Installing Qiskit

- https://qiskit.org/documentation/install.html

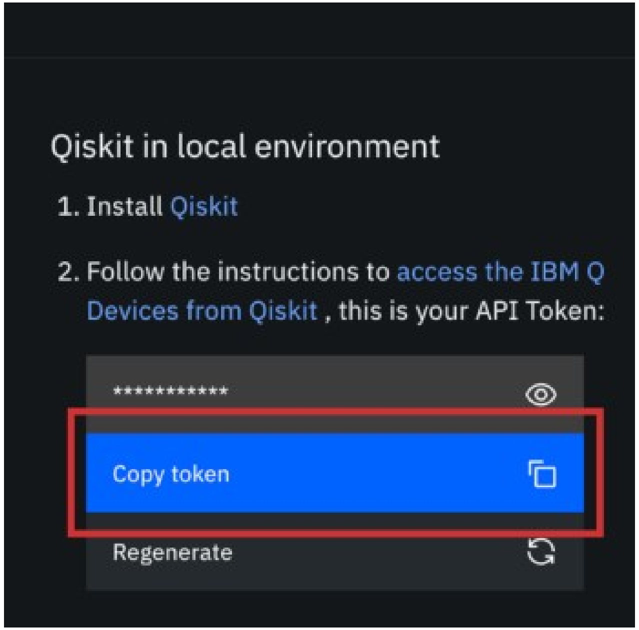
- Install Anaconda (Python distribution)

- pip install qiskit[visualization]

- Create a free IBM Quantum Experience account

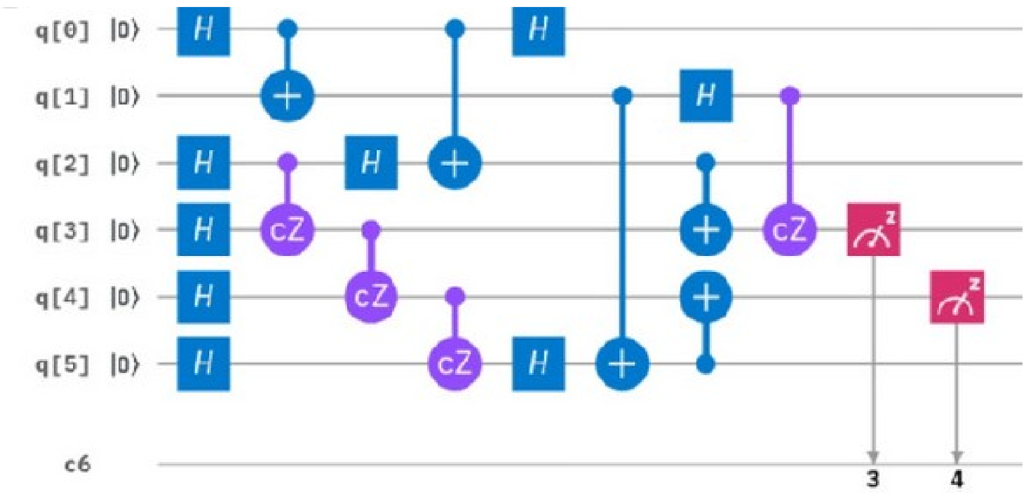
- from qiskit import IBMQ
  IBMQ.save_account('MY_API_TOKEN')

- Download/Run Jupyter notebook of a Qiskit tutorial



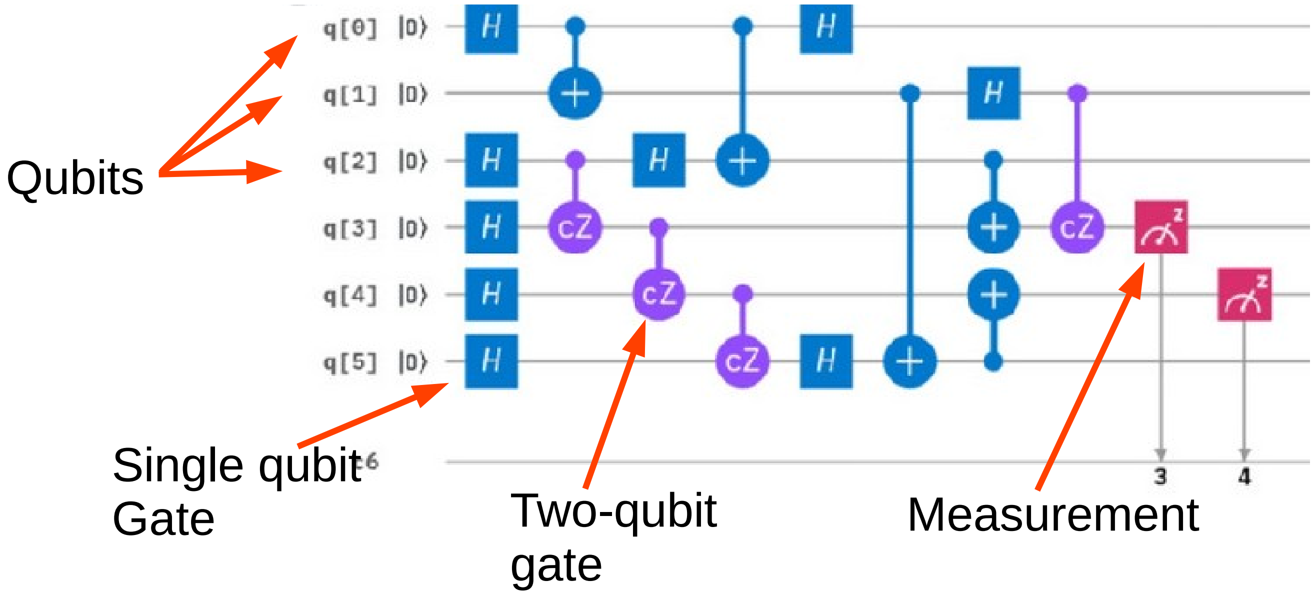
Lecture 2

Quantum algorithms in the quantum circuit model



benoit.vermersch@lpmmc.cnrs.fr

Reminder : A quantum circuit



Goal 1: Having algorithms that are faster (less operations) than classical algorithms

Goal 2: Having algorithms that are protected against errors (Lecture 3)

Warm-up : Deutsch's algorithm

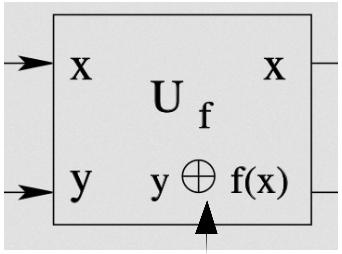
Problem: Given binary function f : [0,1]→ [0,1]. Is f(0)=f(1)?

Classical solution:

Two iterations needed (Iteration 1, I measure f(0). Iteration 2, I measure f(1))

Quantum solution: we will test the two input states simultaneously

Function f implemented via a two-qubit 'quantum oracle'

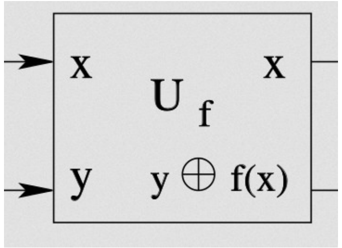


ie y is flipped iif f(x) is 1

XOR (sum mod 2)

Warm-up : Deutsch's algorithm

Remark: How to implement a quantum oracle?

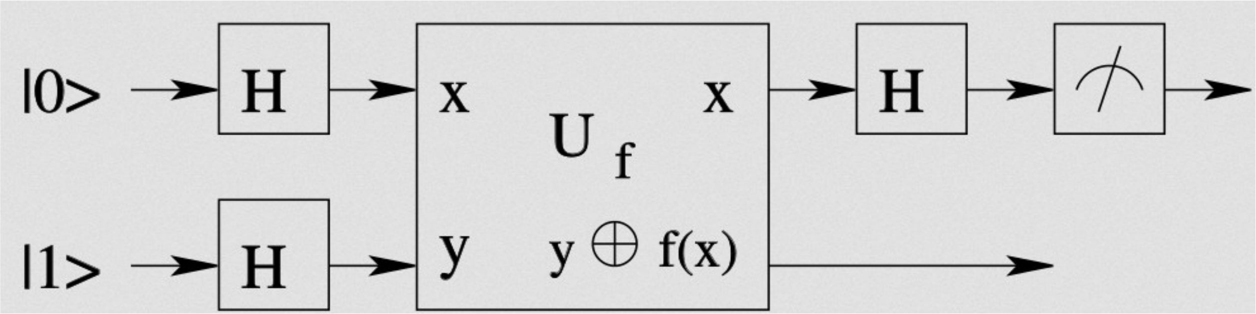


- In **quantum query complexity**, one assumes the oracle given and counts the number of oracle queries to define the complexity

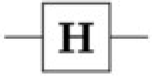
- In practice, if the function can be computed classically via a reversible circuit, we can map the circuit to a quantum circuit, using the technique of `uncomputation'.

- In the rest of this lecture, we won't bother anymore about oracles. However, this does not mean this questions is not important.

Warm-up : Deutsch’s algorithm



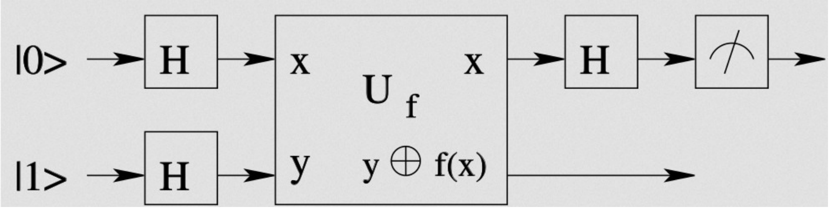
Hadamard (H)



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

What do I measure for f(0)=f(1), for f(0) != f(1) ? (using a single measurement!)

Warm-up : Deutsch's algorithm



$$|\psi\rangle = (|0\rangle + |1\rangle)(|0\rangle - |1\rangle)$$

$$|\psi\rangle' = |0, 0 \oplus f(0)\rangle - |0, 1 \oplus f(0)\rangle + |1, 0 \oplus f(1)\rangle - |1, 1 \oplus f(1)\rangle$$

If $f(0) = f(1)$, let $0 \oplus f(0) = 0 \oplus f(1) = a$, $1 \oplus f(0) = 1 \oplus f(1) = b$

$$|\psi\rangle' = (|0\rangle + |1\rangle)(|a\rangle - |b\rangle)$$

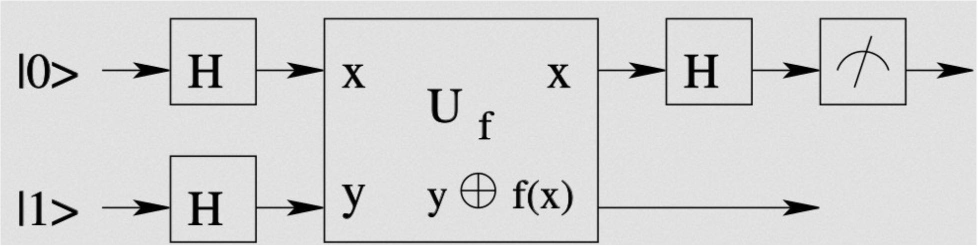
Else, if $f(0) \neq f(1)$, let $0 \oplus f(0) = 1 \oplus f(1) = a$, $1 \oplus f(0) = 0 \oplus f(1) = b$

$$|\psi\rangle' = (|0\rangle - |1\rangle)(|a\rangle - |b\rangle)$$

After the last Hadamard,

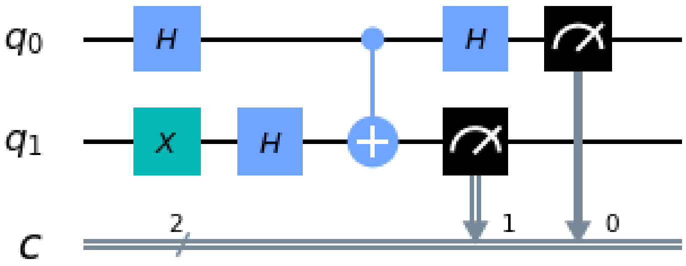
$$\begin{aligned} |\psi\rangle' &= |0\rangle (|a\rangle - |b\rangle) \quad , f(0) = f(1) \\ |\psi\rangle' &= |1\rangle (|a\rangle - |b\rangle) \quad , f(0) \neq f(1) \end{aligned}$$

Warm-up : Deutsch's algorithm

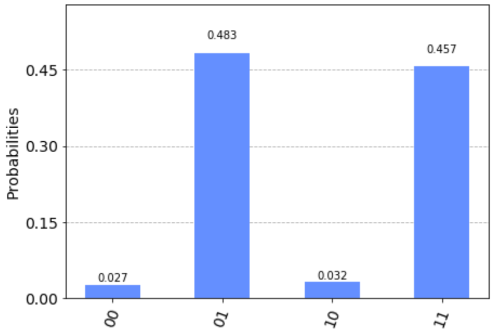


Implementation with IBM Qiskit

Suppose f(x)=x. Then the oracle becomes a CNOT gate.

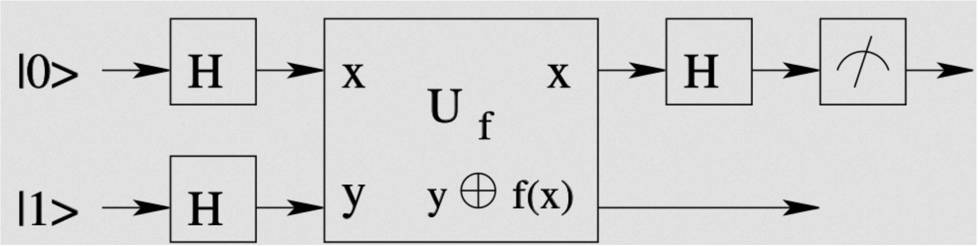


Demo with IBMQ



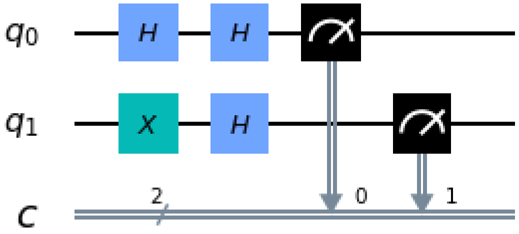
Up to errors, the first qubit ends up in |1> !

Warm-up : Deutsch's algorithm

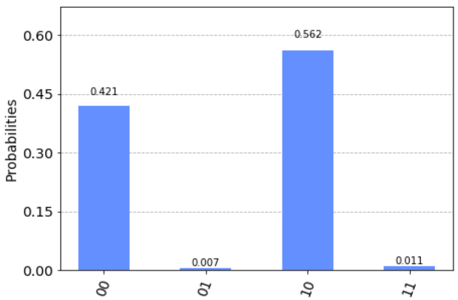


Implementation with IBM Qiskit

Suppose f(x)=0. Then the oracle becomes the identity



Demo with IBMQ



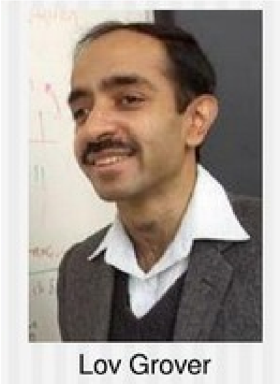
Up to errors, the first qubit ends up in |0> !

Warm-up : Deutsch’s algorithm

Conclusion : First algorithm that outperfoms classical algorithms using quantum parallelism.

Generalizes to n qubits : Deutsch-Josza algorithm

Grover’s algorithm (1996)



Problem (Data search): Given binay function with f(w)=1 for a single n-bit string w
($N=2^n$ is the number of configurations), find w

Application: Database search (applications: SAT problems (circuit design, automatic theorem proving, etc..)

Grover's algorithm (1996)

Classical solution : $O(N=2^n)$ function evaluation

Quantum Grover's algorithm : Simultaneous testing via quantum parallelism

Grover's algorithm (1996)

Grover's oracle : $U_f = I - 2\,|w\rangle\,\langle w|$

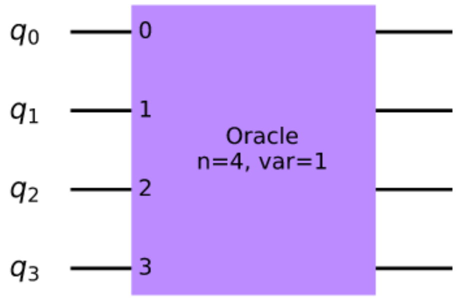
The oracle 'marks' the solution:

$$U_f\,|x \neq w\rangle = |x\rangle$$

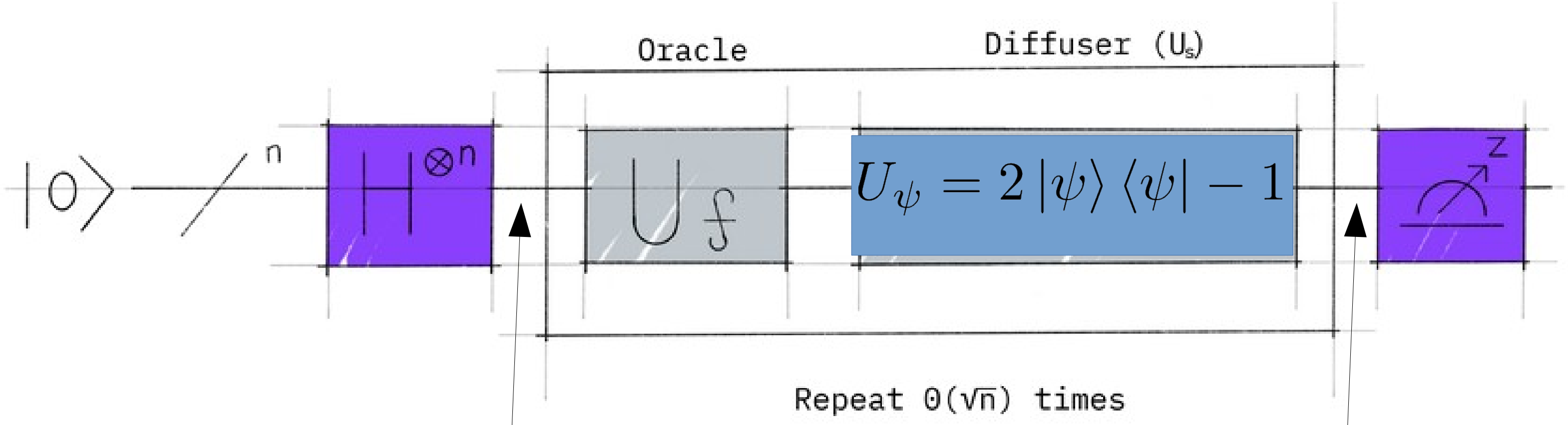
$$U_f\,|w\rangle = -\,|w\rangle$$

Ex: Qiskit's implementation
(the details are not our concern for an oracle..)

```
n = 4
oracle = grover_problem_oracle(n, variant=1)
```



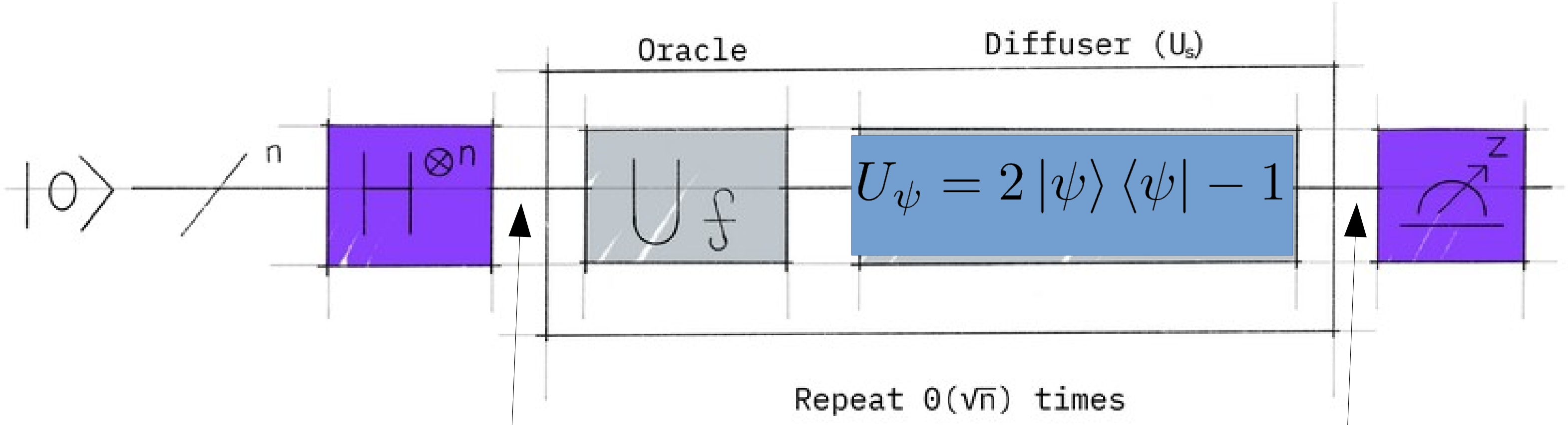
Grover’s algorithm (1996)



$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_x |x\rangle$$

$$|\psi_t\rangle = (U_\psi U_f)^t\,|\psi\rangle$$

Grover’s algorithm (1996)



$$\left|\psi\right\rangle = \frac{1}{\sqrt{N}}\sum_x \left|x\right\rangle$$

$$\left|\psi_t\right\rangle = (U_\psi U_f)^t \left|\psi\right\rangle$$

Grover's algorithm (1996)

After the first Hadamards:

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_x |x\rangle = \underbrace{\sqrt{\frac{N-1}{N}}}_{\cos(\theta/2)} |\alpha\rangle + \underbrace{\sqrt{\frac{1}{N}}}_{\sin(\theta/2)} |w\rangle$$

$$|\alpha\rangle = \frac{1}{\sqrt{N-1}} \sum_{x \neq \omega} |x\rangle$$

Oracle

$$U_f = I - 2\,|w\rangle\langle w|$$

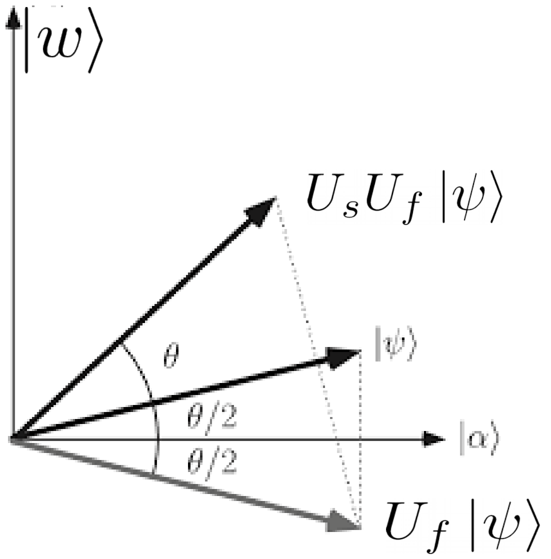
$$U_f\,|\psi\rangle = \cos(\theta/2)\,|\alpha\rangle - \sin(\theta/2)\,|w\rangle$$

Reflection versus $|\alpha\rangle$

Diffuser

$$U_\psi = 2\,|\psi\rangle\langle\psi| - 1$$

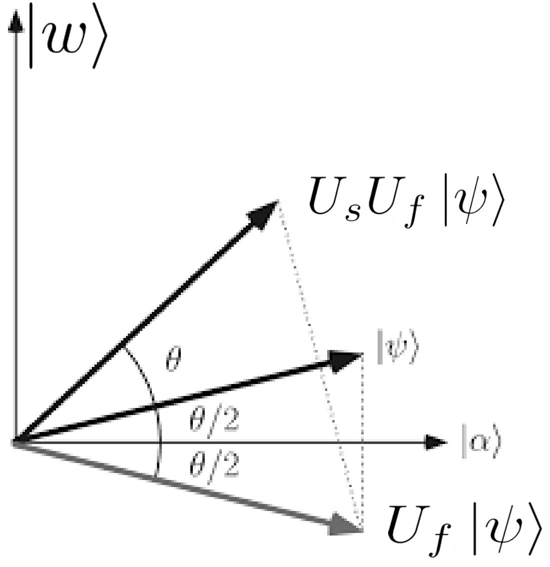
Reflection versus $|\psi\rangle$



Grover’s algorithm (1996)

After one Grover iteration (c.f TD)

$$|\psi_1\rangle = U_s U_f |\psi\rangle = \cos(3\theta/2)|\alpha\rangle + \sin(3\theta/2)|w\rangle$$

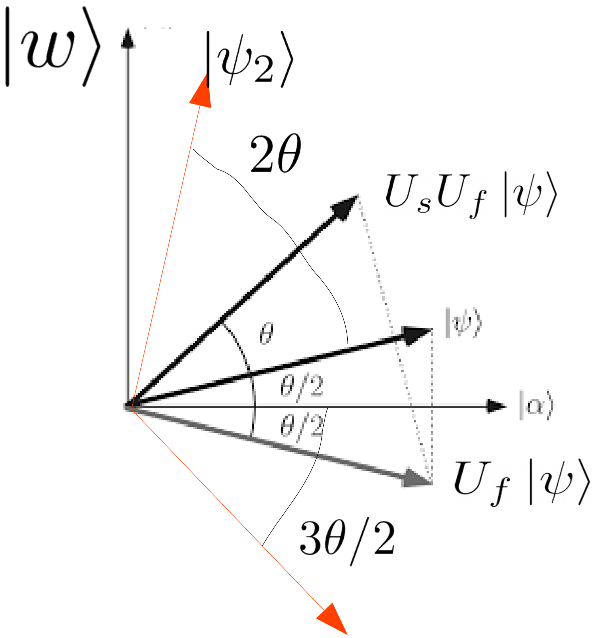


The algorithm brings the quantum state towards the solution w

Grover's algorithm (1996)

Performance

After t iterations
$$|\psi_t\rangle = (U_\psi U_f)^t |\psi\rangle = \cos[(2t+1)\theta/2]\,|\alpha\rangle + \sin[(2t+1)\theta/2]\,|w\rangle$$



Solution obtained for:

$$\theta t \approx \pi/2 \longrightarrow t \approx (\pi/4)\sqrt{N}$$

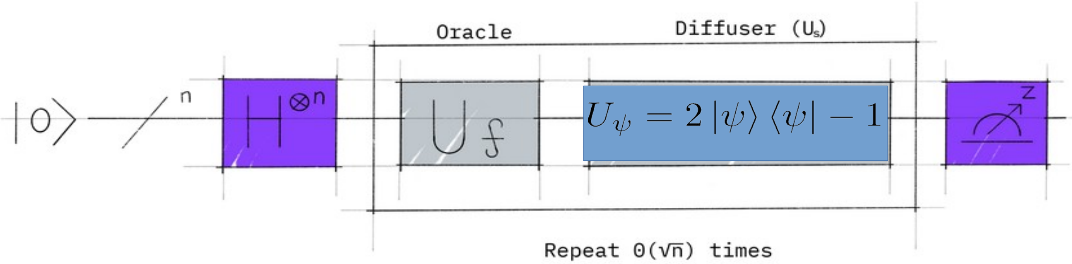
Quadratic speedup !

Ex : 128-bit key 2^{64} iterations instead of 2^{128}

Note: for multiple targets → $t \approx (\pi/4)\sqrt{N/k}$

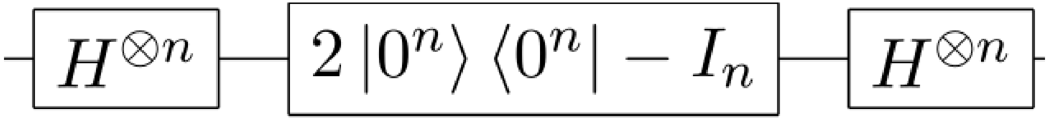
Grover's algorithm (1996)

Implementation



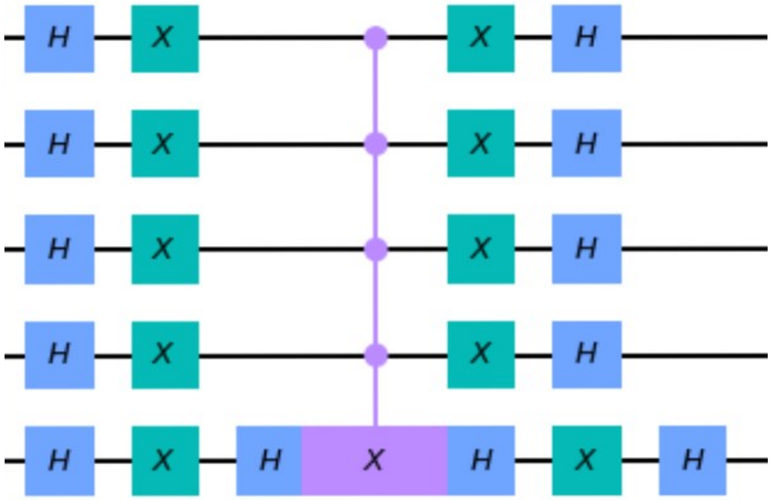
Efficient algorithm for Grover's diffuser (cf TD)

$$U_\psi = 2\,|\psi\rangle\langle\psi| - 1 \qquad |\psi\rangle = H^{\otimes n}\,|000\ldots0\rangle$$



Grover's algorithm (1996)

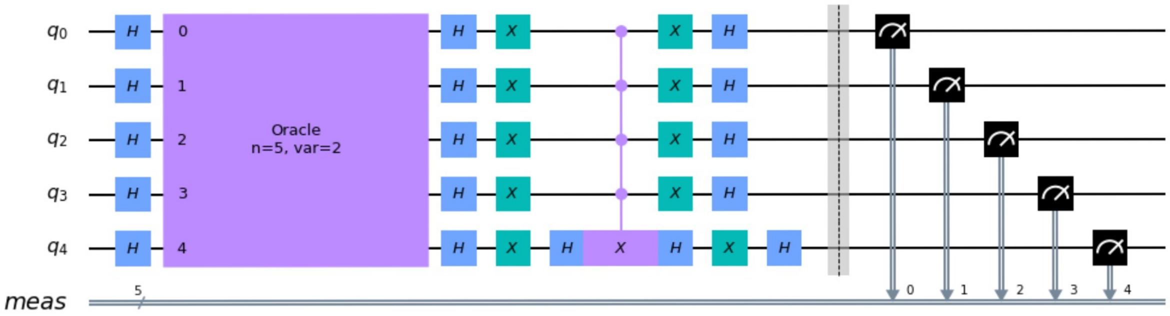
$$- H^{\otimes n} - 2\left|0^n\right\rangle\left\langle 0^n\right| - I_n - H^{\otimes n} -$$

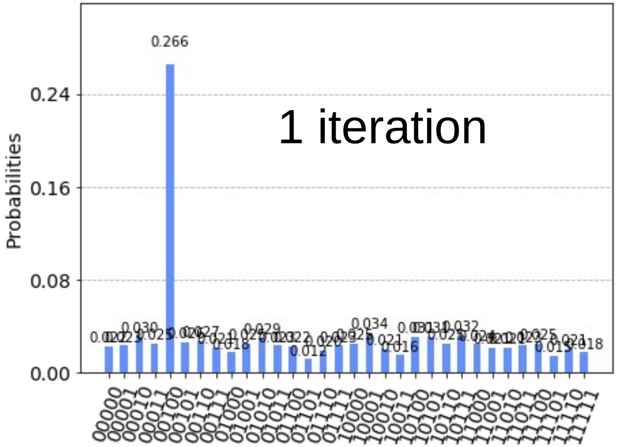


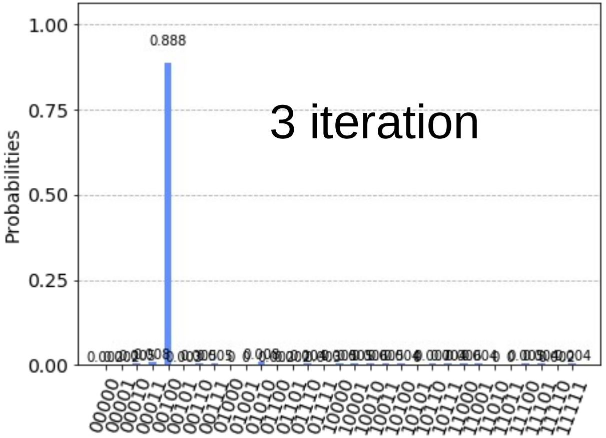
N qubit Toffoli gate

Grover’s algorithm (1996)

Illustration with Qiskit’s Aer simulator







Grover's algorithm (1996)

Remark: Is the complexity of Grover's algorithm a good news for quantum computers?

If an `improved Grover's' algorithm would provide **an exponential speedup** (I.e scaling polynomially with the number of qubits),

Then I could solve any NP problem in polynomial time on a quantum machine!!

- Take a NP problems with 2^n possible solutions
- Each solution can be tested in polynomial time (NP property) → I can define an oracle function f
- Use the oracle in Grover's algorithm → I could find the solution in polynomial time

Unfortunately, the current Grover's algorithm with only quadratic speedup has been shown to be optimal. $\sqrt{N = 2^n}$

Is there an algorithm that can solve a specific NP problem in polynomial time?

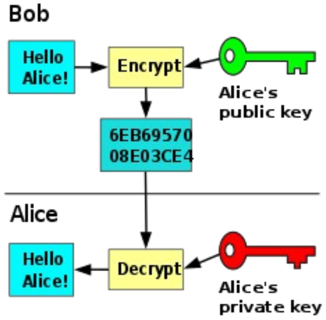
Shor's algorithm (1995)

Factorization Problem of a number N

Classical algorithm: (sub-)exponential in n (number of bits to represent n)

Quantum algorithm: polynomial in n: **Exponential speedup**

→ A potential threat to RSA cryptography...



Shor’s algorithm (1995)

Prerequisites from arithmetic:

If N, product of two coprimes, divides b^2-1 ,
Then gcd(N,b-1) and gcd(N,b+1) are non-trivial factors of N

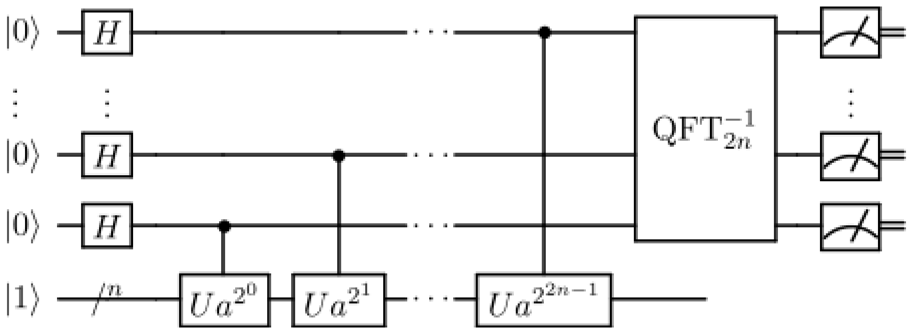
Proof: see e.g., Nielsen and Chuang

***Example*:** N=91. For b=64. N divides b^2-1=4095.
Therefore, gcd(91,63)=7 and gcd(91,65)=13 divide 91

Algorithm

- Take a random in [1,N]
- Find r such that **a^r=1 mod (N)** by finding the period of f(x) = a^x mod(N)
  Then N divides a^r-1
- If r is even, b=$a^{r/2}$, and, N divides b^2-1

Shor’s algorithm (1995)



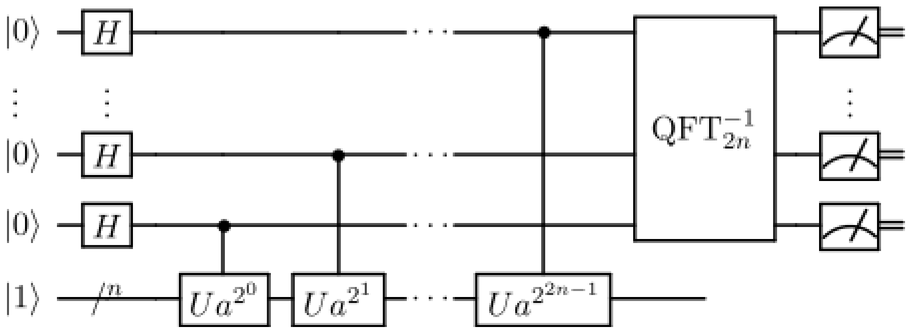
Quantum subroutine : find the period r of f(x) = a^x mod(N)

- Choose q so that $Q=2^q > N^2$ and consider a 2q qubit quantum computer (to provide sufficient spectral resolution in finding r)
- Prepare the first q qubits in a superposition state
- Apply **modular exponentiation**
- Apply **the inverse quantum Fourier transform** on the first q qubits

$$|x\rangle\,|1\rangle^{\otimes N} \rightarrow |x\rangle \otimes |a^x \mathrm{mod}(N)\rangle$$

$$|\psi\rangle = \frac{1}{Q}\sum_x \left(\sum_y e^{-2i\pi xy/Q}\,|y\rangle\right) \otimes |f(x)\rangle$$

Shor's algorithm (1995)



Measurement:

$$|\psi\rangle = \frac{1}{Q}\sum_x \Big(\sum_y e^{-2i\pi xy/Q}|y\rangle\Big)\otimes|f(x)\rangle \longrightarrow |\psi\rangle = \frac{1}{Q}\sum_y \left(|y\rangle \otimes \sum_x e^{-2i\pi xy/Q}|f(x)\rangle\right)$$

$$[f(x) = a^x \bmod(N)]$$

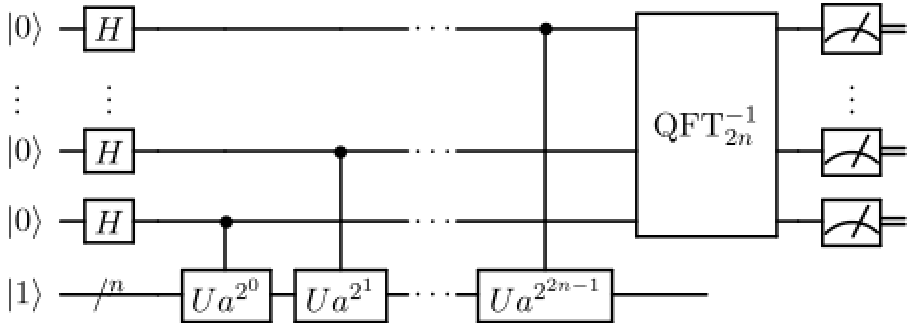
$$P(y) = \frac{1}{Q^2}\sum_{x,x'} e^{2i\pi(x'-x)y/Q}\left\langle f(x)\,\middle|\,f(x')\right\rangle \qquad P(y) \approx \sum_{n,x-x'=nr} e^{2i\pi nry/Q}$$

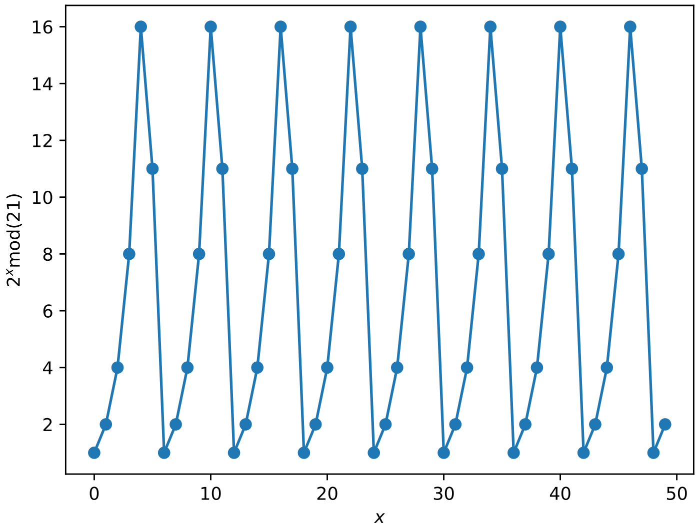
Maximum for yr/Q integer (as a constructive interference in optics)

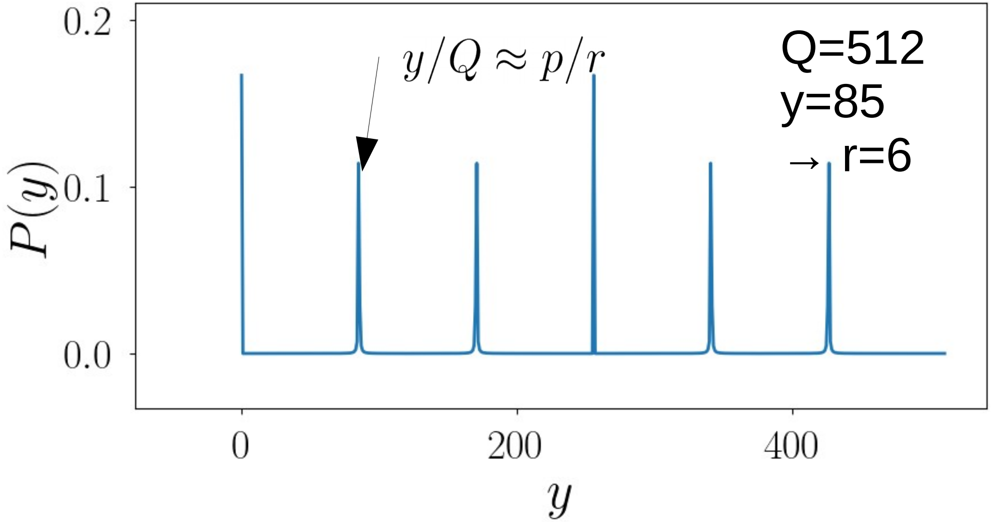
→ r can be extracted (via continued fraction algorithms, see Nielsen)

Shor's algorithm (1995)

Example: Factorizing 21 with a=2 (TD2)

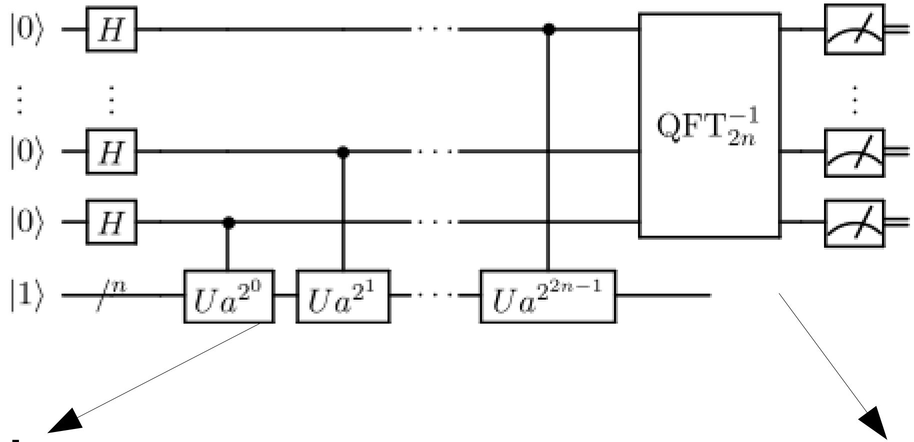






Shor's algorithm (1995)

Implementation aspects



Modular exponentiation
(multiplication in the ancilla space)

Cost $O(n^3)$

Quantum Fourier Transform

Cost $O(n^2)$ (see TD)

The practical implementation of Shor's algorithm is difficult: Many qubits and many gates..

nature physics

LETTERS

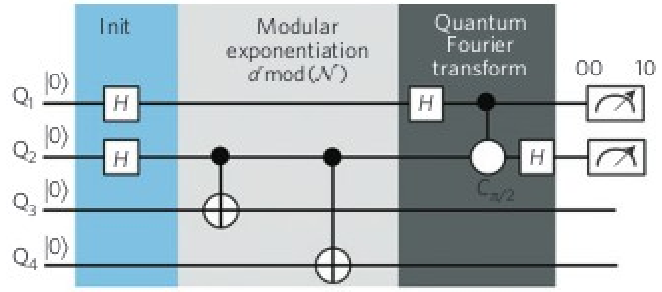
PUBLISHED ONLINE: 19 AUGUST 2012 | DOI: 10.1038/NPHYS2385

Computing prime factors with a Josephson phase qubit quantum processor

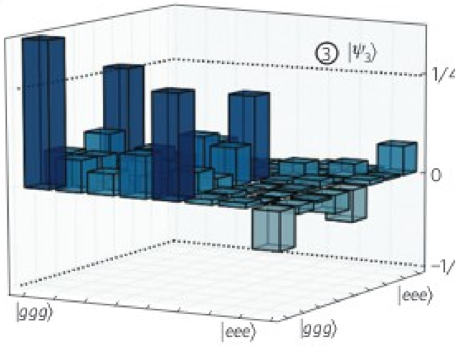
Erik Lucero, R. Barends, Y. Chen, J. Kelly, M. Mariantoni, A. Megrant, P. O'Malley, D. Sank, A. Vainsencher, J. Wenner, T. White, Y. Yin, A. N. Cleland and John M. Martinis*

N=15, a=4, r=2 [$4^2 = 1 \bmod(15)$]

a



f



A long way to go before quantum computers solve difficult problems...

Current effort : **Scaling up quantum devices/deal with errors (Lecture 3)**
Algorithms that are less prone to errors (Lecture 4)

Summary Lecture 2

- We have seen three **algorithms that provide quantum speedup** : Deutsch's, Grover's, and Shor's algorithms

- They can be all realized in today's quantum hardware with limited number of qubits

- Running larger-scale quantum algorithms **require quantum error correction** (Lecture 3), or different types of quantum algorithms (Lecture 4)