Introduction to quantum computing

Lezione 1

We start from bits

- (but things will get more complex quickly)
- A bit is anything which can be in one of two states
- We denote these two states as 0 and 1
- How many possibilities if we have 3 bits? And if we have n bits?

Representing bit-strings as vectors

One bit: 0 or 1

We will represent them as vectors since this easily generalizes to qubits

$$|0\rangle = \begin{bmatrix}1\\0\end{bmatrix} \qquad |1\rangle = \begin{bmatrix}0\\1\end{bmatrix}$$

With 2 bits we have 4 possibilities

$$|00\rangle = \begin{bmatrix}1\\0\\0\\0\end{bmatrix} \quad |01\rangle = \begin{bmatrix}0\\1\\0\\0\end{bmatrix} \quad |10\rangle = \begin{bmatrix}0\\0\\1\\0\end{bmatrix} \quad |11\rangle = \begin{bmatrix}0\\0\\0\\1\end{bmatrix}$$

Tensor products

All the possible combinations, with the product of values

$$\begin{bmatrix}x_0\\x_1\end{bmatrix}\otimes\begin{bmatrix}y_0\\y_1\end{bmatrix}=\begin{bmatrix}x_0y_0\\x_0y_1\\x_1y_0\\x_1y_1\end{bmatrix} \qquad |00\rangle=|0\rangle\otimes|0\rangle=\begin{bmatrix}1\\0\end{bmatrix}\otimes\begin{bmatrix}1\\0\end{bmatrix}=\begin{bmatrix}1\\0\\0\\0\end{bmatrix}$$

For now, values can be just 0 and 1

Observation: exactly one 1 in each vector on the LHS implies exactly one 1 in the result

The operator can be generalized to n vectors

Tensor product corresponds to compose states

Manipulating bits

One bit operation: NOT

NOT(x) =

- 1 if x=0
- 0 if x=1

NOT just flips the bit

Notation for NOT(x): $\bar{x}$

Manipulating bits as vectors

x can be either or

$$\begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}$$

More in general

$$\begin{bmatrix}x_0\\x_1\end{bmatrix}$$

We use matrices to represent operators, and matrix multiplication to apply them

NOT is denoted as X, and represented by the matrix

$$X = \begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}$$

Matrix multiplication

We can multiply a matrix nxm and a matrix mxp and obtain a matrix nxp

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,m} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,m} \end{bmatrix} \qquad B = \begin{bmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,p} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m,1} & b_{m,2} & \cdots & b_{m,p} \end{bmatrix}$$

$$C = \begin{bmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,p} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n,1} & c_{n,2} & \cdots & c_{n,p} \end{bmatrix}$$

$$c_{i,j} = a_{i,1}b_{1,j} + a_{i,2}b_{2,j} + \cdots + a_{i,m}b_{m,j} = \sum_{k=1}^{m} a_{i,k}b_{k,j}$$

Element obtained by multiplying row by column

Flipping a bit

Gli operatori per i qbits sono delle funzioni bi-ettive, matrici identitarie con proprietà tra cui poter tornare indietro. swap fili, si invertono,

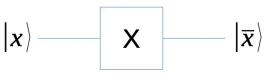
ogni operatore è una matrice:

- per vettore $|0\rangle$ il **not** è il seguente:

$$\begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}$$

è reversibile è ci piace, poi abbiamo anche:

- Circuit diagram:



Ket notation:

$$|\bar{x}\rangle = X|x\rangle$$

Manipulating one bit

If someone hands you a bit, there are two ways you can process it to obtain another bit:

1. Keep it as it is (identity)
2. Flip it (NOT)

That's it. If we want more interesting computations, we'd better have more bits

We could also turn it into constant 0 or 1, but these are not reversible

> 💡 Why is so important the factor reversible? Because for the entropy if we not use something it became heat that is very bad for the efficiency.
>
> So every solution reversible will be more efficient then any other solution not reversible

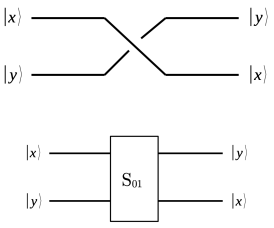
Swapping two bits

Given two bits, what can we do? For instance, swap them!

- lo swap è

$$S_{01=}\begin{bmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1\end{bmatrix}$$

sostanzialmente abbiamo al prima e l'ultima colon invariate e invertiamo le due centrali



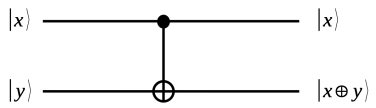
$$S_{01}|x\rangle|y\rangle = |y\rangle|x\rangle$$

- la and non è invertibile, quindi devo aggiungere un bit, che è uno spreco perche con da 2 bit, per disambiguare con un 3 bit posso piu di 4 bit e noi ne dobbiamo scegliere tra tre.

Controlled NOT

controlled not CNOT:

$$CNOT|x\rangle|y\rangle=|x\rangle|x\oplus y\rangle$$
$$where\ x\oplus y=(x+y)\ mod\ 2$$



praticamente lo XOR(non è reversibile e non è biettiva) allora mi salvo la x, cosi da poter rendere la CNOT invertibile.

$$CNOT|x\rangle|y\rangle=|x\rangle|x\oplus y\rangle$$

$$\begin{bmatrix}00\\01\\10\\11\end{bmatrix}\qquad CNOT\ swaps\ 10\ and\ 11$$

$$00\to 00, 01\to 10, 10\to 11, 11\to 10$$

The matrix doing this is $\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}$

Quello che facciamo è avere le prime due righe uguali e le ultime due ivertite

Other operetor

Da due bit ad uno, le operazioni classiche che non fanno un 50/50 split dei propri casi non sono invertibili neanche introducendo un bit di supporto

OR e AND sucano

Gate	Notation	Matrix
NOT (Pauli-X)		$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Z		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard		$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
CNOT (Controlled NOT)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$

Lezione 2

Reversible classical circuits

We have seen some classical operations on bits

- NOT, swap, CNOT

You can create classical circuits composing them

Actually, classical reversible circuits

- Circuits where you can as well compute input from output
- The circuits above are self-inverse: if you compose two copies of them you get the identity

From classical to quantum

Now you could study classical (reversible) computation

Figure out which gates you need to do stuff, how many of them

We will not do this

We will move from classical computation to quantum computation

Our model will be based on qubits, not on bits

Qubits extend bits, and allow for quantum effects

What is a qubit?

A quantum system whose state is a 2-dimensional complex unit vector, namely:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \;\; where \; |\alpha|^2 + |\beta|^2 = 1$$

on complex S1
3 DOF: thetas and 1 rho

2-dimensional vector: linear combination of 2 linearly independent elements (base)

Complex: α and β are complex numbers

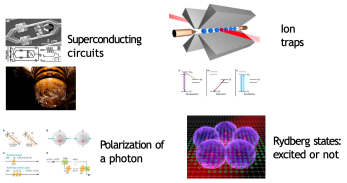
Unit: the length of the vector is 1, as captured by the side condition, where if α=a+ib then $|\alpha| = \sqrt{a^2 + b^2}$

Bits as qubits

A classical bit is just a qubit, which also satisfiesα = 1 or β = 1.

Actually, all bits are really qubits.

But for most systems we are used to, natural physical processes drive an arbitrary state$|\psi\rangle$ to either $|1\rangle$ or $|0\rangle$ really fast. So it's hard to see the quantumness.



A real qubit is quite complex

It would take a whole other course (with WAY more prereqisites) to understand “which systems make good qubits?”

For us, it’s enough to know: people are getting pretty good at making systems with 100’s of qubits right now. Plausible paths to 1,000,000’s within 10’s of years.

Our goal is to understand the computational model that quantum theory implies, and better understand what quantum computers can do.

Levels of abstraction

As usual in computer science, when something is too complex, we use abstraction to hide complexity

In real life: I can use TV without knowing how it works. I just need to know what to do with the TV controller

In programming: I can invoke a function (or a web service) without knowing its implementation, I just need to know its interface and specification

Our level of abstraction

We are in the business of using qubits, not building them or fixing them.

For us, a quantum system is just something whose state is a complex unit vector

Qubits can be the polarization of a photon, two hyperfine states of an atom, etc., but you do not care

This avoids the need for a lot of physics, and will work also on future implementations

Axioms of Quantum Theory

Axiom 1 (states)

The state of a quantum system is a complex unit vector $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \;\; where \; |\alpha|^2 + |\beta|^2 = 1$

- 2-dimensional vector: qubit
- d-dimensional vector: qudit

α and β are called amplitudes

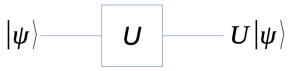
Simple state

$$|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \quad |-\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$$

Sort of like “50% 0, 50% 1”, but also different (we’ll see more later in the measurement axiom)
These are examples of states in a superposition (states not in a superposition are called basis states)

Axiom 2 (dynamics)

The evolution of a closed system is described by a unitary matrix U



Unitary means that U† U = I, where U† is the conjugate transpose

- Transpose: swap rows and columns
- Conjugate: for each element, change the sign of the complex part
  - E.g., an element 3+2i goes to 3-2i

A unitary matrix maps unitary vectors to unitary vectors

Four fantastic unitaries

The identity and the 3 Pauli matrices (rotations in Bloch sphere)

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \;\; X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \;\; Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \;\; Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Check, are these unitaries?

$$Y^\dagger = (Y^T)^* = (\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}^T)^* = (\begin{bmatrix} 0 & i \\ -i & 0 \end{bmatrix})^* = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$

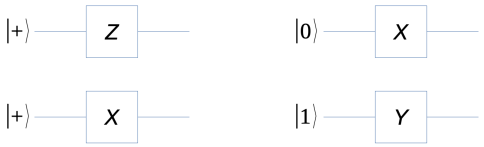
$$Y^\dagger Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 \cdot 0 + (-i) \cdot i & 0 \cdot (-i) + (-i) \cdot 0 \\ i \cdot 0 + 0 \cdot i & i \cdot (-i) + 0 \cdot 0 \end{bmatrix} = \begin{bmatrix} -i^2 & 0+0 \\ 0+0 & -i^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

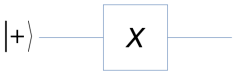
Si è una matrice identitaria

Exercise

Which circuit prepares $|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$?



Solution



Since X is NOT, and is linear:

$$X|+\rangle = X\frac{1}{\sqrt{2}}|0\rangle + X\frac{1}{\sqrt{2}}|1\rangle = \frac{1}{\sqrt{2}}|1\rangle + \frac{1}{\sqrt{2}}|0\rangle = |+\rangle$$

Using matrices:

$$X|+\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}}|0\rangle \\ \frac{1}{\sqrt{2}}|1\rangle \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}}|1\rangle \\ \frac{1}{\sqrt{2}}|0\rangle \end{bmatrix} = |+\rangle$$

Hadamard

$$H = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \quad H|0\rangle = |+\rangle \; and \; H|1\rangle = |-\rangle$$

Since $H^\dagger H = I$ and $H^\dagger = H$ we also have $H|+\rangle = |0\rangle$ and $H|-\rangle = |1\rangle$

Very useful, we can create and destroy superpositions

Axiom 3 (measurements)

We have a theory of vectors that you can rotate. The state is a collection of complex numbers (so an infinite number of bits).

The next axiom tells us that the information we can extract about the state of a system is very limited.

We can measure a system in any basis for its state space. if you measure:

$$|\psi\rangle = \alpha_0|0\rangle + \beta_1|1\rangle$$

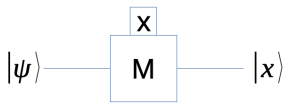
in the basis $\{|0\rangle, |1\rangle\}$ (computational basis) the result is probabilistic.

You get an outcome x with probability $|\alpha_x|^2$

Furthermore, the state of the system collapses to $|x\rangle$

This is called the Born rule

Measurement: circuit diagram



Book notation

More common notation

Bell State measurement $|00\rangle \rightarrow |\Phi^+\rangle$

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = \frac{1}{\sqrt{2}}(\begin{bmatrix}1\\0\\0\\0\end{bmatrix} + \begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}) = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\0\\0\\1\end{bmatrix}$$

Density matrix:

$$\frac{1}{\sqrt{2}}\begin{bmatrix}1\\0\\0\\1\end{bmatrix}\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 0 & 0 & 1\end{bmatrix} = \frac{1}{2}\begin{bmatrix}1 & 0 & 0 & 1\\0 & 0 & 0 & 0\\0 & 0 & 0 & 0\\1 & 0 & 0 & 1\end{bmatrix}$$

expected value

$$\langle 00 \mid \rho_{\Phi+} \mid 00\rangle = \begin{bmatrix}1\\0\\0\\0\end{bmatrix} \cdot \frac{1}{2}\begin{bmatrix}1 & 0 & 0 & 1\\0 & 0 & 0 & 0\\0 & 0 & 0 & 0\\1 & 0 & 0 & 1\end{bmatrix} \cdot \begin{bmatrix}1 & 0 & 0 & 0\end{bmatrix} = \frac{1}{2}$$

Bell state measurement $|0+\rangle \rightarrow |\Phi^-\rangle$

$$|\Phi^-\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) = \frac{1}{\sqrt{2}}(\begin{bmatrix}1\\0\\0\\0\end{bmatrix} - \begin{bmatrix}0\\0\\0\\-1\end{bmatrix}) = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\0\\0\\-1\end{bmatrix}$$

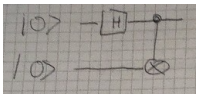
Density matrix:

$$\rho = |\Phi^-\rangle\langle\Phi^-| = \frac{1}{2}\begin{bmatrix}1 & 0 & 0 & -1\\0 & 0 & 0 & 0\\0 & 0 & 0 & 0\\-1 & 0 & 0 & 1\end{bmatrix}$$

expected value

$$\langle\psi \mid M \mid \psi\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\0\\0\\-1\end{bmatrix} \cdot \frac{1}{2}\begin{bmatrix}1 & 0 & 0 & -1\\0 & 0 & 0 & 0\\0 & 0 & 0 & 0\\-1 & 0 & 0 & 1\end{bmatrix} \cdot \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 0 & 0 & -1\end{bmatrix} = \frac{1}{4}$$

An other example of circuits:



$$|\psi\rangle = \frac{1}{\sqrt{2}}(|10\rangle + |11\rangle)$$

Sampling e measurement:

$|\psi\rangle = \frac{1}{\sqrt{2}}\big(|10\rangle + |11\rangle\big)$ *Campionamento classico restituisce* $(10, 11, 10, \dots)$.

Expected value — esempio di calcolo

$\langle\psi \mid 100\rangle = \frac{1}{\sqrt{2}} \implies \big|\langle\psi \mid 100\rangle\big|^2 = \frac{1}{2}$

$$Operatore(ZZ = Z \otimes Z), dove\ Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \implies ZZ = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Dato\ lo\ stato: \quad |\psi\rangle = \frac{1}{\sqrt{2}}\big(|10\rangle + |11\rangle\big)$$
$$[ZZ\,|\psi\rangle = \frac{1}{\sqrt{2}}\big(ZZ\,|10\rangle + ZZ\,|11\rangle\big) = \frac{1}{\sqrt{2}}\big(-|10\rangle + |11\rangle\big)]$$
$$[\langle\psi \mid ZZ \mid \psi\rangle = \langle\psi \mid \big(-\,|10\rangle + |11\rangle\big)\rangle = -\tfrac{1}{2} + \tfrac{1}{2} = 0]$$

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}\big(|00\rangle + |11\rangle\big)$$
$$Applichiamo(ZZ): ZZ\,|\Phi^+\rangle = \frac{1}{\sqrt{2}}\big(ZZ\,|00\rangle + ZZ\,|11\rangle\big) = |\Phi^+\rangle$$
$$\langle\Phi^+ \mid ZZ \mid \Phi^+\rangle = 1$$

Axiom 4 (composite systems)

If

- A has a state in span(V) for some set of vectors V
- B has a state in span(W) for some set of vectors W
- AB has a state in span({v w | v in V, w in W}

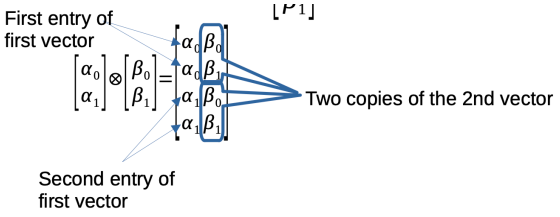
*Span(V) denotes the set of all finite linear combinations of the elements of V, is the tensor product

Lezione 4

Tensor products

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$$
$$|\phi\rangle = \beta_0|0\rangle + \beta_1|1\rangle$$

$$|\psi\rangle \otimes |\phi\rangle =$$
$$= \alpha_0\beta_0|0\rangle\otimes|0\rangle + \alpha_0\beta_1|0\rangle\otimes|1\rangle + \alpha_1\beta_0|1\rangle\otimes|0\rangle + \alpha_1\beta_1|1\rangle\otimes|1\rangle =$$
$$= \alpha_0\beta_0|00\rangle + \alpha_0\beta_1|01\rangle + \alpha_1\beta_0|10\rangle + \alpha_1\beta_1|11\rangle =$$
$$= \begin{bmatrix} \alpha_0\beta_0 \\ \alpha_0\beta_1 \\ \alpha_1\beta_0 \\ \alpha_1\beta_1 \end{bmatrix}$$



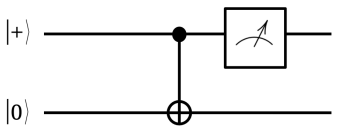
Compute:

$$|\psi\rangle = |+\rangle \otimes |0\rangle$$

sappiamo che $|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$

$$|\psi\rangle = |+\rangle \otimes |0\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}}1 \\ \frac{1}{\sqrt{2}}0 \\ \frac{1}{\sqrt{2}}1 \\ \frac{1}{\sqrt{2}}0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \\ 0 \end{bmatrix}$$

What if I have multiple qubits, and I measure only one of them?



Phi è uno stack verticale di phi zero e phi uno, pesati ciascuno per il proprio scalare

Definizione ricorsiva che ad ogni layer aggiunge tre DOF

Let's refine axiom 3 to cover this case as well, any state of n qubits can be written as:

$$|\psi\rangle = \alpha_0|0\rangle \otimes |\psi_0\rangle + \alpha_1|1\rangle \otimes |\psi_1\rangle$$

Where $|\psi_i\rangle$ are n-1 qubit states and $|\alpha_0|^2 + |\alpha_1|^2 = 1$

Axiom 3′ (partial measurement)

We can measure the first qubit of the n qubit state $|\psi\rangle = \alpha_0|0\rangle \otimes |\psi_0\rangle + \alpha_1|1\rangle \otimes |\psi_1\rangle$ in the basis $\{|0\rangle, |1\rangle\}$.

The outcome is x with probability $|\alpha_x|^2$

Furthermore, the state of the system collapses to $|x\rangle \otimes |\psi_x\rangle$

(**Generalized**) Born rule

So what happens here?

1. We can compute the initial composite state
2. We can compute the result after the CNOT
3. We can compute the effect of the partial measurement (difficult using vectors, much better with kets)

Arbitrary one qubit states

We want to prepare an arbitrary qubit state $|\psi\rangle$ starting from $|0\rangle$ (or $|1\rangle$)

That is, we want a unitary U such that $|\psi\rangle = U|0\rangle$

We can always find such a unitary

Finding U

Take $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$

From $|\psi\rangle = U|0\rangle$ we get $U = \begin{bmatrix} \alpha & x \\ \beta & y \end{bmatrix}$ for some x,y

U needs to be unitary, we show that $U = \begin{bmatrix} \alpha & -\beta^* \\ \beta & \alpha^* \end{bmatrix}$ is unitary hence it is a solution

$$UU^\dagger = \begin{bmatrix} \alpha & -\beta^* \\ \beta & \alpha^* \end{bmatrix} \begin{bmatrix} \alpha^* & \beta^* \\ -\beta & \alpha \end{bmatrix} = \begin{bmatrix} \alpha\alpha^* + -\beta^* - \beta & \alpha\beta^* - \alpha\beta^* \\ \beta\alpha^* - \beta\alpha^* & \beta\beta^* + \alpha^*\alpha \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

How to get the $|0\rangle$

We have learned how to get any 1 qubit state from $|0\rangle$

We will show how to get also two qubits states

But how to get the $|0\rangle$ to start with?

In some cases, it can be obtained using physical processes

– E.g., minimal energy state, by cooling down

Otherwise, take an unknown state, measure it so to get either $|0\rangle$ or $|1\rangle$, and negate it if it is $|1\rangle$

Arbitrary two qubit states

Now we want to prepare an arbitrary two qubit state $|\psi\rangle$ starting from $|0\rangle$ s

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$$

One qubit gates are not enough

$$U|0\rangle \otimes V|0\rangle = (\alpha_0|0\rangle + \alpha_1|1\rangle) \otimes (\beta_0|0\rangle + \beta_1|1\rangle)$$

To have $U|0\rangle \otimes V|0\rangle = |\psi\rangle$ we need $\frac{\alpha_{00}}{\alpha_{01}} = \frac{\alpha_{10}}{\alpha_{11}} = \frac{\beta_0}{\beta_1}$ what is not true in general

Adding a single CNOT in not enught

We want to prepare:
$|\sigma\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$
First, note that
$$|\sigma\rangle = |0\rangle \otimes |\psi\rangle + |1\rangle \otimes |\phi\rangle$$
with $|\psi\rangle = \alpha_{00}|0\rangle + \alpha_{01}|1\rangle$ and $|\phi\rangle = \alpha_{10}|0\rangle + \alpha_{11}|1\rangle$

We want to find U such that
$$U \otimes I|\sigma\rangle = |0\rangle \otimes |\psi'\rangle + |1\rangle \otimes |\phi'\rangle$$
with $|\psi'\rangle$ and $|\phi'\rangle$ orthogonal
(if $|\psi\rangle$ and $|\phi\rangle$ are orthogonal we can choose $U = I$)

Finding U

We want to find U such that
$U \otimes I|\sigma\rangle = |0\rangle \otimes |\psi'\rangle + |1\rangle \otimes |\phi'\rangle$
Let us try with
$U = \begin{bmatrix} a & -b^* \\ b & a^* \end{bmatrix}$ which we know is unitary

$$U \otimes I|\sigma\rangle = (a|0\rangle + b|1\rangle) \otimes |\psi\rangle + (-b^*|0\rangle + a^*|1\rangle) \otimes |\phi\rangle =$$
$$= |0\rangle \otimes |\psi'\rangle + |1\rangle \otimes |\phi'\rangle$$
where $|\psi'\rangle = a|\psi\rangle - b^*|\phi\rangle$ and $|\phi'\rangle = b|\psi\rangle + a^*|\phi\rangle$

Orthogonal vectors

Two vectors $|\psi\rangle, |\phi\rangle$ are orthogonal iff $\langle\psi|\phi\rangle = 0$
This is the inner product
Namely $\sum_{i=1}^{n} \psi_i^* \phi_i$
Note that $\langle\psi|\phi\rangle = \langle\phi|\psi\rangle^*$
It is linear in the right argument, antilinear in the left one
Inner product of a unitary vector for itself is 1

Computing the coefficients

We want
$|\phi'\rangle = b|\psi\rangle + a^*|\phi\rangle$ and $|\psi'\rangle = a|\psi\rangle - b^*|\phi\rangle$
orthogonal

$$0 = \langle\phi'|\psi'\rangle = b^* a\langle\psi|\psi\rangle + aa\langle\phi|\psi\rangle - b^* b^*\langle\psi|\phi\rangle - ab^*\langle\phi|\phi\rangle =$$
$$= a^2\langle\phi|\psi\rangle - b^{*2}\langle\psi|\phi\rangle + ab^*(\langle\psi|\psi\rangle - \langle\phi|\phi\rangle) =$$
$$= a^2\langle\phi|\psi\rangle - b^{*2}\langle\psi|\phi\rangle$$

We can solve the equation to derive values for a and b

Normalization

We have
$U \otimes I|\sigma\rangle = |0\rangle \otimes |\psi'\rangle + |1\rangle \otimes |\phi'\rangle$
with $|\psi'\rangle$ and $|\phi'\rangle$ orthogonal

They may not be normalized yet, we need to divide them for their module, hence we define

$$|\psi''\rangle = \frac{|\psi'\rangle}{\lambda}, |\phi''\rangle = \frac{|\phi'\rangle}{\mu}$$

They are still orthogonal and now unitary

Tensor product of matrices

- We want
  $(U \otimes V)(|\phi\rangle \otimes |\psi\rangle) = U|\phi\rangle \otimes V|\psi\rangle$
- We can actually define it as follows:

$$U = \begin{bmatrix} u_{1,1} & u_{1,2} \\ u_{2,1} & u_{2,2} \end{bmatrix} \quad V = \begin{bmatrix} v_{1,1} & v_{1,2} \\ v_{2,1} & v_{2,2} \end{bmatrix}$$

$$U \otimes V = \begin{bmatrix} u_{1,1}\begin{bmatrix} v_{1,1} & v_{1,2} \\ v_{2,1} & v_{2,2} \end{bmatrix} & u_{1,2}\begin{bmatrix} v_{1,1} & v_{1,2} \\ v_{2,1} & v_{2,2} \end{bmatrix} \\ u_{2,1}\begin{bmatrix} v_{1,1} & v_{1,2} \\ v_{2,1} & v_{2,2} \end{bmatrix} & u_{2,2}\begin{bmatrix} v_{1,1} & v_{1,2} \\ v_{2,1} & v_{2,2} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} u_{1,1}v_{1,1} & u_{1,1}v_{1,2} & u_{1,2}v_{1,1} & u_{1,2}v_{1,2} \\ u_{1,1}v_{2,1} & u_{1,1}v_{2,2} & u_{1,2}v_{2,1} & u_{1,2}v_{2,2} \\ u_{2,1}v_{1,1} & u_{2,1}v_{1,2} & u_{2,2}v_{1,1} & u_{2,2}v_{1,2} \\ u_{2,1}v_{2,1} & u_{2,1}v_{2,2} & u_{2,2}v_{2,1} & u_{2,2}v_{2,2} \end{bmatrix}$$

Building the circuit

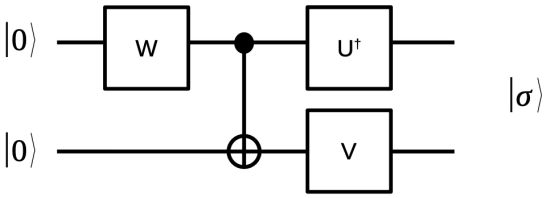
Take V (unitary) such that
$|\psi''\rangle = V|0\rangle, |\phi''\rangle = V|1\rangle$

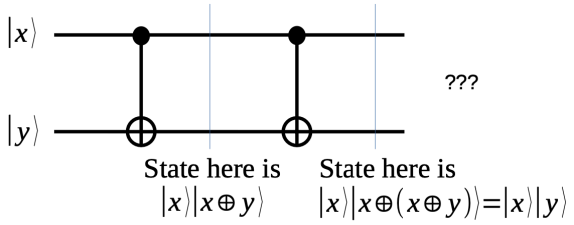
We have

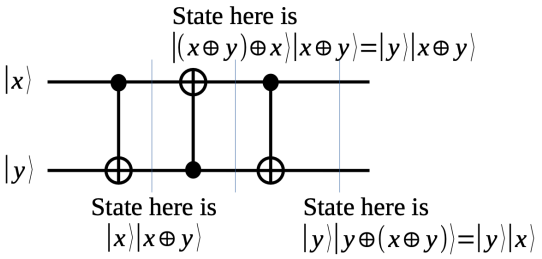
$$U \otimes I|\sigma\rangle = |0\rangle \otimes |\psi'\rangle + |1\rangle \otimes |\phi'\rangle$$
$$U \otimes I|\sigma\rangle = |0\rangle \otimes \lambda|\psi''\rangle + |1\rangle \otimes \mu|\phi''\rangle$$
$$U \otimes I|\sigma\rangle = |0\rangle \otimes \lambda V|0\rangle + |1\rangle \otimes \mu V|1\rangle$$
$$U \otimes I|\sigma\rangle = (I \otimes V)(\lambda|0\rangle \otimes |0\rangle + \mu|1\rangle \otimes |1\rangle)$$
$$|\sigma\rangle = (U^\dagger \otimes V)(\lambda|0\rangle \otimes |0\rangle + \mu|1\rangle \otimes |1\rangle)$$

$$|\sigma\rangle = (U^\dagger \otimes V)(\lambda|0\rangle \otimes |0\rangle + \mu|1\rangle \otimes |1\rangle) =$$
$$= (U^\dagger \otimes V)CNOT((\lambda|0\rangle + \mu|1\rangle) \otimes |0\rangle) =$$
$$= (U^\dagger \otimes V)CNOT((W|0\rangle) \otimes |0\rangle)$$

for some W unitary







This is actually a **swap**

Slide 5

Tensor test

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

What is the dimension of the matrix ? **8 × 8**

Classical computing

Computers act on number x to produce another number f(x)

Treat these numbers as non-negative integers less than 2^k for some k

Each integer is represented in the computer as a k bit-string

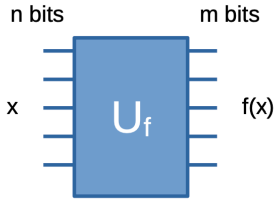
Quantum computing

Quantum computer acts on number x to produce another number f(x)

Treat these numbers as non-negative integers less than 2k for some k

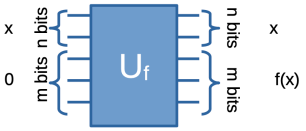
Each integer is represented in the quantum computer with the corresponding computational-basis state of k qubits

General quantum computation



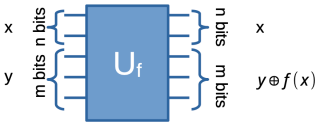
Is this reversible? **NO**

To ensure reversibility we split input and output register



This is standard even if qubits are scarce.

We define the transformation Uf as a reversible transformation (unitary), we give its values for computational basis states, and extend by linearity



$$U_f|x\rangle_n \otimes |0\rangle_m = |x\rangle_n |y \oplus f(x)\rangle_m$$

If the output register is not 0 initially

This is reversible, actually self-inverse

XOR is bitwise

The input register keeps its value

XOR test

If x and y are arbitrary n-bit strings, what is $x \oplus x \oplus y \oplus y$?

$$(x \oplus x) \oplus (y \oplus y) = 0 \oplus 0 = 0.$$

A very important trick

Using two Hadamards we can get an uniform superposition on two qubits

$$H \otimes H|0\rangle \otimes |0\rangle = (H|0\rangle) \otimes (H|0\rangle) = \left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) \otimes \left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) = \frac{1}{2}\left(|00\rangle + |01\rangle + |10\rangle + |11\rangle\right)$$

We can generalize it to n Hadamards

$$H^{\otimes n}|0\rangle^{\otimes n} = \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle \quad \text{dove} \quad H^{\otimes n} = H \otimes H \otimes \cdots \otimes H \quad (n \text{ volte})$$

Computing on superpositions

If we apply Uf to that superposition, with 0 in the output register, we get by linearity:

$$U_f\left(H^{\otimes n} \otimes 1_m\right)|0\rangle_n|0\rangle_m = \frac{1}{2^{n/2}} \sum_{0 \le x < 2^n} U_f\left(|x\rangle_n|0\rangle_m\right) = \frac{1}{2^{n/2}} \sum_{0 \le x < 2^n} |x\rangle_n|f(x)\rangle_m$$

Quantum parallelism

Is this a miracle?

We get all possible evaluations of f

For just 100 qubits, there are 2^{100} evaluations

This magic is called Quantum Parallelism

Is this a miracle? Well...
We cannot say that the result of the computation is all 2^n evaluations of f

No way to find out what the state is unless we measure

In which case the state collapses in one value!!

When we measure the input register, with equal probability, we get any of the values of x

When we measure the output register, we get the value f(x) for that x

So the result is learning a single random x 0 as well as the value of f in x_0

State collapses to $|x_0\rangle|f(x_0)\rangle$

Nothing more we could learn, could have done this with a classical computer, choosing a random value of x and evaluating f

Quantum “weirdness”

Quantum “weirdness”: the selection of the random x for which f(x) was learned is only made after (!!) the computation has been carried out

Quite possibly long after

No practical difference though

No cloning

If we could copy the output register, then we could learn values of f(x) for many random values of x with one computation

No cloning for quantum!

Not even approximate cloning

No Cloning Theorem

“There is no unitary transformation U that takes the state $|y\rangle_n|0\rangle_n$ into $|y\rangle_n|y\rangle_n$ for arbitrary y”

Proof is immediate consequence of linearity

> 💡 **Linearity test**
>
> If $|y\rangle$ and $|x\rangle$ are qubits and U is a unitary such that: $U|y\rangle|0\rangle = |y\rangle|y\rangle$ and $U|x\rangle|0\rangle = |x\rangle|x\rangle$ what is $U((a|y\rangle + b|x\rangle)|0\rangle)$?

It follows from linearity that:

$$U((a|y\rangle + b|x\rangle)|0\rangle) = aU(|y\rangle|0\rangle) + bU(|x\rangle|0\rangle) = a|y\rangle|y\rangle + b|x\rangle|x\rangle$$

But since U clones arbitrary inputs we have:

$$U((a|y\rangle + b|x\rangle)|0\rangle) = (a|y\rangle + b|x\rangle)(a|y\rangle + b|x\rangle) = (a|y\rangle + b|x\rangle)^2 = a^2|y\rangle|y\rangle + b^2|x\rangle|x\rangle + ab|y\rangle|x\rangle + ab|x\rangle|y\rangle$$

Cloning compatible with linearity only if

$$ab|y\rangle|x\rangle + ab|x\rangle|y\rangle = 0$$

Only possible if one of a and b are 0

No Approximate Cloning Theorem

The ability to clone to a reasonable degree of approximation would also be useful

But this is impossible as well

Suppose U approximately clones:

$$U|y\rangle|0\rangle \sim |y\rangle|y\rangle \;\; and \;\; U|x\rangle|0\rangle \sim |x\rangle|x\rangle$$

Properties of inner products

Inner products of tensors is ordinary product of inner products:

$$\langle \psi_1 \otimes \psi_2 | \phi_1 \otimes \phi_2 \rangle = \langle \psi_1 | \phi_1 \rangle \langle \psi_2 | \phi_2 \rangle$$

Unitaries preserve inner product:

$$\langle \psi | \phi \rangle = \langle U\psi | U\phi \rangle$$

Inner product of unitary vectors with themselves is 1 $\langle \psi | \psi \rangle = 1$

Given that U preserves inner products:

$$\begin{aligned} \langle y0 | x0 \rangle &\sim \langle yy | xx \rangle \\ \langle y | x \rangle \langle 0 | 0 \rangle &\sim \langle y | x \rangle \langle y | x \rangle \\ \langle y | x \rangle &\sim \langle y | x \rangle^2 \end{aligned}$$

True only if inner product close to 0 (orthogonal) or to 1 (equal)

Is this it for quantum?

We can be more clever, apply more unitaries to the qubits before or after applying Uf

We can learn something about the relations between different values of f(x)

We however lose the information of f(x)

This tradeoff of information is typical of physics: Heisenberg Uncertainty principle

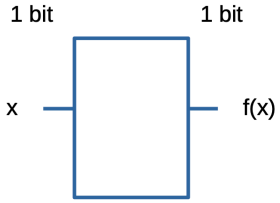
Heisenberg Uncertainty principle, sostanzialmente è un trade-off tra tipo di informazione Vs Relazione tra diversi valori (Come Deutch sappiamo che è costante o meno senza conoscere i valori)

Lezione 5

Is this it for quantum?

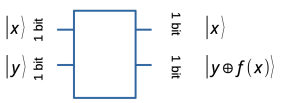
- Measuring destroys most of the information
- We can be more clever, apply more unitaries to the qubits before or after applying Uf
- We can learn something about the relations between different values of f(x)
- We lose the information of f(x)
- This tradeoff of information is typical of physics: Uncertainty

The setup

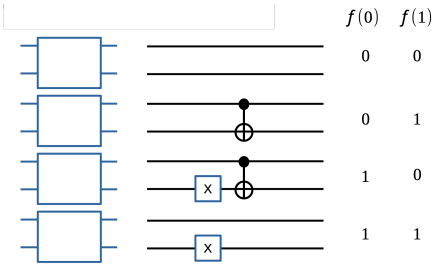


Both input and output registers contain one bit; Functions f that take

The setup, in quantum



The possibilities



one bit to one bit; Two different ways
to think about such f

How many functions?

How many different functions $f : \{0,1\}^n \rightarrow$ {0,1} are there that take as input n bits and output one bit?

- Uf is one of 4 possibly functions (table)
- We are given a black box that calculates one of the 4 f's by performing

$$U_f|x\rangle \otimes |y\rangle = |x\rangle \otimes |y \otimes f(x)\rangle$$

- The black box performs one of the four computations, but we don't know which on
- We are allowed to use the box only once, what can we learn about f?

Deutsch's problem

We want to learn if f is constant (f(0)=f(1), satisfied by f_0 and f_3) or not, with one application of the black box.

With a classical computer, we can either learn the value f(0) or f(1), so we can learn whether the function is one of (f0, f1) with f(0)=0 or (f2, f3) with f(0)=1

A classical computer needs two queries to Uf to determine if it is constant or not!

With a quantum computer we can do better!

Without learning any information about the values of f(0) or f(1), we can determine with one application of the black box if f is constant or not.

There is another way to look at Deutsch's problem, which gives it nontrivial mathematical content

One can think of x as specifying a choice of two different inputs to an elaborate subroutine that requires many additional Qubits, and one can think of f (x) as characterizing a two-valued property of the output of that subroutine

For example f (x) might be the value of the millionth bit in the binary expansion of (2 + x) so that f (0) is the millionth bit in the expansion of √2 while f (1) is the millionth bit of √3

In this case the input register feeds data into the subroutine and the subroutine reports back to the output register.

During the calculation the input and output registers will in general become entangled with the additional Qubits used by the subroutine

If the entanglement persists to the end of the calculation, the input and output registers will have no final states of their own, and it will be impossible to describe the computational process as the simple unitary transformation we saw earlier

However, it is possible to set things up so that at the end of the computation the additional Qubits required for the subroutine are no longer entangled with the input and output registers, so that the additional Qubits can indeed be ignored

The simple linear transformation then correctly characterizes the net effect of the computation on those two registers.

Under the first interpretation, Deutsch's problem answers the question of whether f is or is not constant, allowing one to learn something about the nature of the black box that executes Uf without actually opening it up and looking inside.

Under the second interpretation, it becomes the nontrivial question of whether the millionth bits of √2 and √3 agree or disagree.

Under either interpretation, to answer the question with a classical computer we can do no better than to run the black box twice, with both 0 and 1 as inputs, and compare the two outputs.

Attempt 1: Superposition

To solve Deutsch's problem we could try preparing the input register in superposition of 0 and 1

$$U_f(H \otimes 1)(|0\rangle \otimes |0\rangle) = \frac{1}{\sqrt{2}}|0\rangle \otimes |f(0)\rangle + \frac{1}{\sqrt{2}}|1\rangle \otimes |f(1)\rangle$$

We can measure, and get either 0, f(0) or 1, f(1), but there is no improvement over classical computation

The trick

We can pre and post process the state to yield what we want

$$
\begin{aligned}
&U_f(H\otimes H)(X\otimes X)(|0\rangle\otimes|0\rangle) = U_f(H\otimes H)(|1\rangle\otimes|1\rangle) = \\
&= U_f\frac{|0\rangle-|1\rangle}{\sqrt{2}}\frac{|0\rangle-|1\rangle}{\sqrt{2}} = U_f\frac{1}{2}(|0\rangle|0\rangle-|1\rangle|0\rangle-|0\rangle|1\rangle+|1\rangle|1\rangle) = \\
&= \frac{1}{2}(|0\rangle|0\oplus f(0)\rangle-|1\rangle|0\oplus f(1)\rangle-|0\rangle|1\oplus f(0)\rangle+|1\rangle|1\oplus f(1)\rangle) = \\
&= \frac{1}{2}(|0\rangle|f(0)\rangle-|1\rangle|f(1)\rangle-|0\rangle|1\oplus f(0)\rangle+|1\rangle|1\oplus f(1)\rangle) =
\end{aligned}
$$

Vari casi:

1. f(0)=f(1), the output state is:

$$
\begin{aligned}
&\frac{1}{2}\left(|0\rangle|f(0)\rangle-|1\rangle|f(0)\rangle-|0\rangle|1\oplus f(0)\rangle+|1\rangle|1\oplus f(0)\rangle\right) = \\
&= \frac{1}{2}\left((|0\rangle-|1\rangle)|f(0)\rangle-(|0\rangle-|1\rangle)|1\oplus f(0)\rangle\right) = \\
&= \frac{1}{2}(|0\rangle-|1\rangle)\left(|f(0)\rangle-|1\oplus f(0)\rangle\right)
\end{aligned}
$$

2. $f(0)\neq f(1), f(1)=|1\otimes f(0)\rangle$ the output state is:

$$
\begin{aligned}
&\frac{1}{2}\big(|0\rangle|f(0)\rangle-|1\rangle|1\oplus f(0)\rangle-|0\rangle|1\oplus f(0)\rangle+|1\rangle|1\oplus 1\oplus f(0)\rangle\big) = \\
&= \frac{1}{2}\big(|0\rangle|f(0)\rangle-|1\rangle|1\oplus f(0)\rangle-|0\rangle|1\oplus f(0)\rangle+|1\rangle|f(0)\rangle\big) = \\
&= \frac{1}{2}\big((|0\rangle+|1\rangle)|f(0)\rangle-(|0\rangle+|1\rangle)|1\oplus f(0)\rangle\big) = \\
&= \frac{1}{2}(|0\rangle+|1\rangle)\big(|f(0)\rangle-|1\oplus f(0)\rangle\big)
\end{aligned}
$$

Apply H to the *input register:*

1. f(0)=f(1), the output state is:

$$
\begin{aligned}
&\frac{1}{2}(|0\rangle-|1\rangle)\left(|f(0)\rangle-|1\oplus f(0)\rangle\right) = *Applying\ H*\\
&= \frac{1}{2}(H\oplus 1)(|0\rangle-|1\rangle)\left(|f(0)\rangle-|1\oplus f(0)\rangle\right) = \\
&= \frac{1}{2}|1\rangle\big(|f(0)\rangle-|1\oplus f(0)\rangle\big)
\end{aligned}
$$

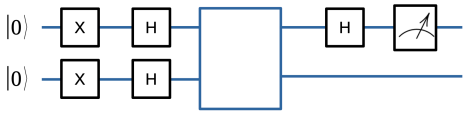
2. $f(0)\neq f(1), f(1)=|1\otimes f(0)\rangle$ the output state is:

$$
\begin{aligned}
&\frac{1}{2}(|0\rangle+|1\rangle)\big(|f(0)\rangle-|1\oplus f(0)\rangle\big) = *ApplyingH*\\
&= \frac{1}{2}(H\oplus 1)(|0\rangle+|1\rangle)\big(|f(0)\rangle-|1\oplus f(0)\rangle\big) = \\
&= \frac{1}{2}|0\rangle\big(|f(0)\rangle-|1\oplus f(0)\rangle\big)
\end{aligned}
$$

Measure the *input register* to decide if f is constant (get 1), or not (get 0)

We learn whether the function is constant or not

$$
\begin{aligned}
&(H\otimes 1)U_f(H\otimes H)(X\otimes X)(|0\rangle\otimes|0\rangle) = \\
&|1\rangle\frac{1}{\sqrt{2}}\left(|f(0)\rangle-|1\oplus f(0)\rangle\right), \quad \text{if } f(0)=f(1)\\
&|0\rangle\frac{1}{\sqrt{2}}\left(|f(0)\rangle-|1\oplus f(0)\rangle\right), \quad \text{if } f(0)\neq f(1)
\end{aligned}
$$



Instead of querying the values of f(0) , f(1) the algorithm queries the XOR f(0) $\oplus$ f(1)

However, we learn nothing about output, since it is in uniform superposition of 0 and 1 (remember that f(0) and 1 $\oplus$ f(0) always have opposite values)

Nice trick, but who really cares? Saving one query seems not so useful

> The idea - querying in superposition and putting the answer into the input register - is much broader and more powerful

There is a straight line from Deutsch's problem to Shor's factoring algorithm. You will get there in the second module

Our intuition fails

Intuitively, X and H work on single qubits, and Uf is the identity on the first qubit. Our intuition says that the first qubit can have no information on the output of f. Yet, mathematics tells us that we find the result exactly on the first qubit. This is due to entanglment: when entangled, one qubit can impact the other. Intuitively, which qubit should influence which qubit in the following two circuits? We can se all the phase to convert one to the others.

Actually, they define the same unitary!

$$|1\rangle \otimes |0\rangle$$
$$CNOT(|1\rangle \otimes |0\rangle)$$
$$CNOT((H \otimes H)|1\rangle|0\rangle)$$
$$CNOT(\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle))$$
$$CNOT(\frac{1}{\sqrt{2}}(|00\rangle + |01\rangle - |10\rangle - |11\rangle))$$
$$applichiamo\ la\ CNOT\ a\ tutti\ (swap\ degli\ ultimi\ due)$$
$$\frac{1}{\sqrt{2}}(|00\rangle + |01\rangle - |11\rangle - |10\rangle)$$
$$\frac{1}{\sqrt{2}}(|0\rangle(|0\rangle + |1\rangle) - |1\rangle(|0\rangle + |1\rangle)) === (H \otimes H)|1\rangle|0\rangle(H \otimes H)\ semplifichiamo\ (H \otimes H)$$
$$\frac{1}{\sqrt{2}}((|0\rangle - |1\rangle)(|0\rangle + |1\rangle)) -> *(H \otimes H) -> |1\rangle|0\rangle$$
$$\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$
$$H|1\rangle \otimes H|0\rangle -> (H \otimes H)(|0\rangle - |1\rangle) -> |0\rangle - |1\rangle$$

Incastonare una CNOT in tra due H dovrebbe far scambiare i ruoli di controllo e target della CNOT stessa

Lezione 6

Bernstein-Vazirani Problem

Bitwise Inner Product

Let x=($x_0, \ldots, x_n$) and a=($a_0, \ldots, a_n$) be two integers, represented as n-bit strings.

The bitwise inner product of x and a, denoted x a · modulo 2 is:

$$x_0a_0 \otimes x_1a_1 \otimes \ldots \otimes x_na_n$$

Binary arithmetic test

Let a=$a_n...a_0$ be an n-bit binary string (encoded as unsigned integer). What is the number a expressed in the decimal system?

What is the value of the m-th bit of a?

Bernstein-Vazirani

Let a be an unknown non-negative integer less than 2^n

Represent it as an n-bit string

Let f(x) = a x =$x_0a_0 \otimes x_1a_1 \otimes \ldots \otimes x_na_n$

Suppose we have an oracle (subroutine) that given x, it gives you f(x)

How many times do we need to call the oracle to determine a?

Classically?

We could learn the n bits of a by applying f to the n values x = 2^m, 0 ≤ m < n

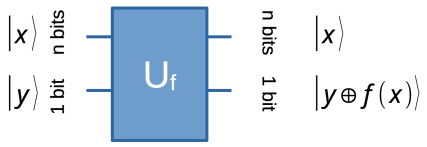
Each invocation tells me a bit of a

Totally, n invocations of the subroutine

With quantum we can ask once!

With some tricks...

The setup, in quantum



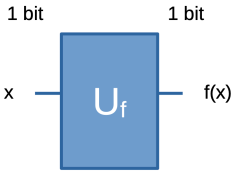
U_f applied to the computational basis state $|x\rangle_n|y\rangle_1$ flips the value y of the output register if f(x)=1

- $U_f|x\rangle_n|0\rangle$
- $U_f|x\rangle_n|1\rangle$

Quindi abbiamo 2 casi, con due esiti per caso:

- $U_f|x\rangle_n|0\rangle = |x\rangle_n|0 \otimes f(x)\rangle =$
  - $|x\rangle_n|0\rangle$ if $f_x = 0$
  - $|x\rangle_n|1\rangle$ if $f_x = 1$
- $U_f|x\rangle_n|1\rangle = |x\rangle_n|1 \otimes f(x)\rangle =$
  - $|x\rangle_n|1\rangle$ if $f_x = 0$
  - $|x\rangle_n|0\rangle$ if $f_x = 1$

The setup



Both input and output registers contain one bit

Functions f that take one bit to one bit

Two different ways to think about such f

The trick

It is useful here as well to set the output register to $HX|0\rangle = H|1\rangle = \frac{1}{\sqrt{2}}\big(|0\rangle - |1\rangle\big)$

$$U_f|x\rangle_n \otimes \frac{1}{\sqrt{2}}\big(|0\rangle - |1\rangle\big) =$$
$$\begin{cases} \frac{1}{\sqrt{2}}\big(|x\rangle_n \otimes |0\rangle - |x\rangle_n \otimes |1\rangle\big) & \text{if } f(x) = 0 \\ \frac{1}{\sqrt{2}}\big(|x\rangle_n \otimes |1\rangle - |x\rangle_n \otimes |0\rangle\big) & \text{if } f(x) = 1 \end{cases}$$
$$= \frac{1}{\sqrt{2}}(-1)^{f(x)}\big(|x\rangle_n \otimes |0\rangle - |x\rangle_n \otimes |1\rangle\big)$$

This allows to change a bit flip to a sign change

Which is the formula for $H|x\rangle_1$?

$(A)\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)$ $(B)\frac{1}{\sqrt{2}}(|0\rangle-|1\rangle)$

$(C)\frac{1}{\sqrt{2}}(|0\rangle+(-1)^x|1\rangle)$ $(D)|x\rangle$

Risposta C

Recall: $H^{\otimes n}|0\rangle_n = \frac{1}{2^{n/2}} \sum_{0 \leq x < 2^n} |x\rangle_n$

Dalla slide di sopra:

$$H(x) = \frac{1}{\sqrt{2}} \sum_{y \in \{0,1\}} (-1)^{xy} |y\rangle$$

L'Hadamard test sfrutta proprio questa proprietà per misurare valori attesi di operatori quantistici, sfruttando l'interferenza tra i termini |+ ⟩ 1⟩ e − ⟩ 1⟩. We need to generalize it to n qubits

▼ Hadamard test, level up

Hadamard test, level up $H^{\otimes}|n\rangle_n$?

$$\begin{aligned} H^{\otimes n}\big|x\big\rangle_n &= \frac{1}{\sqrt{2}} \sum_{y=0}^{1} (-1)^{x_0 y} \big|y\big\rangle \otimes \ldots \otimes \frac{1}{\sqrt{2}} \sum_{y=0}^{1} (-1)^{x_{n-1} y} \big|y\big\rangle \\ &= \frac{1}{2^{n/2}} \sum_{0 \leq y < 2^n} (-1)^{x \cdot y} \big|y\big\rangle_n \end{aligned}$$

Since x·y is used as exponent of -1, only its value mod 2 matters

The algorithm

- Prepare the input and output register

$$(H^{\otimes n} \otimes H)|0\rangle^{\otimes n}|1\rangle = \left(\frac{1}{2^{n/2}} \sum_x |x\rangle \right) \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

- Apply the function oracle

$$\begin{aligned} & U_f(H^{\otimes n} \otimes H)|0\rangle_n|1\rangle_1 \\ &= U_f \left(\frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle_n \right) \frac{1}{\sqrt{2}}\big(|0\rangle - |1\rangle\big) \\ &= \left(\frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} (-1)^{f(x)} |x\rangle_n \right) \frac{|0\rangle - |1\rangle}{\sqrt{2}} \end{aligned}$$

Math test

Let a =(a1,a2), y= (y1,y2) be arbitrary 2-bit strings such that y is not the same as x. What is $\sum_{x1,x_2 \in \{0,1\}} (-1)^{x(y+a)}$?

a. 0

b. 4

c. -4

d. a*y

- Apply Hadamard to the input register

$$\begin{aligned} & (H^{\otimes n} \otimes 1) U_f (H^{\otimes n} \otimes H)|0\rangle_n|1\rangle_1 \\ &= \left(\frac{1}{2^{n/2}} H^{\otimes n} \sum_{x=0}^{2^n-1} (-1)^{f(x)} |x\rangle_n \right) \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\ &= \left(\frac{1}{2^n} \sum_{x=0}^{2^n-1} \sum_{y=0}^{2^n-1} (-1)^{f(x)+x \cdot y} |y\rangle_n \right) \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\ &= \left(\frac{1}{2^n} \sum_{0 \leq x < 2^n} \sum_{0 \leq y < 2^n} (-1)^{f(x)+x \cdot y} |y\rangle_n \right) \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = \\ &= \left(\frac{1}{2^n} \sum_{0 \leq x < 2^n} \sum_{0 \leq y < 2^n} (-1)^{a \cdot x + x \cdot y} |y\rangle_n \right) \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = \\ &= \left(\frac{1}{2^n} 2^n |a\rangle_n \right) \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\ &= |a\rangle_n \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{aligned}$$

$$\sum_{x_1 x_2 \in \{0,1\}} (-1)^{x \cdot (y+a)} = \sum_{x_1 x_2 \in \{0,1\}} (-$$

Since this is the exponent of -1, only its parity counts

For x_i = 0, the term is 1, for x_i = 1 it is 1 if y_i = a_i, -1 otherwise

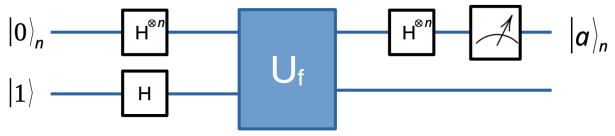
We sum on all the possibilities, hence we alternate x_i = 0 and x_i = 1

If $y_i \neq a_i$ they simplify, if $y_i = a_i$ they add up

If we measure the input register we deterministically get a

The final circuit

$(H^{\otimes n} \otimes 1)U_f(H^{\otimes n} \otimes H)|0\rangle_n|1\rangle_1$



We can restore the output register to $|1\rangle$ with an additional Hadamard

Lezione 7

A different representation for qubits

1. Polar form of complex numbers

Remember that a qubit is represented as $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$

We can equivalently represent coefficients(reali) in polar form

$$\alpha = a + ib$$
$$\alpha = r(cos\ \phi + i\ sin\ \phi)\ where\ r = \sqrt{a^2 + b^2}\ and\ tan\ \phi = \frac{b}{a}$$

This captures the distance from the origin and rotation from the x axis

2. Exponential form

By using Euler formula we get

$$e^{i\phi} = cos\ \phi + i\ sin\ \phi$$
$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = r_1e^{i\phi_1}|0\rangle + r_2e^{i\phi_2}|1\rangle = e^{i\phi_1}(r_1|0\rangle + r_2e^{i(\phi_2-\phi_1)}|1\rangle)$$

Here $e^{i\phi_1}$ is called a global phase, and has no physical relevance (togliamo la fase globale)

Global phase

$$|\psi\rangle = e^{i\phi_1}(r_1|0\rangle + r_2e^{i(\phi_2-\phi_1)}|1\rangle)$$

Let us compute the probability of getting $|0\rangle$

$$Pr_{|\psi\rangle}(0) = |e^{i\phi_1}r_1|^2 = |(cos\phi_1 + i\ sin\ \phi_1)r_1|^2 = r_1^2(cos^2\phi_1 + sin^2\phi_1) = r_1^2$$

The probability does not depend on ϕ_1

If we apply a unitary to vectors differing only for the global phase then we get vectors differing only for the global phase, thanks to linearity

Simpler representation

Since the global phase has no physical meaning we can set it to 0

We also know that $r_1^2 + r_2^2 = 1$, hence we are down to only two parameters

Parameters r_1 and r_2 are on a circle, hence we can derive r_1 and r_2

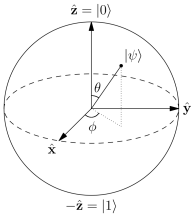
Set $r_1 = cos\frac{\theta}{2}, r_2 = sin\frac{\theta}{2}\ and\ \phi = \phi_2 - \phi_1$

$$|\psi\rangle = cos\frac{\theta}{2}|0\rangle + e^{i\phi}sin\frac{\theta}{2}|1\rangle$$

Bloch sphere

θ and ϕ are real numbers

$$0 \leq \theta < \pi;\ 0 \leq \phi < 2\pi$$
$$x = sin\,\theta\,cos\,\phi$$
$$y = sin\,\theta\,sin\,\phi$$
$$z = cos\,\theta$$



We have a 3-dimensional representation of the states of a qubit

Properties of the Bloch sphere

Unitary transformations correspond to rotations

Two orthogonal vectors are mapped to opposite points on the sphere

Let us prove the latter

Orthogonal vectors

$$|\psi_1\rangle = cos\frac{\theta_1}{2}|0\rangle + e^{i\phi_1}sin\frac{\theta_1}{2}|1\rangle$$
$$|\psi_2\rangle = cos\frac{\theta_2}{2}|0\rangle + e^{i\phi_2}sin\frac{\theta_2}{2}|1\rangle$$

Orthogonal $\langle\psi_1|\psi_2\rangle = 0$

$$\langle\psi_1|\psi_2\rangle = \cos\frac{\theta_1}{2}\cos\frac{\theta_2}{2} + e^{i(\phi_2-\phi_1)}\sin\frac{\theta_1}{2}\sin\frac{\theta_2}{2}$$
$$\cos\frac{\theta_1}{2}\cos\frac{\theta_2}{2} = -e^{i(\phi_2-\phi_1)}\sin\frac{\theta_1}{2}\sin\frac{\theta_2}{2}$$
$$e^{i(\phi_2-\phi_1)} = \cos(\phi_2-\phi_1) + i\sin(\phi_2-\phi_1)\ must\ be\ real$$

Hence $sin(\phi_2 - \phi_1)$=0, thus $\phi_2 - \phi_1 = n\pi$

Looking at the absolute values:

$$|\cos\frac{\theta_1}{2}\cos\frac{\theta_2}{2}| = |\sin\frac{\theta_1}{2}\sin\frac{\theta_2}{2}|$$
$$\sin\frac{\theta_1}{2} = \cos\frac{\pi-\theta_1}{2}$$
$$|\cos\frac{\theta_1}{2}\cos\frac{\theta_2}{2}| = |\cos\frac{\pi-\theta_1}{2}\cos\frac{\pi-\theta_2}{2}|$$

We have $\theta_1 = \theta_2 - \pi$

The solution can be checked in the Bloch sphere or by turning to Cartesian coordinates

The Phase (P) gate

The Phase maps $\alpha|0\rangle + \beta|1\rangle\ to\ \alpha|0\rangle + i\beta|1\rangle$

- Exercise: compute the corresponding matrix
- Exercise: show that P = √Z
- Exercise: show that applying P does not change the probability of getting 0 or 1 after a measurement

Remember that $Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

Phase shifts just change the phase. They do not change the probability of getting 0 or 1

- -

$$R_\phi = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{bmatrix}$$

This is also denoted as $R_z(\phi)$ since it is a rotation around axis z

Relevant phase shifts

$$R_\phi = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{bmatrix}$$
$$R_\pi = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = Z$$
$$R_{\frac{\pi}{2}} = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{2}} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} = P$$

Rotations are related to Pauli matrices

$$R_x(\phi) = \cos\frac{\phi}{2} I - i \sin\frac{\phi}{2} X \qquad X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$
$$R_y(\phi) = \cos\frac{\phi}{2} I - i \sin\frac{\phi}{2} Y \qquad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$
$$R_z(\phi) = \cos\frac{\phi}{2} I - i \sin\frac{\phi}{2} Z \qquad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Projective measurements

We have discussed measurements w.r.t. the computational basis

In general, one can measure w.r.t. other basis, and measure multiple qubits at once

This can be formulated in terms of observables

Observables

An observable is represented by a self-adjoint matrix M, namely M†=M

M can be expressed as M=Σ_m mPm where m are eigenvalues of M

Eigenvalues and eigenvectors

Whenever Av=λv we say that v is an eigenvalue of A and v the corresponding eigenvector

Projective measurements

Given an observable M=Σ_m mP_m we have that there are m possible outcomes of the measurement

The probability p(m) is given by $\langle\psi|P_m|\psi\rangle$

The status after the measurement is $\frac{P_m|\psi\rangle}{\sqrt{p(m)}}$

Computational base recovered

Let us take M = I, and $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$

$|\psi\rangle$ denotes the conjugate transpose of $\langle\psi|$

$$M = |0\rangle\langle 0| + |1\rangle\langle 1| = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} = I$$

Then the probability of getting the first outcome is

$$\langle\psi|P_m|\psi\rangle = [\alpha^* \ \beta^*] \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = [\alpha * \beta *] \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = |\alpha|^2$$

The state collapses to

$$\frac{P_m|\psi\rangle}{\sqrt{p(m)}} = \frac{\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix}}{\sqrt{|\alpha|^2}} = \frac{1}{\sqrt{|\alpha|^2}} \begin{bmatrix} \alpha \\ 0 \end{bmatrix}$$

Further examples

Let us take M = I, and $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$

$$M = |+\rangle\langle+| + |-\rangle\langle-| = \frac{1}{2}\begin{bmatrix}1 & 1\\ 1 & 1\end{bmatrix} + \frac{1}{2}\begin{bmatrix}1 & -1\\ -1 & 1\end{bmatrix} = \frac{1}{2}\begin{bmatrix}2 & 0\\ 0 & 2\end{bmatrix}$$

Then the probability of getting the first outcome is

$$\langle\psi|P_m|\psi\rangle = [\alpha^*\beta^*]\,\frac{1}{2}\begin{bmatrix}1 & 1\\ 1 & 1\end{bmatrix}\begin{bmatrix}\alpha\\ \beta\end{bmatrix} = [\alpha^*\beta^*]\,\frac{1}{2}\begin{bmatrix}\alpha+\beta\\ \alpha+\beta\end{bmatrix} = \frac{1}{2}\left(\alpha^*\alpha + \alpha^*\beta + \beta^*\alpha + \beta^*\beta\right)$$

For α=0 or β=0 you get 0.5 as expected

Multiple measurements

We can perform multiple measurements at once

By using tensor product we can measure qubits independently

We can also measure qubits correlation

Lezione 8

Universality

We have seen a few sample circuits, as well as a few sample gates

In order to build circuits it is useful to have universal sets of gates

What does it mean for a set of gates to be universal?

Which behaviors are all the possible behaviors?

- Quantum computations are described by unitary matrices

Quantum states have continuous sets of values

- It makes no sense to require to build exactly a given unitary matrix
- But we can require to reduce the error as much as desired

Measuring errors

We need to define a distance between matrices

$$E(U,V) = max_{|\psi\rangle}\left\|(U|\psi\rangle - V|\psi\rangle)\right\| \;\; where \left\||\psi\rangle\right\| = \langle\psi|\psi\rangle$$

Matrices are close if they send each vector to close by vectors

Formal definition

A set of gates is universal if, for any integer n≥1, given a unitary matrix U with size 2^n and an error ε we can build using only gates in the set a circuit whose unitary matrix V is such that E(U,V)<ε

Above, n is the number of qubits

What we already know

From quantum state preparation we know that any set of 1-qubit gates is not universal

- E.g., H is not universal

Rationale: any set of 1-qubit gates cannot transform a non-entangled state into an entangled one

There are other criteria to show that some sets of gates are NOT universal

Need for superposition

Any set of classical gates is not universal

They send classical states to classical states

- E.g., you can not create Hadamard using classical gates

- E.g., CNOT is not universal

Need for complex coefficients

Composing matrices with real coefficients always produces matrices with real coefficients

It will never be possible to approximate matrices with coefficients with non-zero imaginary part

- E.g., the set {CNOT, H} is not universal

Anything else?

Actually yes, but it starts getting more complex

The set {CNOT,H,P} satisfies all the conditions we have seen (hence can show many relevant quantum effects) but it is not universal

Circuits obtained from them are called stabilizer circuits

You will find them again when discussing error correction

Entangling gates

A 2-qubit gate is an entangling gate if there is an input product state such that the output is not a product state

That is, the gate can create entanglement

- E.g., CNOT is an entangling gate

A first universal set

Any set composed of all 1-qubit gates and an entangling gate is universal

- E.g., all 1-qubit gates and a CNOT

We will not give a proof of this

We will however refine this result

A first result: limitations

Any set composed of all 1-qubit gates and an entangling gate is universal

This provides us a universal set of gates which is infinite

Not really satisfactory

A first universal set, refined

Any set composed of all 1-qubit gates and an entangling gate is universal

However, any 1-qubit gate is a rotation in the Bloch sphere

Any rotation can be decomposed into rotations around any two non-parallel axis

Any set composed of all 1-qubit gates corresponding to rotations around two non-parallel axes and an entangling gate is universal

Approximating rotations

Fix an axis and consider a rotation around it of φ radians

Applying n gates in a row gives rotations of nφ

However, rotations of 2π are the identity

If the ration between φ and 2π is irrational then we will approximate all the rotations around the considered axis

Fix an error ε and divide the circle in portions of size ε

Put in the circle all values nφ

Since there are at most 2π/ε portions, for n>2π/ε we have at least two points, corresponding to tries n_1 and n_2, closer than ε (pigeonhole principle)

Hence applying our rotation $n_2 - n_1$ times allows us to move less than ε (and more than 0 due to irrationality)

Hence we can cover the whole circumference with step less than ε

A finite universal set

Any set composed of two 1-qubit gates corresponding to rotations around two non-parallel axes whose step has an irrational ratio with 2π and an entangling gate is universal

Consider

$$R_{\frac{\pi}{4}} = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{bmatrix} = T$$

One can show that HTHT and THTH satisfy the conditions above

- Hence {H,T,CNOT} is a universal set of gates

Lezione 9

Implement Boolean functions

We know that NAND gate is (classically) universal

If we could obtain NAND with quantum gate then we could implement any Boolean function

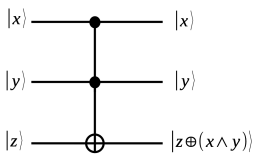
However, NAND is not reversible

Toffoli gate

Also known as controlled-controlled NOT

3-qubit gate, negates the third qubit iff the first two are both 1

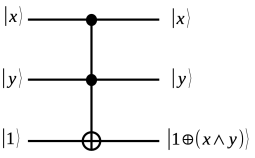
That is, if their AND is 1



From Toffoli to NAND

Toffoli can implement NAND

We need an ancilla bit

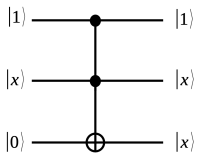


From Toffoli to fanout

In classical computing we can also duplicate bits

We can do this as well with Toffoli

Not in contrast with the no-cloning theorem since we are only duplicating classical bits



From Toffoli to classical

Toffoli can implement both NAND and fanout

Hence Toffoli can implement any Boolean function

We need many ancilla bits

Would CNOT be enough?

NOT is not enough

Given a CNOT circuits, all outputs can be expressed as XOR of inputs

Select an input: the set of outputs which include the selected input in their expression flip, the others stay the same

The set of outputs that flip does not depend on the actual values of other inputs

Toffoli has a different behavior: the output flips by flipping the first bit only if the second is 1

Probabilistic computing

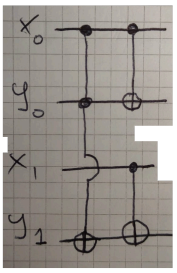
Quantum provides perfect probabilistic choices

Put a state in a superposition and measure it

Hence quantum can simulate any algorithm doable with classical computing + probabilities

Exercise

Construct a quantum circuit to add two two-bit numbers x and y modulo 4. That is, the circuit should perform the transformation$|x, y\rangle \ |x, x + y \bmod 4\rangle$



Simulating quantum circuits

We can always simulate quantum circuits using classical gates up to an arbitrary small approximation

Basic idea: store all the coefficients in the vector representing the state, and mimic what unitaries do

Note: for n qubits, we have 2^n coefficients, hence we have in the worst case an exponential slowdown

Consequences

Quantum circuits cannot compute non-classically computable functions (e.g., termination)

Quantum computing can provide at most an exponential speed-up

There is no exponential speed-up without entanglement

> If I have no entanglement I can decompose the state in a tensor product, which can be represented using 2^n coefficients

Modulo 1 - Parte I

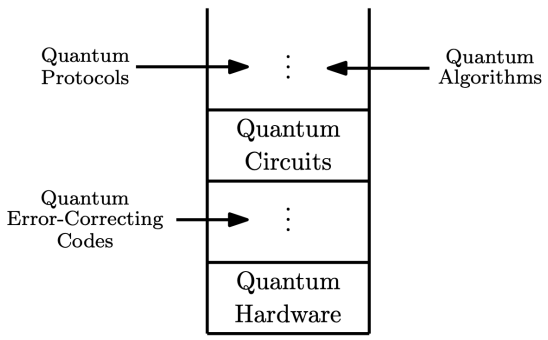
ci sono protocolli che non so algoritmi, tipo per la sicurezza.

altre forme di comunicazione che non sono classica sfruttando l'entanglement quantistico.

Quindi queste cose vanno integrate anche dal libro perche le slide non so abbastanza?

thelepaty non c'è nel libro

What?



How?

- The most basic concepts are described on these slides, which will be made available to all students.
  - Please do not expect all details to be present in the slides, which are of course meant to be just a summary.
- Some details will be presented at an interactive whiteboard, whose transcripts will be made available themselves to all students.
  - That will be extremely useful when giving formal proofs, of which we will se just very few.
- Many nice textbooks about quantum computing which cover all what we say are available. We will follow the very nice textbook by Mermin:

- N. David Mermin. Quantum Computer Science: An Introduction. Cambridge University Press, New York, NY, USA. 2007.

- Please feel free to ask any question during lectures, or to write me at ugo.dallago@unibo.it

Why?

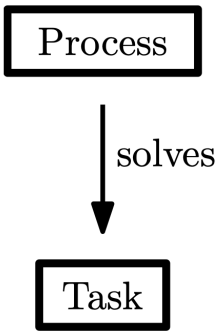
- Quantum computing is considered to have very strong potential for revolutionize the way certain computational problems are solved, and this is why there is a strong interest around it.
- It's important to understand where this potential comes from.
  - You know what a quantum circuit is.
  - But perhaps you do not know in which sense quantum circuits can be helpful.
- But beware: most of what we are going to say is about the quantum circuit model, and nobody currently knows whether this model is realistic when the number of qubits grows.
  - That is why we will also talk about quantum error correcting codes.

This Slideset

- A glimpse of circuit complexity.
  - Classical circuit complexity.
  - Quantum circuit complexity.
  - How all this relates to Turing machines.
- Some basic quantum algorithmics.
  - A Recap on Deutsch and Bernstein-Vazirani Algorithms.
  - Simon Algorithm.
  - Shor Algorithm.
    - We will try to give most details, but we will perhaps skip some of them
  - Grover Algorithm

Modulo 1 - Part II - Basic Circuit Complexity

Tasks and Processes

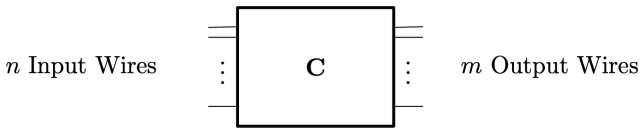


Tasks

- A string from an alphabet Σ is any ordered, finite sequence of symbols from Σ.
- The set of all strings of length n from Σ indicated as Σ^n.
- A particularly interesting alphabet is Σ = {0, 1}.
- We are interested in tasks formulated as mathematical functions from $\{0,1\}^n$ to $\{0,1\}^m$. The set of all such **boolean functions** is indicated as F_n^m .
- Sometimes, we are also interested in **parametric boolean functions** namely in functionals which turn a function in F_q^p to a function in F_n^m.
- An example when p=q=2,n=0 and m=1 is the task of checking whether the input function f ∈ F_q^p returns 00 for all possible inputs.

Classical Processes

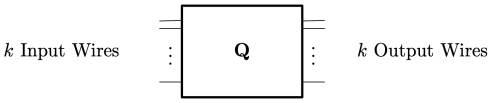
A classical process will be a boolean circuit :



One such circuit C is said to **compute** the boolean function fC in Fmn such that f_C(x) = y precisely when C when executed on x on the n input wires produces y on its m output wires.

Quantum Processes

Similarly, a quantum process is just a quantum circuit :



We can generalize all what we have said about boolean circuits, keeping in mind that

We can generalize the view above to parametric boolean functions by considering boolean circuits with a special gate corresponding to the parameter function in F_q^p, called a parametric boolean circuits.

The size of C is a measure of how many resources C would consume when executed.

- The size of C, that we write as |C|, can be measured in different ways, e.g., as the number of gates, as the width, or as the depth.
- An interesting size parameter of any parametric boolean circuit C is the number of calls C to the special gate computing the parameter function, which we indicate as $|C|_P$.
- Quantum circuits, due to measurements, can have a probabilistic behaviour. We then say that one such circuit computes a function $f_Q \; in \; F_m^n$ if it does so up to a small probability of error.
- The circuit Q, if we do not consider measurements, must be reversible. It is thus convenient to assume that for such a circuit to compute f ∈ F_m^n , it must be that Q has n + m + r inputs and n + m + r outputs an that, with high probability, on input |x⟩ ⊗ |y⟩ ⊗ $|0^{\otimes r}\rangle$ it produces (with high probability) the value |x⟩ ⊗ |f(x) ⊕ y⟩ ⊗ |ψ⟩ in output.
  - The r qubits are auxiliary, and their output value is not relevant.

The Fundamental Question

The question now is the following: Is there any (parametric) function f such that there is a quantum circuit Q computing f such that all (known) boolean circuits C computing f are such that |Q| < |C|?

We will give some examples of such functions:

- Starting from toy functions, for which you have already seen the corresponding algorithms.
- And ending in relevant functions.
- Many of them will be parametric functions.

A Bridge to "Uniform" Complexity Theory

In classical computation, a circuit family is a family $\{C_n\}_{n\in N}$ such that for every n ∈ N, the circuit C_n computes a function in F_1^n.

> Such a circuit family $\{C_n\}_{n\in N}$ can thus be seen as computing a function f$\{C_n\}$: {0, 1}* → {0, 1} where {0, 1}* = $\bigcup_{n=0}^{\infty}\{0,1\}^n$.

Again, something very similar can be said for quantum circuits.

The class of languages (i.e., subsets of {0,1}*) which can be computed by classic circuit families having polynomially bounded size and being generated in polynomial time through algorithms is precisely P.

A very similar result holds for polysize quantum circuit families generated in classical polynomial time and a class called BQP.

Modulo 1 - Part III - Basic Quantum Algorithmics

The Deutsch Problem

In the Deutsch Problem, we are interested in determining whether a parameter function f ∈ F_1^1 is such that f(0) = f(1). In doing so, we can use a gate computing f.

Classically, we do not have any hope of getting the job done without invoking the f twice.

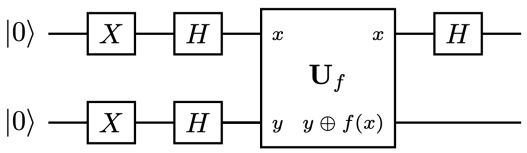
In quantum computing, one can do strictly better, and invoke f just once!

In doing so, we will assume that there is a gate U_f computing f without using any auxiliary qubit:

$$U_f(|x\rangle|y\rangle) = |x\rangle|y \oplus f(x)\rangle$$

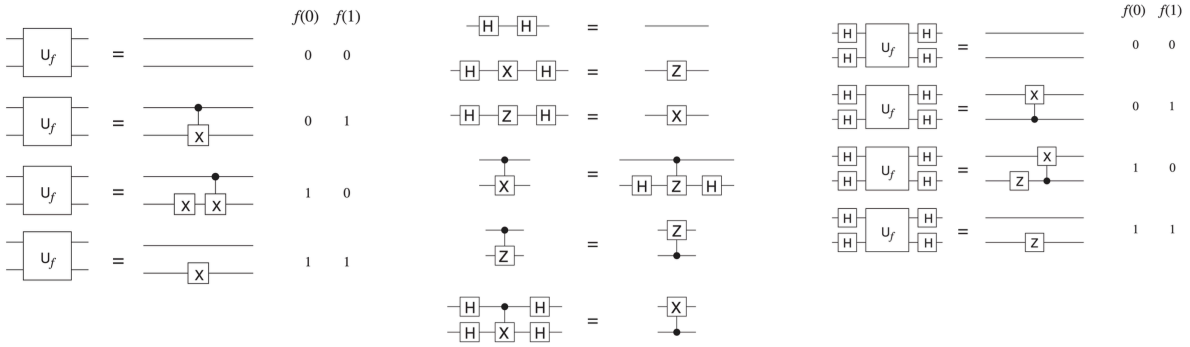
The Deutsch Circuit $C_{DEUTSCH}$

Correctness



$$(\mathbf{H}\otimes\mathbf{1})\mathbf{U}_f(\mathbf{H}\otimes\mathbf{H})(\mathbf{X}\otimes\mathbf{X})(|0\rangle|0\rangle)$$
$$= \begin{cases} |1\rangle\frac{1}{\sqrt{2}}\Big(|f(0)\rangle - |\tilde{f}(0)\rangle\Big), & f(0) = f(1), \\ |0\rangle\frac{1}{\sqrt{2}}\Big(|f(0)\rangle - |\tilde{f}(0)\rangle\Big), & f(0) \neq f(1). \end{cases}$$

Equational Interpretation



The Bernstein-Vazirani Problem

Given a, x $\in \{0,1\}^n$. We write a · x for the bitwise inner-product of a and x:

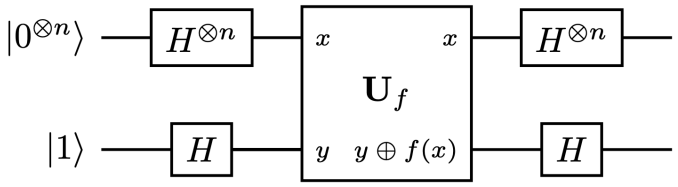
$$a_1x_1 \oplus a_2x_2 \oplus \cdots \oplus a_nx_n$$

The Bernstein-Vazirani problem has to do with parametric functions whose input function fa $\in F_1^n$ is such that f_a(x) = a · x.

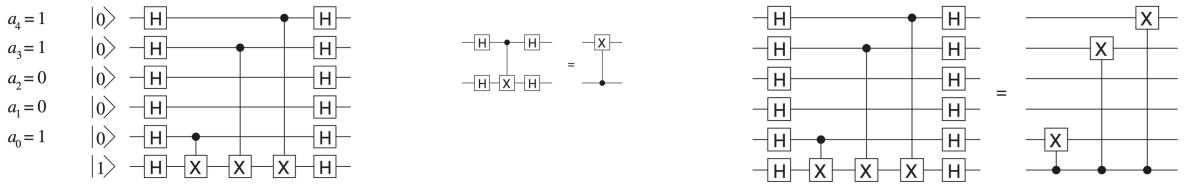
Simply, we want to determine a invoking f_a , but trying to minimize the amount of a times we invoke the function.

Classically, we cannot do that by invoking f_a less than n times. There is a quantum circuit which invokes f_a **just once**.

The Bernstein-Vazirani Circuit C_{BV}



Equational Interpretation



Modulo 1 - Part IV- The Simon Problem

Like in the Bernstein-Vazirani problem, in the Simon Problem we deal with a function f which depends on a $\in \{0,1\}^n$.

The function f is defined differently, and is in F_n^n. There are many such functions for the same a $\in \{0,1\}^n$

We only know that f is periodic modulo a, namely that

$$f(x) = f(y) \iff y = x \oplus a \iff x \oplus y = a$$

for every x, y $\in \{0,1\}^n$.

Like in the Bernstein-Vazirani problem, we want to compute parametric functions whose input function f $\in F_n^n$ is periodic modulo a in the sense above, and we want to determine a minimizing the amount of times one invokes f.

Classically, one cannot succeed (with high probability) unless one invokes f around 2^n times, so **exponentially many** times.

There are, instead, quantum circuits which succeed in determining a by invoking f an amount of time which is **linear** in n.

Classical Strategy

Any classical algorithm querying a function f $\in F_n^n$ which is periodic modulo a accumulates knowledge as follows:

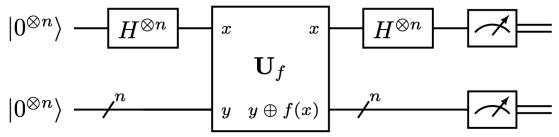
- If the queries done so far, namely

  $$y_1 = f(x_1), ..., y_k = f(x_k)$$

  are such that $y_i \neq y_j$ for every distinct i,j $\in$ {1,...,k}, then the only thing the algorithm knows is that for every distinct i,j $\in$ {1,...,k}, it holds that $a \neq x_i \oplus x_j$.
- If, instead, $y_i = y_j$, then a can be easily computed as $x_i \oplus x_j$.

Since there are only $\frac{k(k-1)}{2}$ pairs of distinct indices in {1, . . . , k}, in the worst case the 2 number of queries is has to be exponential.

The Simon Circuit C_{SIMON}



The Simon Algorithm

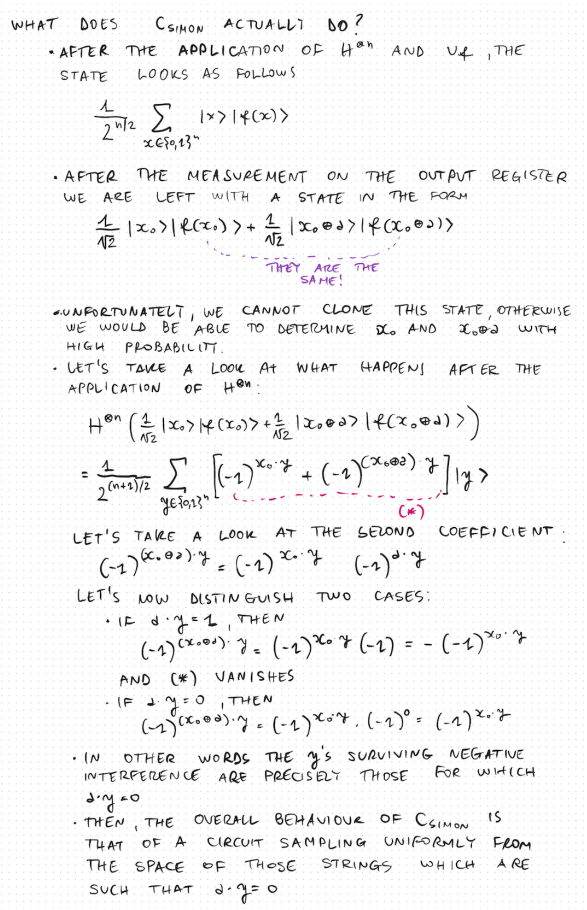
The actual Simon's Algorithm has the following structure:

1. i ← 1.
2. Apply C_{SIMON} and obtain in the first n qubits the value $|x_i\rangle$
3. If i = n+3, then go to step 4, otherwise, increment i by 1 and go back to 2.
4. Solve the linear system of equations

$$\begin{cases} a \cdot x1 = 0 \\ . \\ . \\ a \cdot x_{n+3} = 0 \end{cases} \quad (1)$$

5. If there is a unique non null solution a, then return it, otherwise fail.

The system of equation above has a unique solution with probability at least 23 .



Modulo 1 - Part V - Shor's Algorithm

A Groundbreaking Result

Shor's Algorithm is considered as **one of the major results** in the theory of computation since the inception of the discipline, in the sixties. It is a witnesses of the fact that some useful problems can be solved by quantum circuits of polynomial size in the input n.

This includes:

- Breaking **RSA**, one of the most widely used public-key encryption scheme.
- Integer **Factorization**;
- Discrete **Logarithm**;

A whole sub-field of cryptography, called **post-quantum cryptography**, grew out of all this. Ultimately, however, Shor's Algorithm is an algorithm about **period finding**.

▼ Number-Theoretic Preliminaries — I

let G be the set of all positive integers (including 1) which are strictly less than N and which has no prime factor in common with N.

On G_N, one can define a binary operation, namely multiplication modulo N:

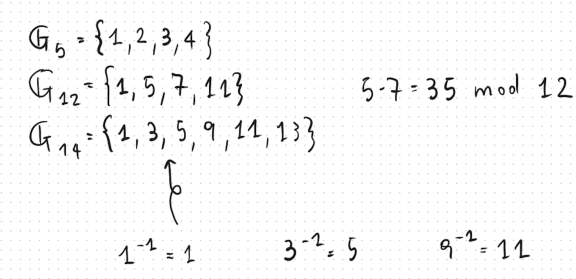
$$(x, y) \rightarrow x \cdot y \; mod \; N$$

Multiplication modulo N is well-defined, associative, and has an identity, namely 1, and has inverses. It is thus a (finite) **group**.

The cardinality Φ(N)∈N of G_N is at moste qual to N−1.

- If N is a prime, then Φ(N)=N−1.
- If N is the product of two prime numbers p and q, then Φ(N)=(p−1)(q−1).
- Euler's theorem tells us that every a ∈ GN , it holds that

$$a^{\Phi(N)} \equiv \; 1 \; mod \; N$$



▼ Number-Theoretic Preliminaries — II

The **order** of n ∈ G_N is the smallest number r such that

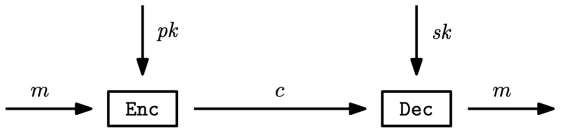
$$n^r \equiv 1 \; mod \; N$$

The order of n always divides Φ(N).

A **subgroup** of G_N generated by n ∈ G_N is the subset ⟨n⟩ of G_N of those elements of G_N which can be written as n^k mod N, where k ∈ Z. If ⟨n⟩ = ⟨m⟩ then n and m have the same order.

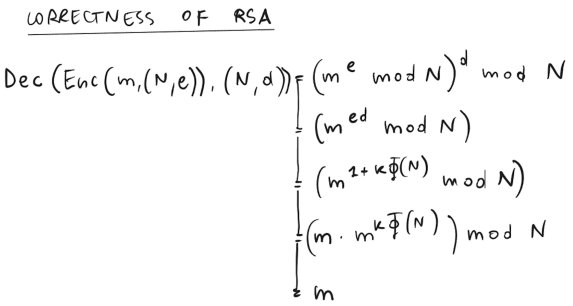
RSA

RSA is one of the most common **public-key** encryption schemes.



Given N = pq product of two (large) primes and e,d such that ed ≡ 1 mod Φ(N), the public key will be (N,e), while the secret key is (N,d).

Correctness of RSA



Messages are simply elements of G_N

Given a message m $\in G_N$ and a public key (N, e), its encryption is Enc(m,(N,e))=m^e mod N.

Given a ciphertext c $\in G_N$ and a private key (N, d), its decryption is Dec(c,(N,d))=c^d mod N.

Noticeably, Dec(Enc(m, (N, e)), (N, d)) = m.

Breaking RSA

Clearly, **one way** of breaking RSA consists in, given N = pq and e $\in G_{(p-1)(q-1)}$ determine d such that

$$ed \equiv 1 \; mod \; (p-1)(q-1)$$

- This way one can determine the private key from the public key.
- We do not even know (p − 1)(q − 1), however.

If we can find, given a and N, the smallest number r ≥ 1 such that

$$c^r \equiv 1 \; mod \; N$$

(where c is the ciphertext) then we can retrieve the message from the cryptogram in a natural way. Such an r is nothing more and nothing less than the period of the function $x \to c^x$ mod N.

Again About Period-Finding

Shor's Algorithm is precisely about finding periods of functions in the form

$$x \to a^x \ mod \ N$$

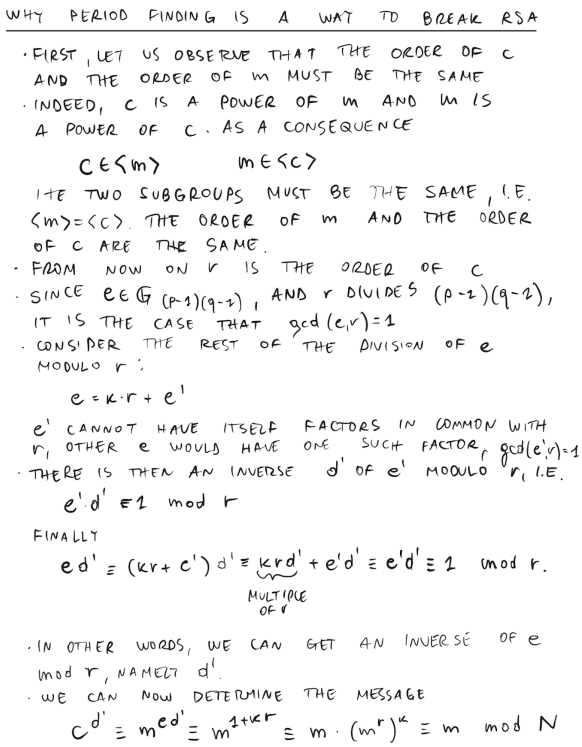
We can thus proceed as in Simon's Algorithm, thus applying a gate for the function above to the result of calling Hadamard functions, and subsequently measuring the output qubits, obtaining f(x0). This way we get the state

$$|\Psi\rangle_n = \frac{1}{\sqrt{m}} \sum_{k=0}^{m-1} |x_0 + kr\rangle \, .$$

where m is the smallest integer such that $mr + x_0 \geq 2^n$

here are two problems, however, namely:

- The presence of x_0
- The fact that f is not necessarily easy to be computed.



QFT

The **Quantum Fourier Transform** is a way to get rid of the term x_0 in the sum above.

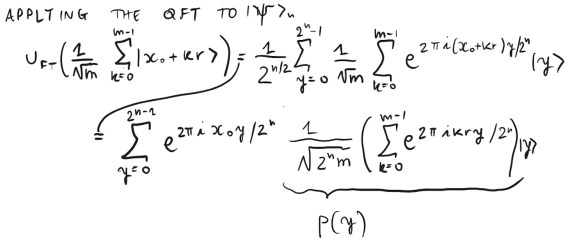
It is defined as a unitary transform U_{FT} such that

$$U_{FT}^n |x\rangle = \frac{1}{2^{\frac{n}{2}}} \sum_{y=0}^{2^n-1} e^{\frac{2\pi i x y}{2^n}} |y\rangle$$

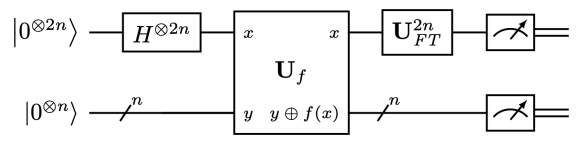
Remarkably, U_{FT}^n can be implemented with a circuit of quadratic size consisting of unary gates, only.

If one applies U_{FT}^n to $|\Psi\rangle_n$, after that measuring all bits, then the probability of observing y when measuring the input qubits is

$$p(y) = \frac{1}{2^n m} \left| \sum_{k=0}^{m-1} e^{\frac{2\pi i k r y}{2^n}} \right|^2$$



Shor's Circuit C_{SHOR}



About Post-Processing and Replication

By measuring y, however, we do not get exacctly what we need, namely the period r. We can get it, however, by way of some **post-processing**, perhaps having to **redo** the quantum computation from scratch.

The probability p(y) has maxima when y is close to integral multiples of $2^{2n}/r$.

If y is within $\frac{1}{2}$ of j2^{2n}/r for some integer j, we have that

$$\left|\frac{y}{2^{2n}} - \frac{j}{r}\right| \leq \frac{1}{2^{2n+1}}$$

ONCE YOU HAVE COMPUTE $y/2^{2n}$ AND

$$\left|\frac{y}{2^{2n}} - \frac{j}{r}\right| \leq \frac{1}{2^{2n+1}}$$

YOU CAN GET j_0, r_0 SUCH THAT $j_0/r_0 = j/r$ AND j_0/r_0 IS IRREDUCIBLE. IN PARTICULAR, THEN r_0 IS A DIVISOR OF r.
YOU THEN CHECK WHETHER r_0 IS ITSELF A PERIOD. ACTUALLY, THE PROBABILITY THAT $r_0 = r$ IS QUITE HIGH.

We can thus learn terms in the form j/r by repeating the process not so many times.

Out of them, we can finally find r.

Fast Exponentiation

Computing the function x→ b^x mod N is of course necessary for Shor's algorithm to work. In particular, we need a (quantum) circuit doing that and not so big. Just iterating the multiplication in the obvious way does not work, because the number of iterations will be in the worst case equal to N. We need a more clever algorithm, called **fast exponentiation**, working on the input register x, the output register y and a work register w.

- Initially, y=1 and w=b.
- We then update y and w in an iterative way, squaring w and multiplying y by w only if the corresponding bit of x is 1.

FAST EXPONENTIATION – EXAMPLE

w	y	x	
b	1	101101 (45)	
b^2	b		1 (LEAST SIGNIFICANT BIT)
b^4	b		2
b^8	b^{1+4}		3
b^{16}	b^{1+4+8}		4
b^{32}	b^{1+4+8}		5
b^{64}	$b^{1+4+8+32}$		6
	b^{45}		

Factoring and Discrete Logarithm

We saw Shor as an algorithm for computing the period of a function in the form

$$x \to b^x \ mod \ N,$$

this way breaking RSA in polynomial time.

However, the same algorithm can be used to solve other problems of interest:

- **Integer Factorization.**
  - After checking if the number N in input is prime, generate 1 < a < N at random, and if a is prime with N, compute the period r of x → a^x mod N. If r is even, then any factor of N is either a factor of $a^{r/2}$ −1 or a factor of $a^{r/2}$ +1.
- **Discrete Logarithm.**

Modulo 1 - Part VI - Grover Algorithm

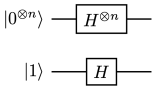
Grover's algorithm solves the following computational problem:

- Input: a boolean circuit (whose structure is not accessible) computing an unknown function f : $\{0,1\}^n$ → {0,1} such that there exists a ∈$\{0,1\}^n$ with f(x) = 1 iff x = a.
- Output: the (unique) string a ∈ $\{0,1\}^n$ such that f (a) = 1.

This is a search problem. In the worst case, any classic algorithm must evaluate f on all the 2^n coordinates.

Grover's Algorithm, by way of a technique called **amplitude amplification**, manages to solve the same problem in time at most O($\sqrt{2}^n$)

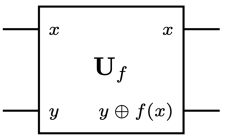
First Component



The resulting state is

$$|\phi\rangle \otimes H(|1\rangle) = \left(\frac{1}{2^{n/2}} \sum_{x \in \{0,1\}^n} |x\rangle\right) \otimes H(|1\rangle)$$

Second Component

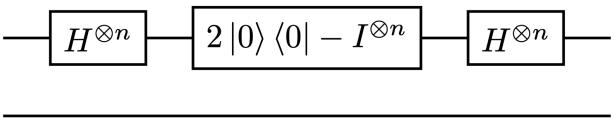


On an input $|x\rangle \otimes H(|1\rangle)$, the output is

$$(-1)^{f(x)} |x\rangle \otimes H(|1\rangle)$$

and so there must be a unitary transformation $\mathbf{V}$ on n qubits capturing its behaviour.

Third Component



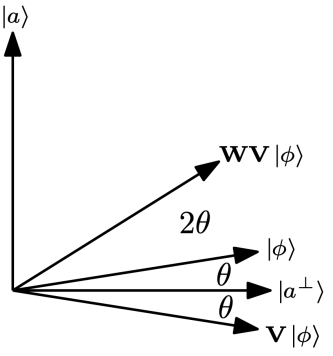
The action of the component on the first n qubits can be seen as that of a unitary transformation, call it $\mathbf{W}$

A Geometric Analysis

Let us take a look at how V and W behave on the states |a⟩ and |φ⟩:

$$\mathbf{V}|a\rangle = -|a\rangle; \qquad \mathbf{V}|\phi\rangle = |\phi\rangle - \frac{2}{2^{n/2}}|a\rangle;$$

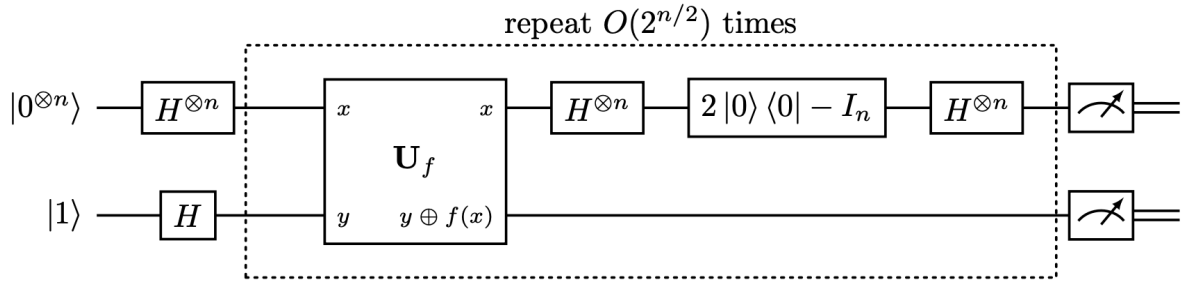
$$\mathbf{W}|a\rangle = \frac{2}{2^{n/2}}|\phi\rangle - |a\rangle \qquad \mathbf{W}|\phi\rangle = |\phi\rangle$$



Applying V and W, then keep the following invariant true: the state is in the form |ψ⟩ ⊗ H(|1⟩), where |ψ⟩ lives in the two-dimensional space of all linear combinations α |φ⟩ + β |a⟩, where α, β ∈ R and $\alpha^2 + \beta^2 = 1$.

The states |a⟩ and |φ⟩ are almost orthogonal, since θ = ⟨a|φ⟩ = $2^{-n/2}$. There is in particular a vector forming a small angle with |φ⟩ and being orthogonal with |a⟩, call it $|a^{\perp}\rangle$. Both V and W are reflections, and their product WV is thus a rotation.

Grover's Circuit C_{GROVER}



Module 2 - part 1

Quantum Protocols - Part I

Circuits vs. Protocols

- Quantum Circuits

- These are sequences of (unitary) operations applied to qubits in a prescribed manner.
- Meant to be executed by quantum hardware or simulated through classical hardware.
- They are seen as ways to compute functions from $\{0,1\}_n$ to $\{0,1\}_m$.

- ▶ Quantum Protocols
  - Communication protocols, i.e., sequence of computation and communication steps, perormed by a set of (possibly distributed) agents.
  - Besides using and exchanging classical data, the agents can use and exchange quantum data in the form of qubits.
  - The underlying task to be solved can be different from the mere computation of a function, e.g.,
    - Transmission or distribution of some information.
    - Consensus.
    - Commitment.
    - ...

What is a Quantum Protocol?

One can formally define what a quantum protocol actually is, and there are many languages and models of quantum protocols.

We will stay very informal, and describe quantum protocols by:

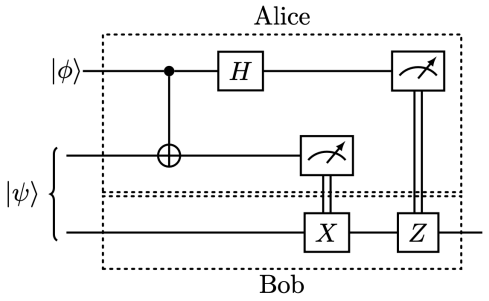
- Either giving a circuit, describing who owns each of the qubits, and/or when they are exchanges between the parties.
- Or/and describing what each of the agents is supposed to do with its data.

Quantum Teleportation - Part II

Teleporting a Qubit?

- Suppose Alice has a qubit she does not want to measure, and that she wants to “send it” to Bob.
- In doing so, se is allowed to actually send classical information, but that she cannot send quantum information.
- Surprisingly, this can be achieved, provided Alice and Bob share a pair on entrangled qubits $|\psi\rangle$, which can be setup prior to the creation of Alice’s qubit.
- The byproduct of the communication is that $|\psi\rangle$ is destroyed.

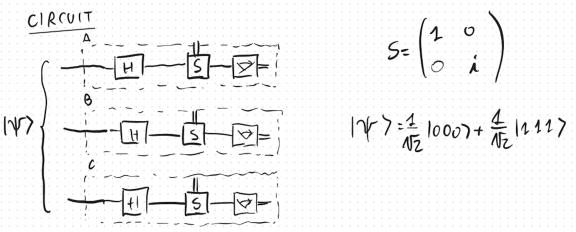
Quantum Teleportation as a Circuit



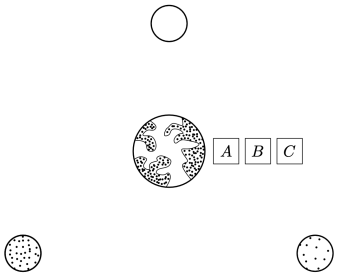
Quantum Pseudotelepathy - Part III

Quantum Pseudotelepahy

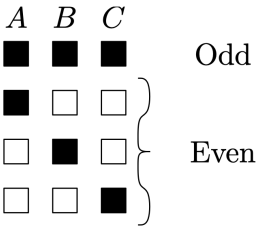
- This term refers to protocols which uses entanglement for the sake of proving that (part of the) communication between the parties can be avoided.
- This is often provably impossible in classical computing.
- We will introduce only a very simple example of quantum pseudotelepathy, through a game.

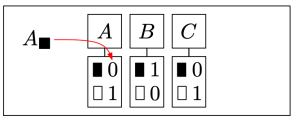


A Game



When do A, B and C Win?



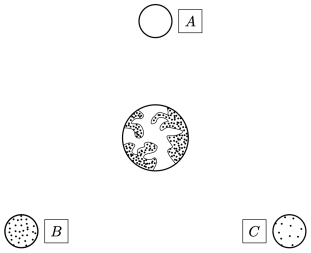


$$A_■ + B_■ + C_■ = \text{Odd}$$
$$A_■ + B_□ + C_□ = \text{Even}$$
$$A_□ + B_■ + C_□ = \text{Even}$$
$$A_□ + B_□ + C_■ = \text{Even}$$

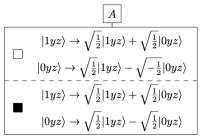
$$2A_■ + 2A_□ + 2B_■ + 2B_□ + 2C_■ + 2C_□ = \text{Odd}$$

A, B e C Cannot Win! in Usual Game

A Quantum Game



Quantum Strategies



If ■ arrives...

$$\sqrt{\tfrac{1}{2}}|111\rangle + \sqrt{\tfrac{1}{2}}|000\rangle \to \tfrac{1}{2}|111\rangle + \tfrac{1}{2}|011\rangle + \tfrac{1}{2}|100\rangle - \tfrac{1}{2}|000\rangle$$

Suppose that A, B, C Receive ■ ...

$$\begin{aligned}\sqrt{\tfrac{1}{2}}|111\rangle + \sqrt{\tfrac{1}{2}}|000\rangle \to\ & \tfrac{1}{4}|111\rangle + \tfrac{1}{4}|110\rangle + \tfrac{1}{4}|111\rangle - \tfrac{1}{4}|110\rangle \\ & + \tfrac{1}{4}|101\rangle + \tfrac{1}{4}|100\rangle - \tfrac{1}{4}|101\rangle + \tfrac{1}{4}|100\rangle \\ & + \tfrac{1}{4}|011\rangle + \tfrac{1}{4}|010\rangle - \tfrac{1}{4}|011\rangle + \tfrac{1}{4}|010\rangle \\ & + \tfrac{1}{4}|001\rangle + \tfrac{1}{4}|000\rangle + \tfrac{1}{4}|001\rangle - \tfrac{1}{4}|000\rangle \\ =\ & \tfrac{1}{2}|111\rangle + \tfrac{1}{2}|100\rangle + \tfrac{1}{2}|010\rangle - \tfrac{1}{2}|001\rangle\end{aligned}$$

LET'S CONSIDER THE CASE IN WHICH THE THREE ALIENS ALL RECEIVE ■:

- AFTER A RECEIVES ■, IT APPLIES H TO ITS QUBIT:

$$\tfrac{1}{2}|1117\rangle + \tfrac{1}{2}|0117\rangle + \tfrac{1}{2}|1007\rangle - \tfrac{1}{2}|0007\rangle$$

- AFTER B AND C RECEIVE ■, AND APPLY H TO THEIR QUBITS, THE STATE BECOMES

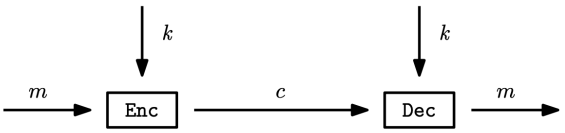
$$\tfrac{1}{4}|1117\rangle + \tfrac{1}{4}|1107\rangle + \tfrac{1}{4}|1117\rangle - \tfrac{1}{4}|1107\rangle$$
$$+ \tfrac{1}{4}|1017\rangle + \tfrac{1}{4}|1007\rangle - \tfrac{1}{4}|1017\rangle + \tfrac{1}{4}|1007\rangle$$
$$+ \tfrac{1}{4}|0117\rangle + \tfrac{1}{4}|0107\rangle - \tfrac{1}{4}|0117\rangle + \tfrac{1}{4}|0107\rangle$$
$$+ \tfrac{1}{4}|0017\rangle + \tfrac{1}{4}|0007\rangle + \tfrac{1}{4}|0017\rangle - \tfrac{1}{4}|0007\rangle$$
$$= \tfrac{1}{2}|1117\rangle + \tfrac{1}{2}|1007\rangle + \tfrac{1}{2}|0107\rangle + \tfrac{1}{2}|0017\rangle$$

THE NUMBER OF 1s IS ODD, EVEN IF IT IS NOT ALWAYS THE SAME

Quantum Key Distribution - Part IV

Private-Key Cryptography

Sometimes, encryption is done symmetrically rather than with distinct public and private keys:



As a matter of fact, one can design (secure) symmetric encryption schemes such that Enc and Dec are quite efficient.

Symmetric encryption schemes, however, pose a challenge: how could we let the sender and receiver **share** a key k in the first place?

Many solutions possible:

- Using a **secure** channel.
- Using **public-key** encryption to share the key, then switching to the private-key setting.
- Relying on **quantum computing**

Exploiting Quantum Channels

Communication channels are traditionally assumed to be just classical.

> Data flowing along the channel can be **observed** without being altered, and the value observed can be **duplicated**

In practice, they often work according to the rules of quantum mechanics.

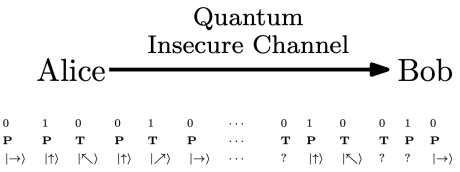
> Observing some piece of data becomes a **quantum measurement**, and duplicating it is forbidden, due to the **No-Cloning Theorem**.

Can this be exploited? Is there a way to take advantage of that?

Working with Two Orthonormal Bases

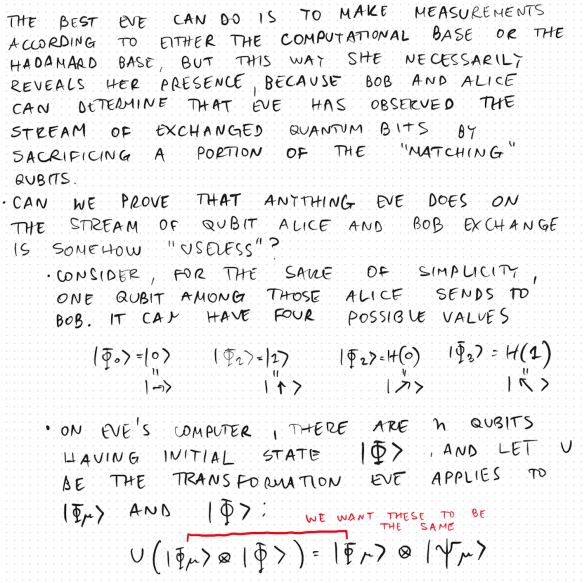
- Both P = {$|\rightarrow\rangle$, $|\uparrow\rangle$} and T = {$|\nwarrow\rangle$, $|\nearrow\rangle$} are orthonormal bases.
- If the state of your system is in P, and you measure it with respect to T you will end up in either $|\nwarrow\rangle$ or $|\nearrow\rangle$ each with probability $\frac{1}{2}$.
- If the state of your system is in T, and you measure it with respect to P you will end up in either $|\rightarrow\rangle$ or $|\uparrow\rangle$ each with probability $\frac{1}{2}$.

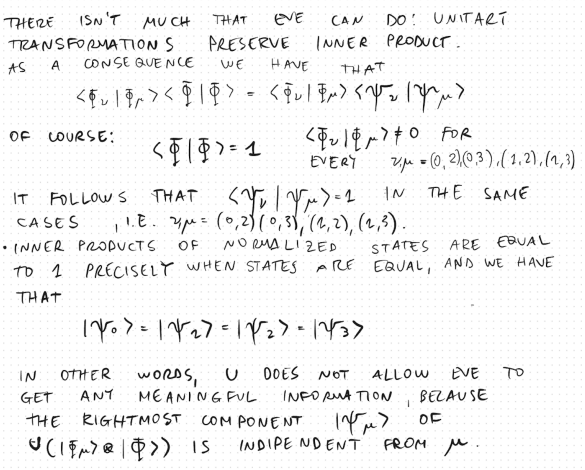
The BB84 Protocol



Alice and Bob, **after** having transmitted sequence of {0, 1}, can just compare their sequences of {T, P}, and they can do it on an insecure channel.

They can also **realize** somebody has seen or altered the sequence, by comparing some (around half) of the exchanged values.





Quantum Commitment - Part V

Commitment Protocols

A **commitment protocol** allows two parties Alice and Bob to interact so as to allow Alice to commit to a chosen value, guaranteeing that:

- The value chosen by Alice remains hidden until Alice decides to reveal it.
- Neither Alice nor Bob can change the value chosen by Alice, i.e., the protocol is **binding**.

1. Nascondiamo il valore
2. Non possiamo cambiarlo una volta scelto
3. Rivelarlo quando vogliamo

In (classical) cryptography, various forms of commitment protocols have been introduced. All of them rely on cyrptographic assumptions, like the existence of one-way functions.

Quantum computing, given the fact that observations can possibly alter the value of data, seems like a promising approach.

Quantum Bit Commitment

Suppose Alice and Bob wants to commit on the value of a single bit, chosen by Alice.

Alice chooses a value b, and prepares n qubits $|x_1\rangle, ..., |x_n\rangle$ as follows, after having drawn n bits at random, obtaining values $v_1, ..., v_n \in \{0, 1\}$:

- If b=1, then $|x_i\rangle$ will be set to $|v_i\rangle$.
- If b = 0, then $|x_i\rangle$ will be set to $H(|v_i\rangle)$.

She then noted the values $v_1..., v_n$, and send $|x_1\rangle, ..., |x_n\rangle$ to Bob, who keeps them in a safe place.

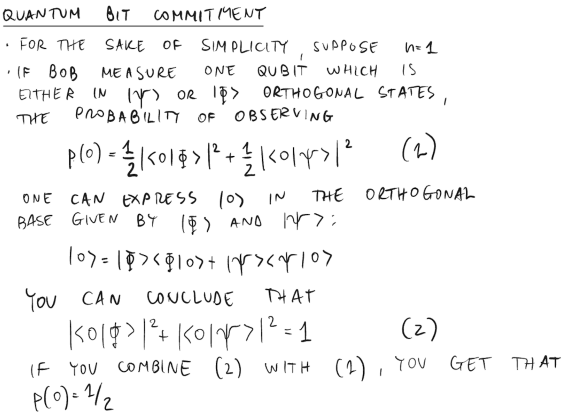
When the time comes for Alice to reveal the value of b, she tells Bob not only b, but also $v_1, ..., v_n$. Bob can then verify whether Alice is cheating by just measuring $|x_1\rangle, ..., |x_n\rangle$.

On the Security of the Quantum Bit Commitment

One can show that **no information** Bob can extract from the collection of qubits Alice sends him can distinguish between the case b = 0 and the case b = 1. The protocol, however, **cannot be considered secure**:

- Instead of preparing $|x_1\rangle, ..., |x_n\rangle$ as prescribed by the protocol, Alice could have prepared n pairs of entangled qubits, each pair being a Bell state, and send the first qubit of each pair to Bob.
- The qubits Bob receives do not have states of their own, being entangled with the qubits Alice keeps for herself.
- Before revealing her choice, Alice makes a direct measurement on each of the qubits she has kept and correctly informs Bob. Crucially, however, **she can change the value of b** by applying the H gate to its qubit.

There is more: as proved by Mayers, unconditionally secure quantum bit commitment is impossible.



Modulo 2 - Parte 2

Error Correction - Part I

Errors in Classical Computation and Communication

The possibility of observing **errors** is present both in realm of computation and communication devices, already in classical computing.

- Logical gates can indeed behave in a way which is different from their specification, but the possibility that this happens is so low that it can be taken to be 0.
- It can well be, instead, that long-distance communication channels are subject to errors.

One can deal with errors occurring along communication channels in a variety of ways, as studied in **information theory**

- For example, one can encode every logical bit to be sent as three instances of the same bit:

$$0 \to 000 \qquad 1 \to 111$$

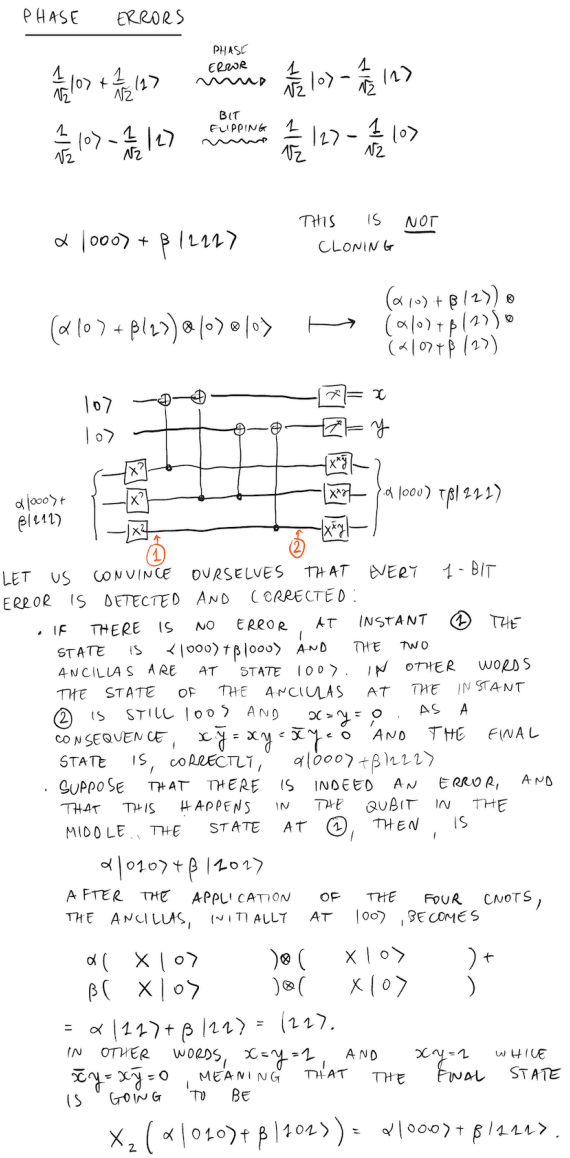
- This way, if errors affect either 0 or 1 of the bits in the triple, errors can be not only recognized, but also corrected.
- This is the so-called **repetition code.**

Errors in Quantum Computating

When we switch from classical to quantum communication/computation, a number of additional problems show up:

- Already at the level of quantum **computation**, error correction is absolutely crucial, because qubits are implemented by individual atomic-scale physical systems.
- Quantum error correction is problematic, given the very nature of quantum computing, because the typical way of detecting an error in any piece of data consists in observing it, but doing this would **disturb** the underlying quantum state.
- The kind of errors one can observe are more general: not only the state of a qubit can be flipped, but there could be, e.g. phase errors.

Despite all these additional problem, quantum error correction is indeed possible, and there is a whole research field whose objective is precisely that of designing quantum **error correcting codes** (QECCs for short).



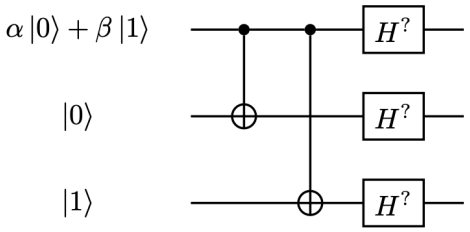
QECC in a Simplified Setting - Part II

Quantum Repetition Code?

One could try to replicate the idea behind classical repetition codes, and effectively replicate the value of a qubit to two other qubits, by way of controlled-not gates.

For simplicity, we can assume that the only kind of errors we are interested in correcting are **single qubit flipping errors**.

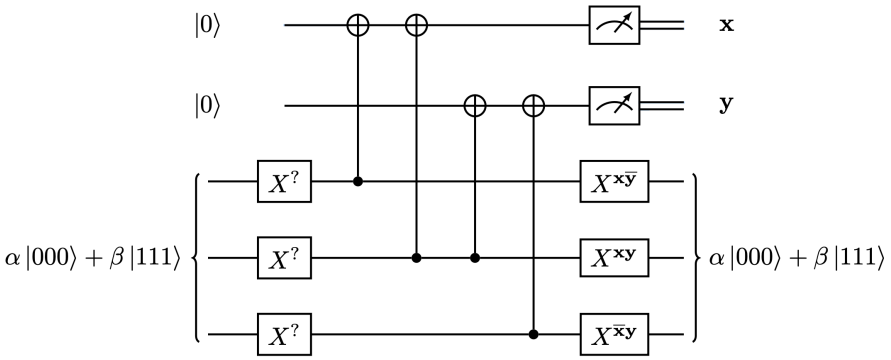
In other words, we are in the following situation:



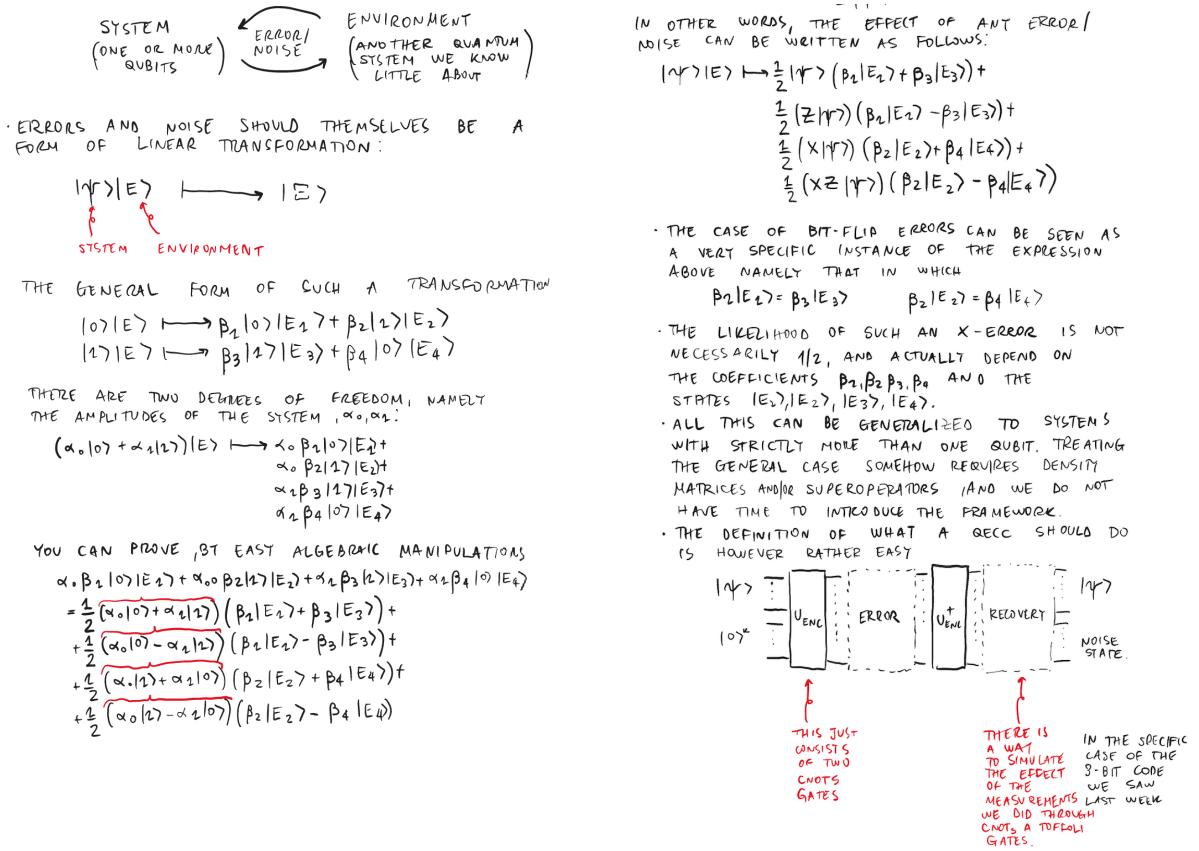
How can we **detect** and **correct** errors, then?

> 💡 Rotation trick:
>
> Setting a imaginary number for the angle rotation, this will allow you to have a rotation minor then the error.

A Simple QECC



General Error Model



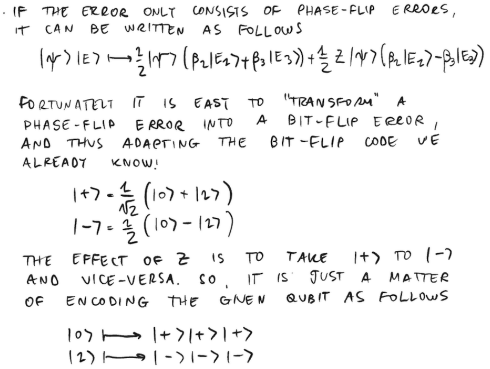
decomposizione generale di Pauli per gli errori quantistici

$$|\psi\rangle|E\rangle \to 1/2|\psi\rangle(\beta_1|E_1\rangle + \beta_3|E_3\rangle) + 1/2(Z|\psi\rangle)(\beta_1|E_1\rangle - \beta_3|E_3\rangle) + 1/2(X|\psi\rangle)(\beta_2|E_2\rangle + \beta_4|E_4\rangle) + 1/2(XZ|\psi\rangle)(\beta_2|E_2\rangle - \beta_4|E_4\rangle)$$

Dove:

- **Termine 1**: Identità (nessun errore)
- **Termine 2**: Errore di fase Z
- **Termine 3**: Errore di bit-flip X
- **Termine 4**: Errore combinato XZ = -iY

Dealing with Phase-Flip errors

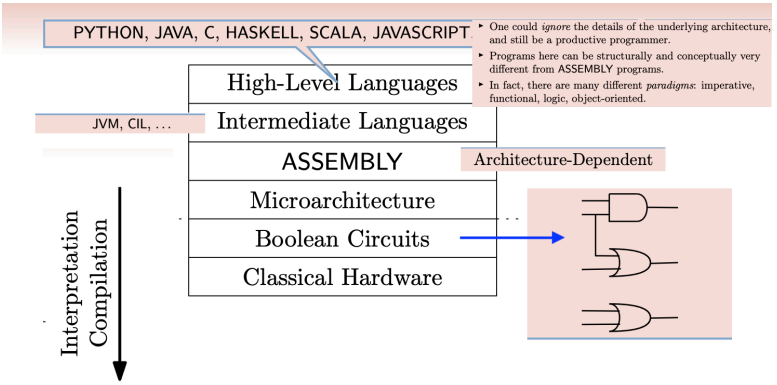


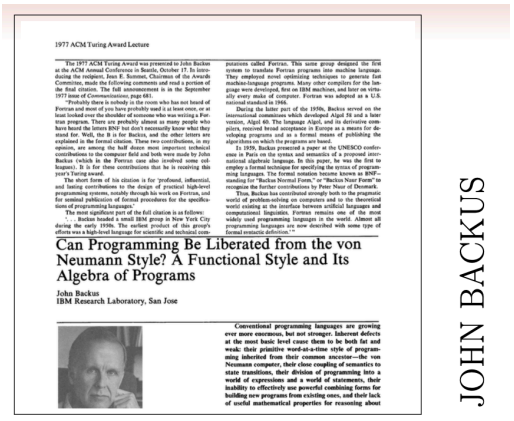
Il prof ha mancato un $\sqrt{2}$ quindi noi mappiamo cosi i i bit:

$$|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$
$$|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

Modulo 2 - Parte 3

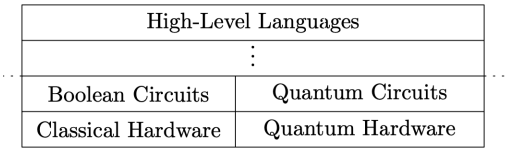
Quantum Programming Languages - Part I

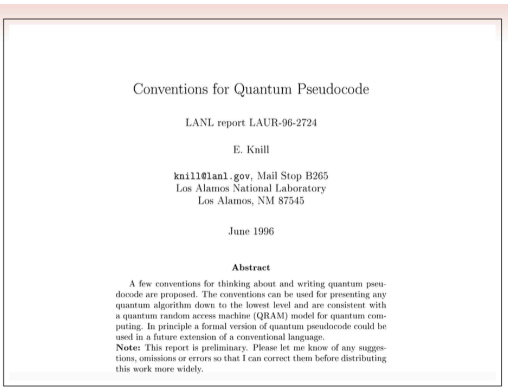




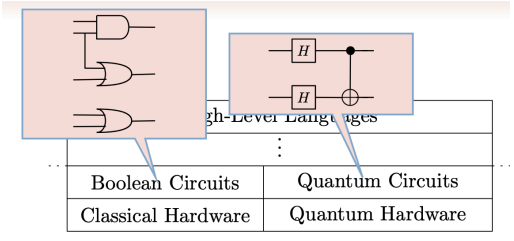
JOHN BACKUS

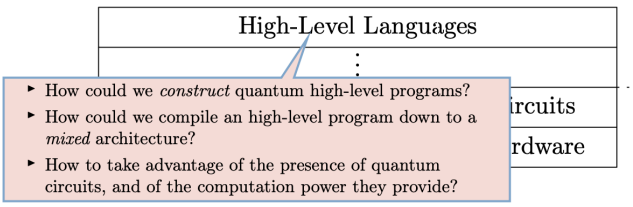
ABSTRACT MACHINES



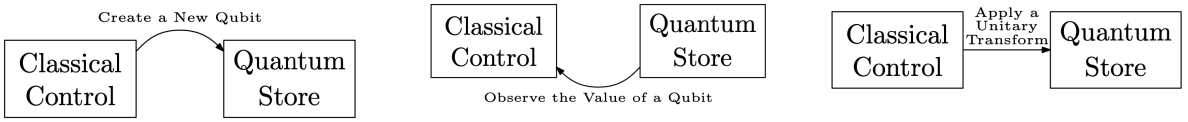


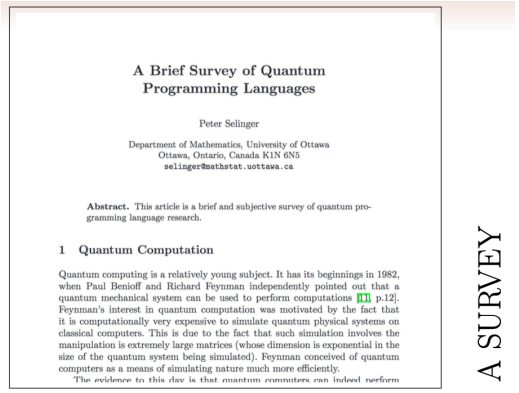
THE QRAM MODEL





Quantum Data and Classical Control





A Brief Survey of Quantum Programming Languages

Peter Selinger

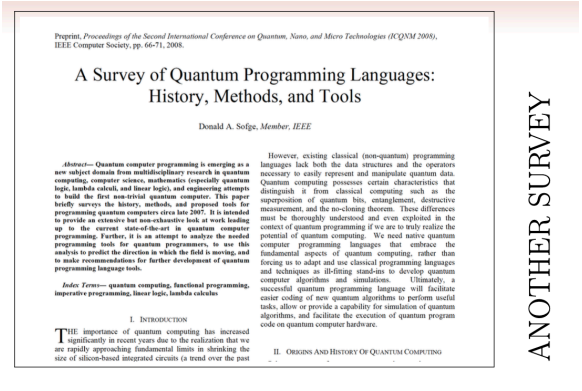
Department of Mathematics, University of Ottawa
Ottawa, Ontario, Canada K1N 6N5
selinger@mathstat.uottawa.ca

Abstract. This article is a brief and subjective survey of quantum programming language research.

1 Quantum Computation

Quantum computing is a relatively young subject. It has its beginnings in 1982, when Paul Benioff and Richard Feynman independently pointed out that a quantum mechanical system can be used to perform computations [11, p.12]. Feynman's interest in quantum computation was motivated by the fact that it is computationally very expensive to simulate quantum physical systems on classical computers. This is due to the fact that such simulation involves the manipulation is extremely large matrices (whose dimension is exponential in the size of the quantum system being simulated). Feynman conceived of quantum computers as a means of simulating nature much more efficiently.

A SURVEY



Preprint, *Proceedings of the Second International Conference on Quantum, Nano, and Micro Technologies (ICQNM 2008)*, IEEE Computer Society, pp. 66-71, 2008.

A Survey of Quantum Programming Languages: History, Methods, and Tools

Donald A. Sofge, *Member, IEEE*

Abstract— **Quantum computer programming is emerging as a new subject domain from multidisciplinary research in quantum computing, computer science, mathematics (especially quantum logic, lambda calculi, and linear logic), and engineering attempts to build the first non-trivial quantum computer. This paper briefly surveys the history, methods, and proposed tools for programming quantum computers circa late 2007. It is intended to provide an extensive but non-exhaustive look at work leading up to the current state-of-the-art in quantum computer programming. Further, it is an attempt to analyze the needed programming tools for quantum programmers, to use this analysis to predict the direction in which the field is moving, and to make recommendations for further development of quantum programming language tools.**

Index Terms— **quantum computing, functional programming, imperative programming, linear logic, lambda calculus**

I. Introduction

THE importance of quantum computing has increased significantly in recent years due to the realization that we are rapidly approaching fundamental limits in shrinking the size of silicon-based integrated circuits (a trend over the past

However, existing classical (non-quantum) programming languages lack both the data structures and the operators necessary to easily represent and manipulate quantum data. Quantum computing possesses certain characteristics that distinguish it from classical computing such as the superposition of quantum bits, entanglement, destructive measurement, and the no-cloning theorem. These differences must be thoroughly understood and even exploited in the context of quantum programming if we are to truly realize the potential of quantum computing. We need native quantum computer programming languages that embrace the fundamental aspects of quantum computing, rather than forcing us to adapt and use classical programming languages and techniques as ill-fitting stand-ins to develop quantum computer algorithms and simulations. Ultimately, a successful quantum programming language will facilitate easier coding of new quantum algorithms to perform useful tasks, allow or provide a capability for simulation of quantum algorithms, and facilitate the execution of quantum program code on quantum computer hardware.

II. Origins And History Of Quantum Computing

ANOTHER SURVEY

A Multitude of Idioms

- Functional Languages
- Imperative Languages
- Logic Programming Languages
- Quantum Constraint Programming Languages
- ...

Quantum Circuit Description Languages

Over the past decade, we have witnessed the introduction (and use) of programming languages in which programs are not designed to execute (quantum) instructions, but to rather build quantum circuits, to be then executed by quantum hardware architectures or simulators.

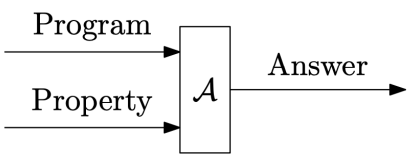
It is clear that in this way any classical programming language can become a quantum language, simply through the implementation of libraries designed for the manipulation of circuits.

- This is the case of Qiskit, Cirq, Quipper, . . .

Quantum Program and System Verification - Part II

Classic Program Verification

In classic program verification, one is interested in checking whether a program satisfies a property:



where:

- The property specifies the expected behaviour of the program, in the form of *functional*

> 💡 *functional:* Whether the program guarantees a certain “quality of service”, or not.

or *non-functional*

> 💡 *non-functional:* Whether the program does what it is supposed to do (or not).

specifications.

- The answer, typically, is either YES, NO, or UNKNOWN.
- A can be an algorithm, or some kind of analytic methodology.

> 💡 *analytic methodology:* A precise analysis is in most cases impractical

This can be done in high-level languages only if the property is preserved through compilation or interpretation.

- Most functional properties are preserved.
- Many non-functional properties (e.g. performances) requires more care.

Examples of Program Verification Techniques

- Program Logics
  - Very popular for imperative programs.
  - One prove, deductively, the validty of triples in the form $\vdash \{\Phi\}P\{\Psi\}$, meaning that the program P when run on an input state satisfying Φ, ends up in a state satisfying Ψ.
  - Particularly suited for proving functional properties, but adapted to non-functional ones.
- Model Checking
  - A system or program S is seen as a structure interpreting a formula A written in modal logics, e.g., temporal logics.
  - One then checks whether $S \models A$ in an exhaustive way, i.e., by checking in which states of S the formula A (and its subformulas) hold, and then checking whether the initial states of S are among those.
- Type Systems
  - Supported by most functional programming languages
  - one derives judgments in the form $\Gamma \vdash P : A$ where A is a type and Γ attributes a type to each of the variables which P refers to.
  - Originally meant to guarantee safety (“Well-typed programs cannot go wrong”), but then generalized to other functional and non-functional properties

Quantum Program Verification: Challenges

Verifying quantum programs and system can be rewarding:

- Testing them can be very expensive!
- Certain properties are arguably of paramount importance, like resource consumption.

On the other hand, verifying quantum programs against functional or nonfunctional properties is challenging for a number of reasons:

- The underlying computational model is simply different, and more complicated.
- Besides the usual exponential blowup (there are $\Theta(2^n)$ states of size n), there is another one coming from superposition.
- Most quantum algorithms are only approximately correct.

Laboratorio 1

Extend to n qubit GHZ state

Let's create a more extended version of the previous circuit. Create a function that takes in input the number of qubits *n* and returns the circuit that creates a GHZ state (''Extension'' of one of the bell states for more qubits) so that the final state of the circuit is:

$|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle^{\otimes n} + \frac{1}{\sqrt{2}}|1\rangle^{\otimes n}$

Fixed Gates and Parametric gates

Up until now we only utilized fixed gates. These gates are the most famous and are used in basically every quantum algorithms, some example include:

- Pauli gates: *X, Y, Z, I*
- Phase shift gates: *S, T*
- Controlled (fixed) gates: *CX, CZ, CCX, ...*

Another type of gate that are usually found in quantum circuits are parametric gates that rotate the qubit by an angle specified at *calling time*, giving more freedom in the creation of the circuit.

These gates are usually the generalizzation of Pauli gates $RX(\theta)$,$RY(\theta)$,$RZ(\theta)$ (controlled and not) but assume the most general form in what is called the U3 gate:

$$U3(\theta, \phi, \lambda) = \begin{pmatrix} cos(\frac{\theta}{2}) & -e^{i\phi}sin(\frac{\theta}{2}) \\ e^{i\lambda}sin(\frac{\theta}{2}) & e^{i(\phi+\lambda)}cos(\frac{\theta}{2}) \end{pmatrix}$$

This gate, depending on the parameter θ,ϕ,λ can become any other single qubit gate:

$$U3(\theta, -\frac{\pi}{2}, \frac{\pi}{2}) = RX(\theta)$$
$$U3(\theta, 0, 0) = RY(\theta)$$

Data encoding circuits

Data-encoding circuits These parameterized circuits encode data onto quantum states for processing by quantum machine learning algorithms. Some circuits supported by Qiskit are:

- **Amplitude** encoding, which encodes each number into the amplitude of a basis state. This can store 2^n numbers in a single state, but can be costly to implement.
- **Basis** encoding, which encodes an integer by preparing the corresponding basis state $|k\rangle$
- **Angle** encoding, which sets each number in the data as a rotation angle in a parameterized circuit.

The best approach depends upon the specifics of your application. On current quantum computers, however, we often use angle-encoding circuits such as the ZZFeatureMap.

Laboratorio 2

Shor's algorithm

In Shor's algorithm, the quantum part finds the period of a function of the form:

$f(x)=a^x mod N$

Where:

- N is the integer we want to factor.
- a is a random number such that 1<a<N and gcd(a,N)=1

Finding the period r of this function is the key to discovering a nontrivial factor of N with high probability.

Recap

Shor’s algorithm is designed to factor a composite number N efficiently using a quantum computer. The classical part selects a random base a, checks if it shares a common factor with N, and if not, proceeds to find the period r of the function:

$f(x)=a^x mod N$

The quantum subroutine finds the period r, and then classical post-processing uses r to derive a nontrivial factor of N.

Steps

1. Pick a random integer such that 1<a<N
2. Compute gcd(a,N). If it's not 1, we already found a factor.
3. Use quantum period finding to determine r, the period of $f(x)=a^x \mod N$
4. If r is even and $a^{r/2} \neq -1 \ mod \ N$, then

   $gcd(a^{r/2}+1, N)$

   yields a nontrivial factor of N

In practice

1. We create a function that can compute $a^x \mod N$ where we iterate trough different exponents.
2. we create a 4 + 4 qubit circuit where the first four are the ''input'' qubits and the second 4 are the anchillas
3. we create a full superposition on the input register by applyng H gates
4. We pose the second register to 1 (for this implementation we are computing $|x\rangle \otimes |1\rangle \rightarrow |x\rangle \otimes |a^x \mod N\rangle$)

In general we divide the algorithm in 2 parts:

- quantum circuit called quantum phase estimation.
- classical post processing.

The phase estimation is composed by the state preparation, modular exponentiation, Inverse Quantum Furier transform and measurement. Classical post processing extract and checks if the measurement contains the correct solution

Grover's algorithm

Another pillar of quantum computing that improves on classical performance **(quadratic improvement)** dvided into 4 steps, it is used to search for a marked element in a unsorted database. In our example we will search for the string 0101 accross the Database $D=\{0,1\}^{\otimes 4}$.

Note that the database has $|D|=2^n=N$ elelments with $n=4$

The algorithm

1. **State preparation**: We prepare a register of qubit in complete superposition by applying H gates to all n qubits in the state $|0\rangle$

$$|\psi_1\rangle = H^{\otimes n}|0\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle$$

2. **Oracle**: The oracle O_f is an operation that acts only on the marked datapoint im by flipping the sign of the amplitude:

$$|\psi_2\rangle = O_f|\psi_1\rangle = \frac{1}{\sqrt{N}} \sum_{i=0, i\neq i_m}^{N-1} |i\rangle - \frac{1}{\sqrt{N}}|i_m\rangle$$

3. **Diffusion operator**: we than apply the operator D that reflects the state about the average amplitude:

$$D = 2|\psi_0\rangle\langle\psi_0| - I$$

4. **Repeat**: we repeat the process $O_f D$ as many time as we need. Mathematically the probability of finding the right solution is enough after r repetiotions defined as:

$$r \approx \left\lfloor \frac{\pi}{4}\sqrt{N} \right\rfloor$$

Usefull link

https://medium.com/@marcell.ujlaki/exploring-quantum-computing-demystifying-quantum-fourier-transformations-unveiling-the-math-with-5d74f3f8025f

https://medium.com/mit-6-s089-intro-to-quantum-computing/bqnp-the-quantum-analogue-of-np-486ed2469c1d

https://medium.com/@marcell.ujlaki/exploring-quantum-computing-computational-complexity-theory-and-complexity-classes-3233adf8337a

https://pennylane.ai/qml/demos/tutorial_qpe

https://github.com/Qiskit/textbook/tree/main/notebooks/ch-algorithms

https://algassert.com/post/1718

https://medium.com/@olgaokrut/an-example-of-shors-factoring-algorithm-d80e9b8d6ed0

https://www.qutube.nl/quantum-algorithms/application-of-a-quantum-computer

https://www.iac.rm.cnr.it/ciclo-di-lezioni-dedicate-al-quantum-computing

Esame