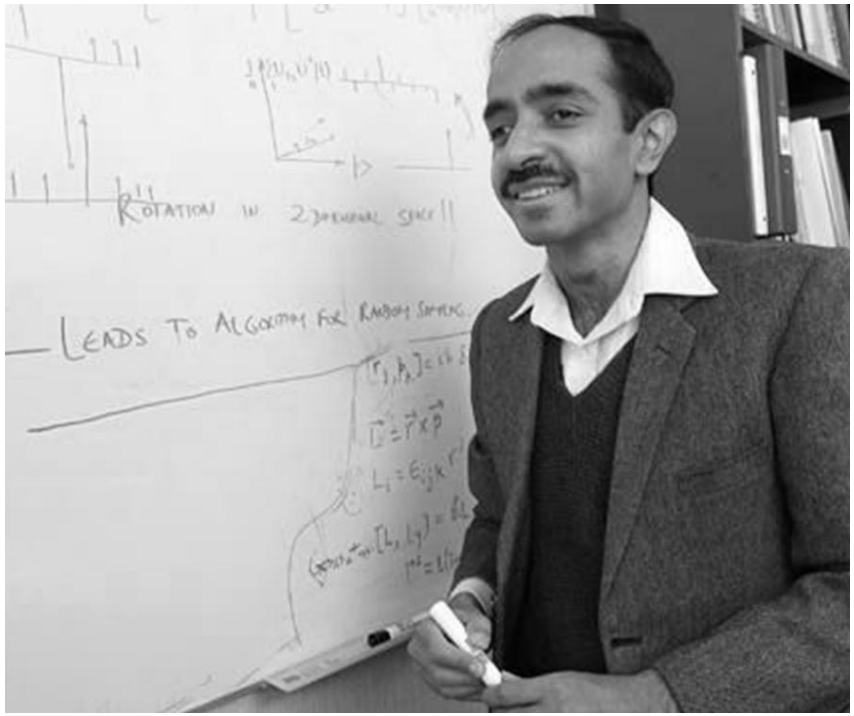
Grover's algorithm

(time-dependent bonus / might skip)

Introduction

Grover's algorithm is used to solve search problems

- Imagine we have an unsorted list of N elements
- One of them verifies a certain condition and we want to find it
- Any classical algorithm requires O(N) queries to the list in the worst case
- Grover's algorithm can find the element with O(√N) queries



The oracle function

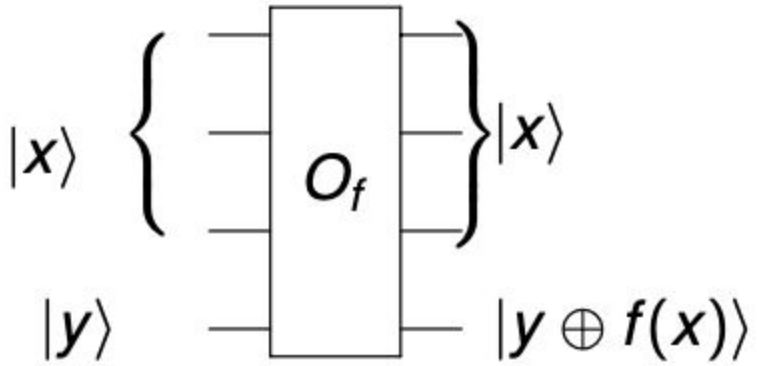
We are given a circuit (an oracle) that implements a one-bit boolean function

An oracle is treated as a black box, a circuit whose interior we cannot know

- This circuit computes, in a reversible way, a certain function f
- For the computation to be reversible, it uses as many inputs as outputs and "writes the result" with an XOR

The oracle computes the function $f : \{0, 1\}^n \Rightarrow \{0, 1\}$ (with $N = 2^n$)

- The element we want to find is the one that verifies f(x) = 1



The strategy

The quantum search algorithm is based on the idea of inversion about the mean

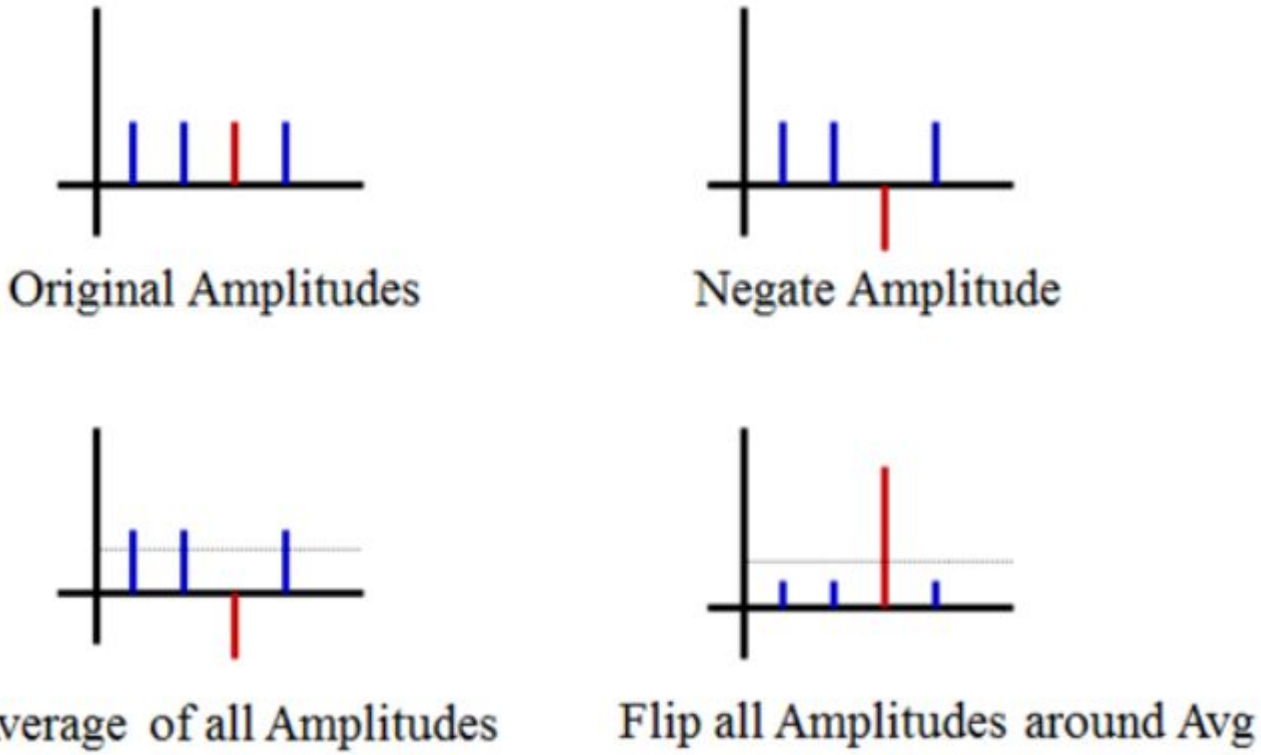


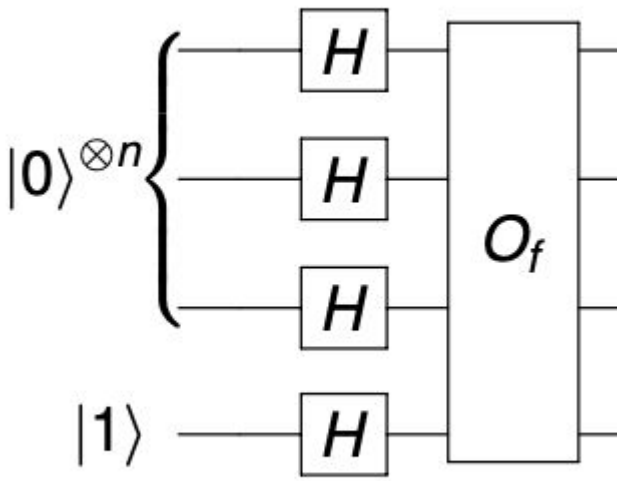
Image credits: quantumcomputing.stackexchange.com

Using the oracle to negate the amplitude

We create the state $| 0 \ldots 0 \rangle | 1 \rangle$

We use Hadamard gates to create the superposition

$$\sum_{x \in \{0,1\}^n} \frac{1}{\sqrt{2^{n+1}}} |x\rangle \left(|0\rangle - |1\rangle\right)$$

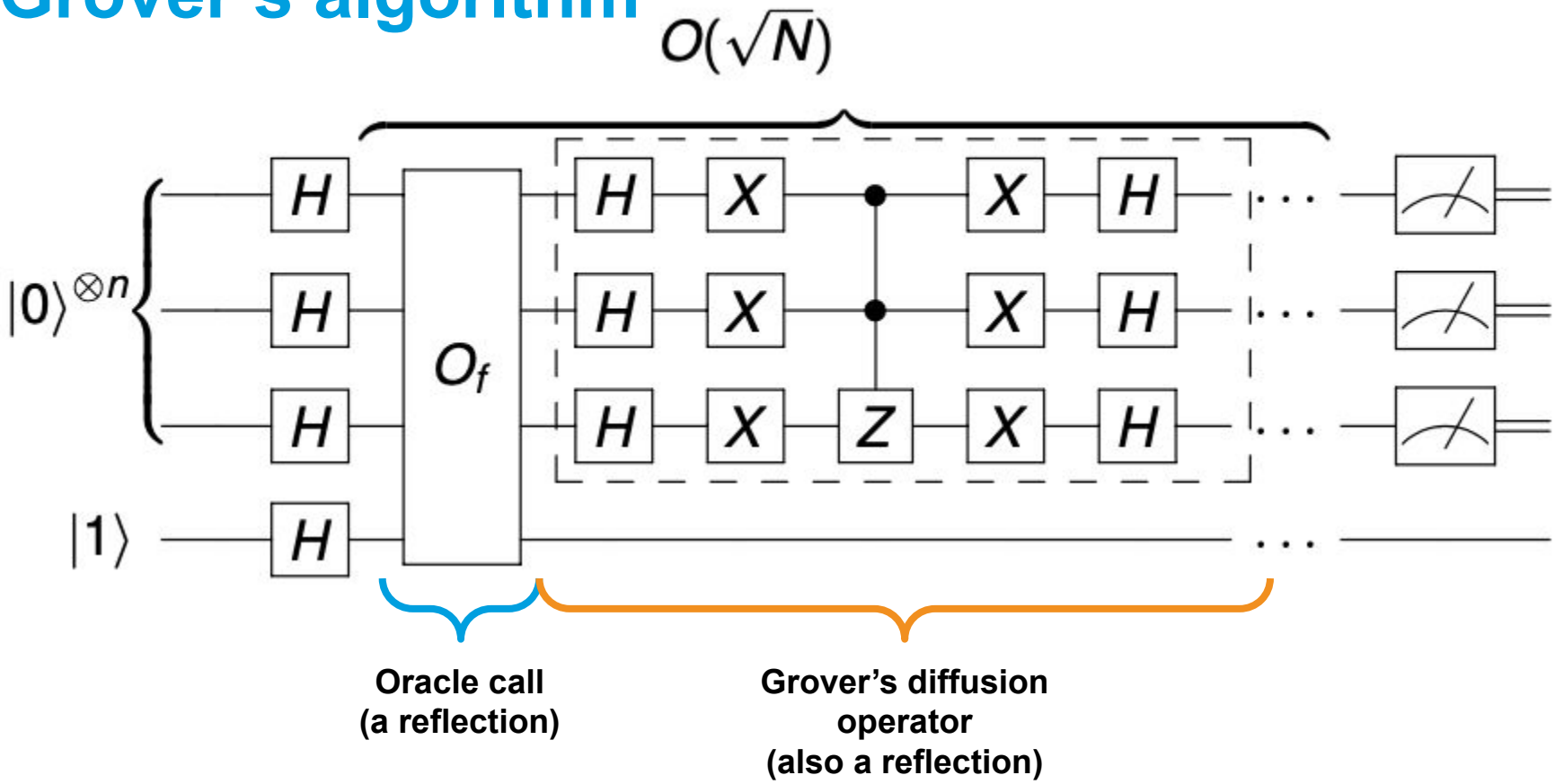


We apply the oracle, getting

$$\sum_{x \in \{0,1\}^n} \frac{1}{\sqrt{2^{n+1}}} |x\rangle \left(|0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle\right) =$$

$$\sum_{x \in \{0,1\}^n} \frac{(-1)^{f(x)}}{\sqrt{2^{n+1}}} |x\rangle \left(|0\rangle - |1\rangle\right)$$

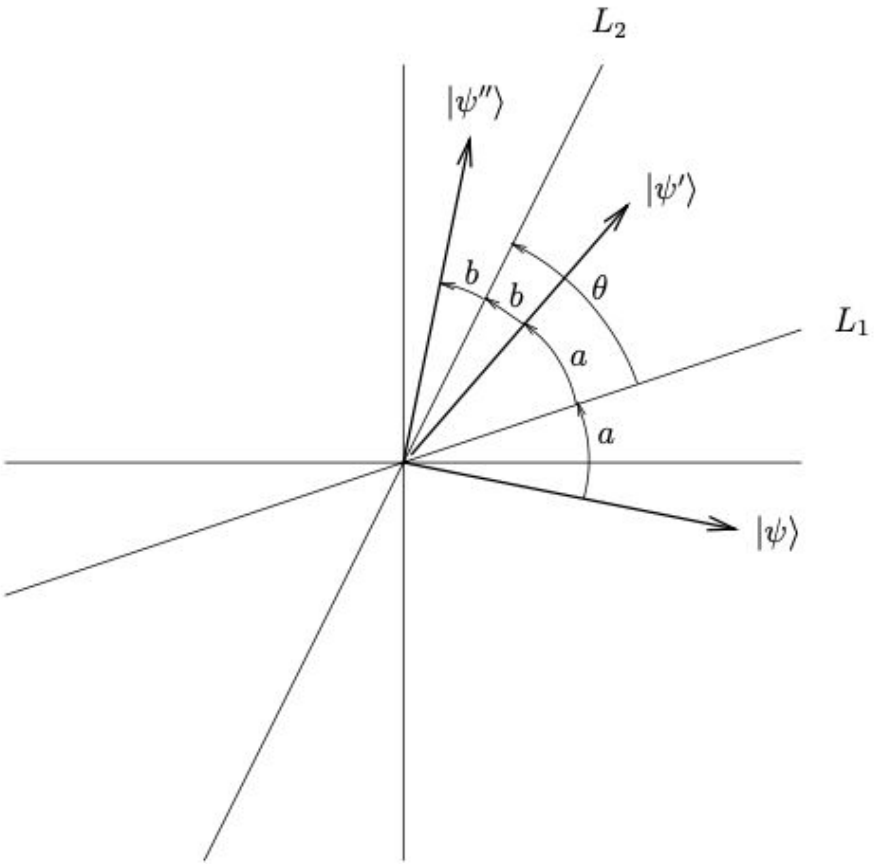
Grover's algorithm



Grover's algorithm performs O(√N) iterations, each one consisting of two steps

- The oracle "marks" those states that verify the condition
- The diffusion operator "amplifies" the amplitudes of the marked states

Reflection x Reflection = Rotation



If we denote by $|x_1\rangle$ the marked element, the initial state of the upper n qubits is

$$\underbrace{\sqrt{\frac{N-1}{N}}}_{\cos\theta}|x_0\rangle + \underbrace{\sqrt{\frac{1}{N}}}_{\sin\theta}|x_1\rangle$$

If D is the diffusion operator and $G = DO_f$

- G acts on the 2-dimensional space spawned by $|x_0\rangle$ and $|x_1\rangle$ as a rotation of angle 2θ

After m iterations:

$$\cos(2m+1)\theta\,|x_0\rangle + \sin(2m+1)\theta\,|x_1\rangle$$

In order to obtain $|x_1\rangle$ with high probability when we measure we need

$$(2m+1)\theta \approx \frac{\pi}{2}$$

The solution

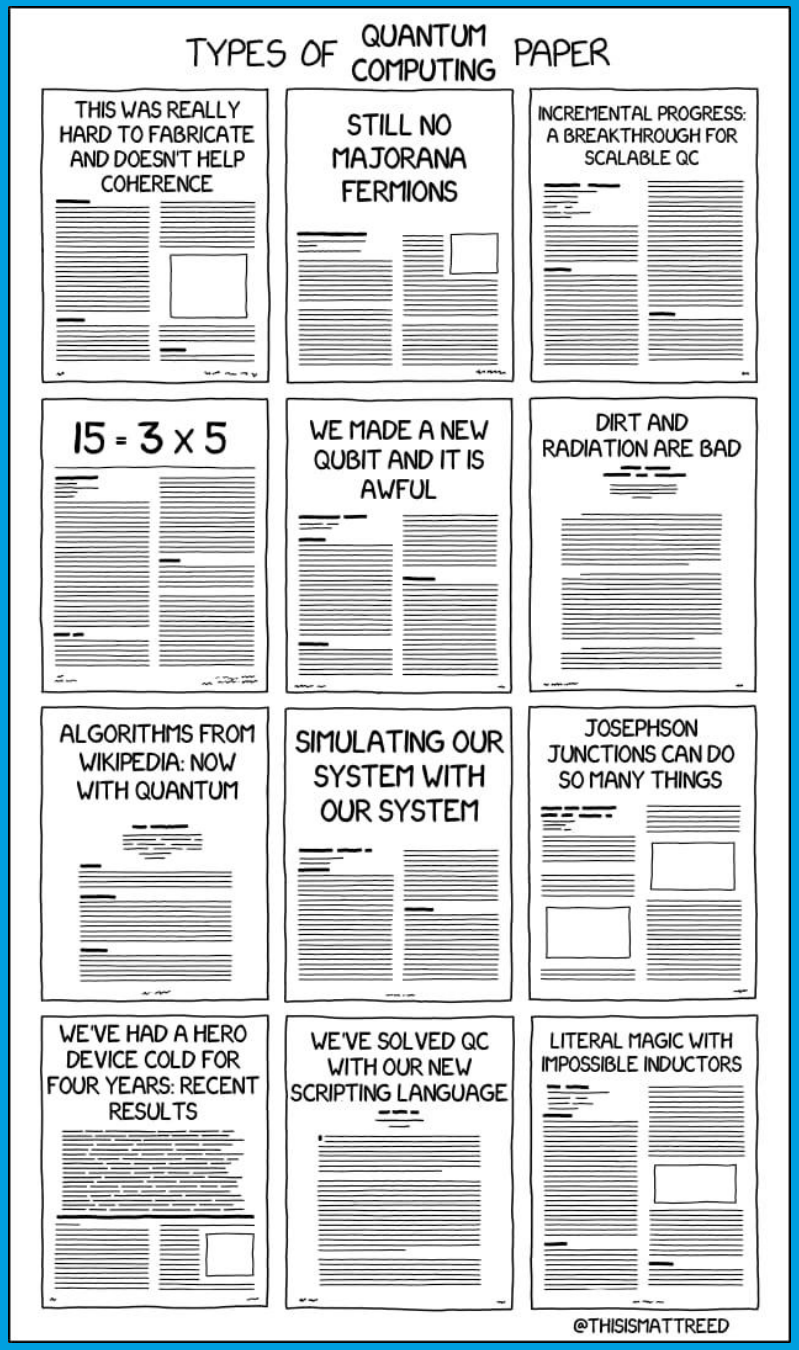
When we measure, we will obtain x such that f(x) = 1 depending on:

- The number m of iterations
- The fraction of values x that satisfy the condition

If we perform too many iterations, we can overshoot and not find a marked element

It can be shown that no other quantum algorithm can obtain more than a quadratic speed-up over over classical algorithms in the same setting

Conclusions



Summary

Discussed the concept of fault-tolerant quantum computing

- Gave one example of error correction

Discussed some of the most famous quantum algorithms

- Quantum Fourier Transform
- Shor’s algorithm
- (maybe) Grover’s algorithm

Hands-on demonstrations:

- Shor's algorithm
- Grover's algorithm

Thank you!