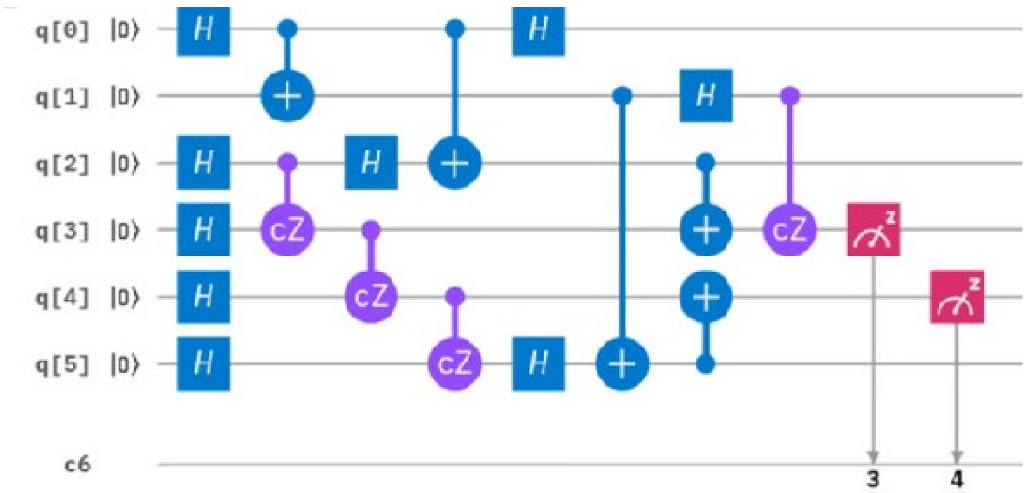
Lecture 2

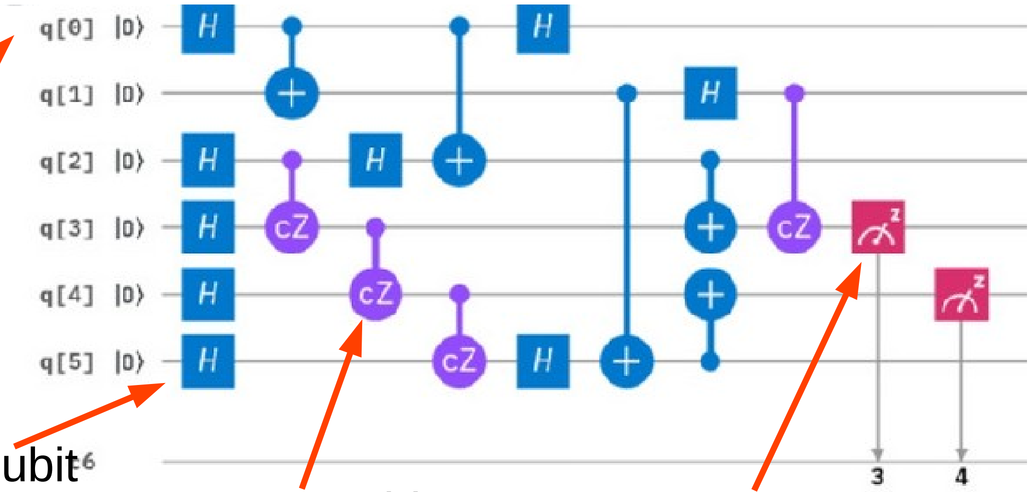
Quantum algorithms in the quantum circuit model



Benoit.vermersch@lpmmc.cnrs.fr

Reminder : A quantum circuit

Qubits



Single qubit Gate

Two-qubit gate

Measurement

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

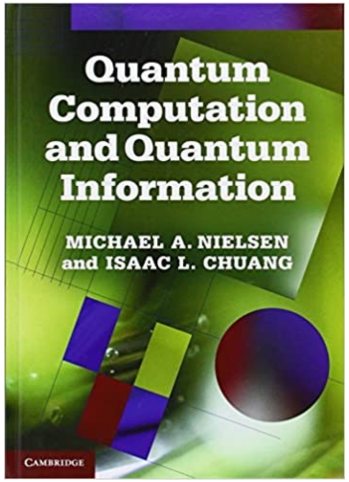
Goal 2: Having algorithms that are protected against errors (Lecture 3)

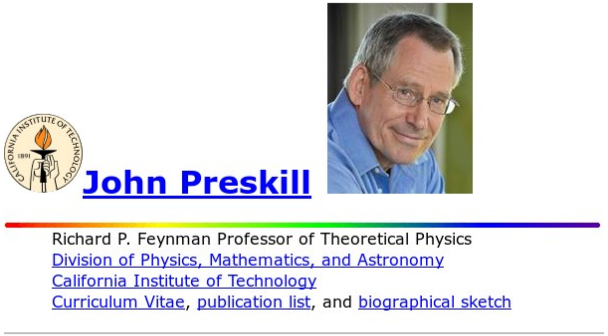
Useful references

- **Quantum computation and quantum information**

  (Nielsen and Chuang)

- **John Preskill's quantum information course:**
  http://theory.caltech.edu/~preskill/ph219/index.html





Warm-up : Deutsch’s algorithm

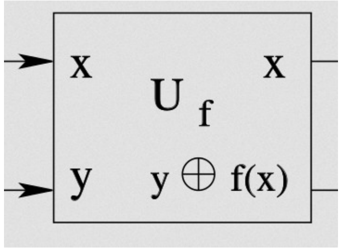
Problem: Given binary function f : [0,1]→ [0,1]. Is f(0)=f(1)?

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’ (something which is given)

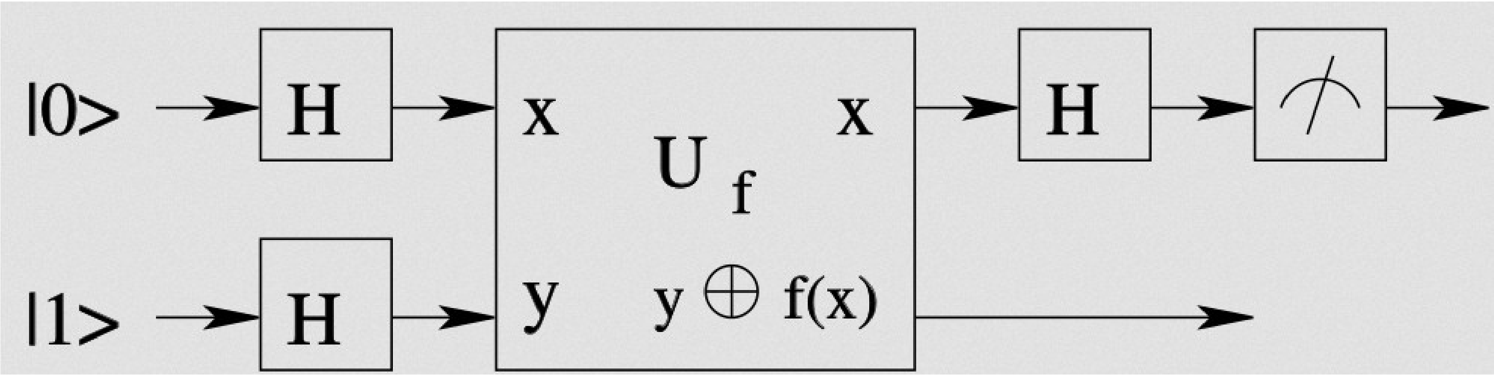


ie y is flipped iif f(x) is 1

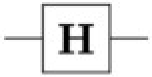
Warm-up : Deutsch's algorithm

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

Algorithm: Inject a superposition state and measure!



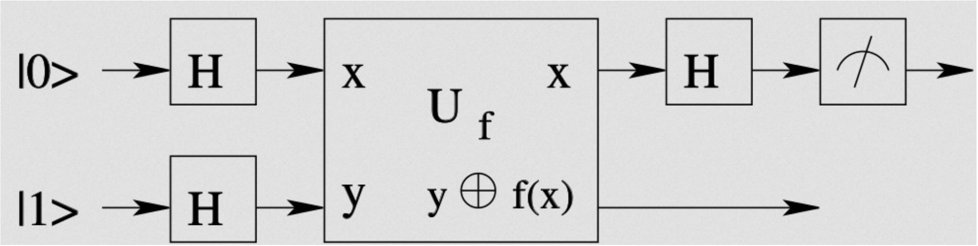
Hadamard (H)



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

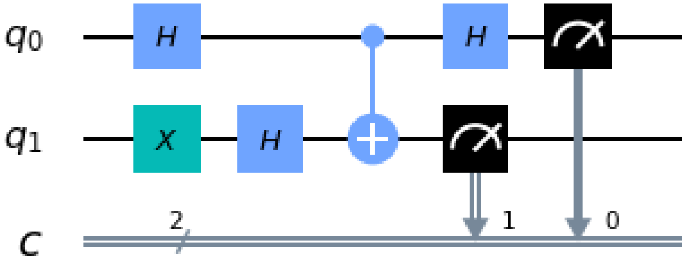
Question: What do I measure for f(0)=f(1), for f(0) != f(1) ? (using a single measurement!)

Warm-up : Deutsch’s algorithm

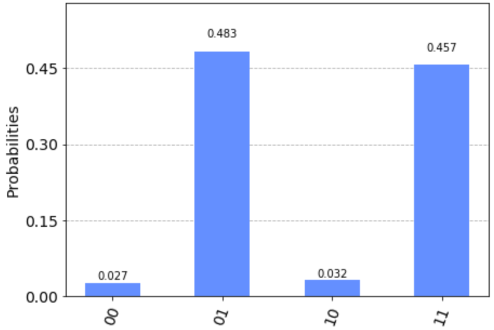


Implementation with IBM Qiskit

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

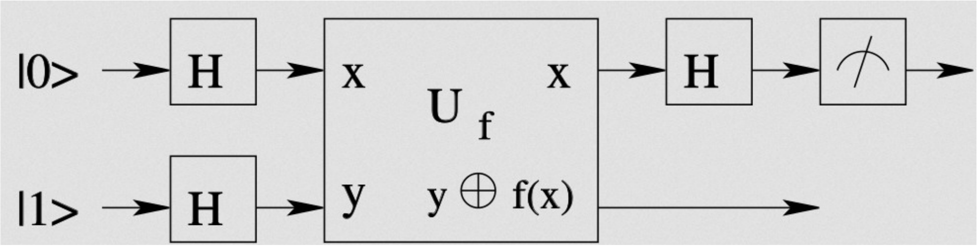


Demo with IBMQ_ourense.



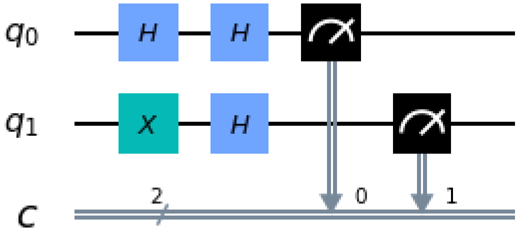
Up to errors, the first qubit ends up in |1> !

Warm-up : Deutsch’s algorithm

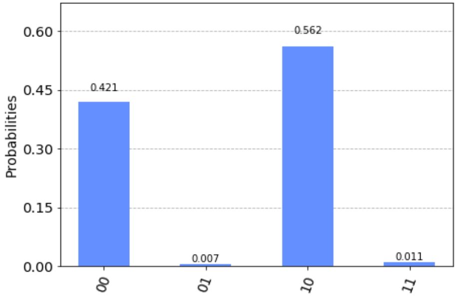


Implementation with IBM Qiskit

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



Demo with IBMQ_ourense.



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

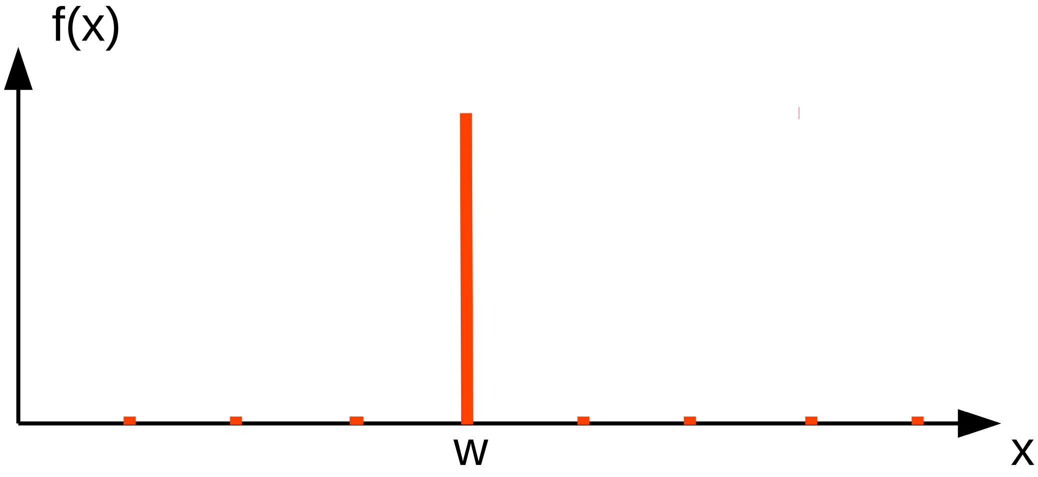
Warm-up : Deutsch’s algorithm

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

Generalizes to n qubits : Deutsch-Josza algorithm

Grover's algorithm (1996)

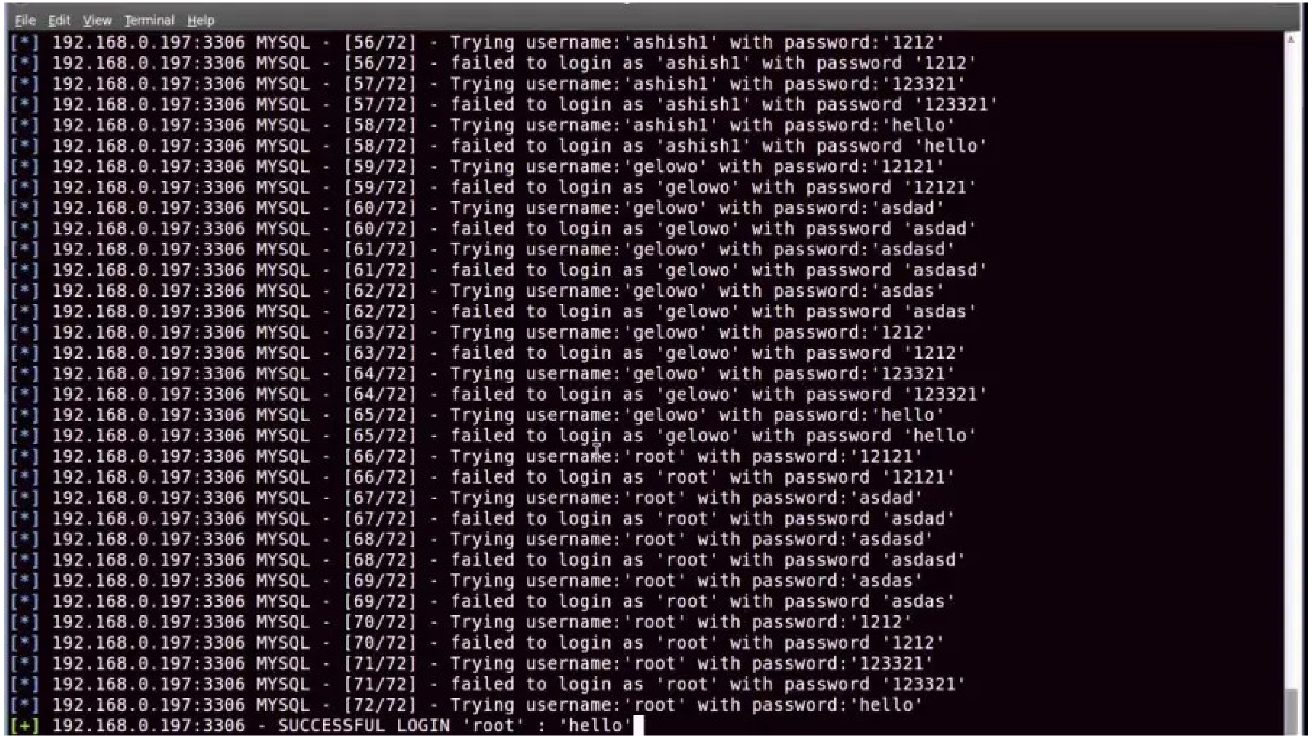
Problem: Given binay function with f(w)=1 for a single n-bit string w ($N=2^n$ is the number of configurations), find w



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

Grover’s algorithm (1996)

Classical solution : O(N) function evaluation (blind testing, brute force attacks via GPUs)



Quantum Grover’s algorithm : Simultaneous testing via quantum parallelism

Grover's algorithm (1996)

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

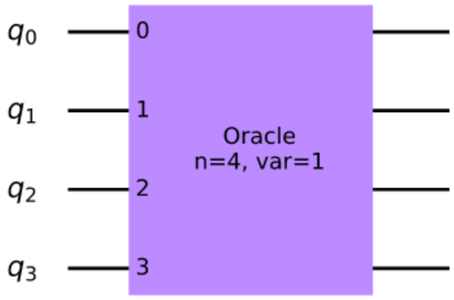
(Calling the oracle corresponds to one function f evalutation, e.g using an ancilla qubit)

$$U_f\left|x \neq w\right\rangle = \left|x\right\rangle$$

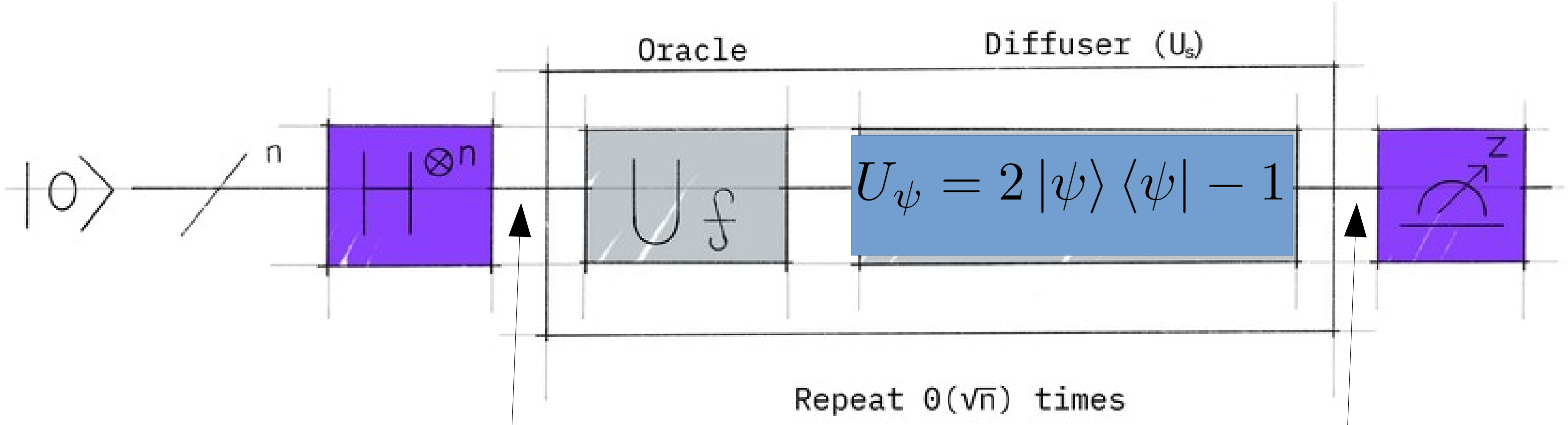
$$U_f\left|w\right\rangle = -\left|w\right\rangle$$

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)

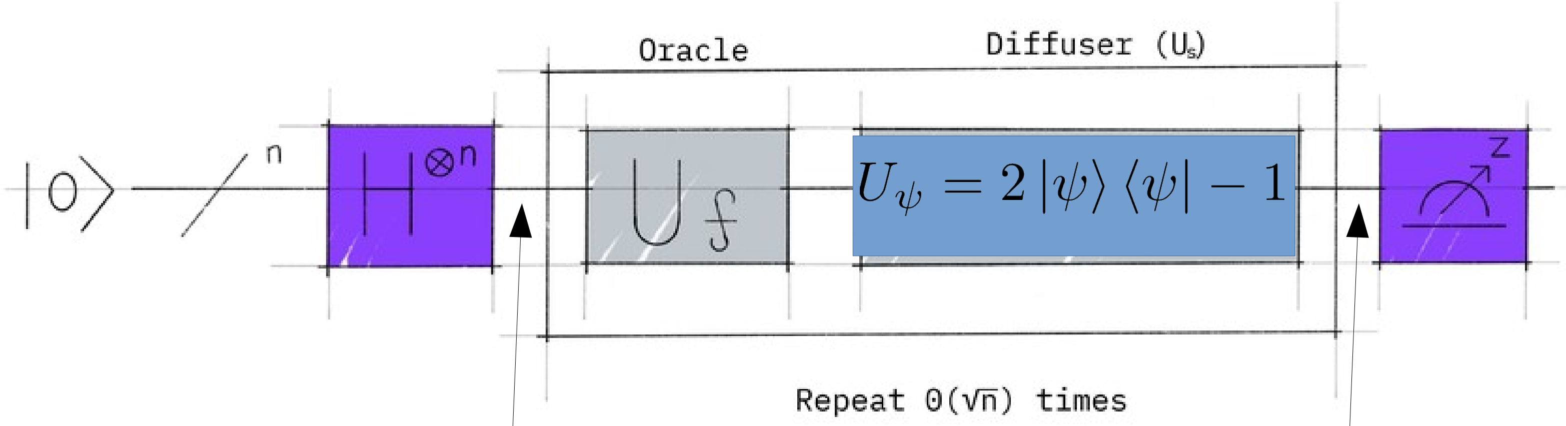


$$U_\psi = 2\,|\psi\rangle\langle\psi| - 1$$

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

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

Grover’s algorithm (1996)



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

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

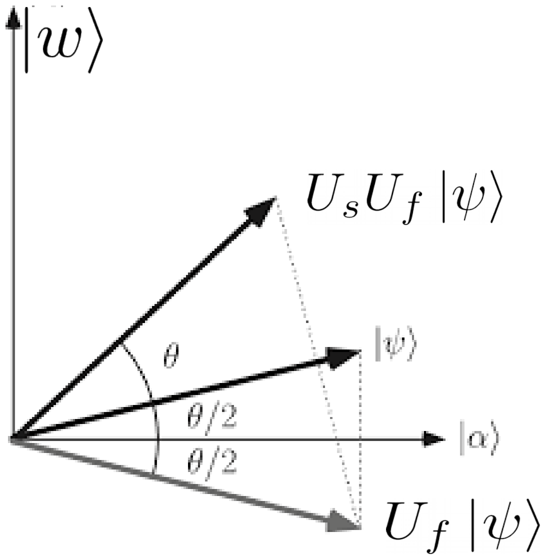
Grover's algorithm (1996)

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

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

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

$U_\psi = 2|\psi\rangle\langle\psi| - 1$ is also a reflection (in a Hilbert space of dim 2)



After one Grover iteration (let's prove it)

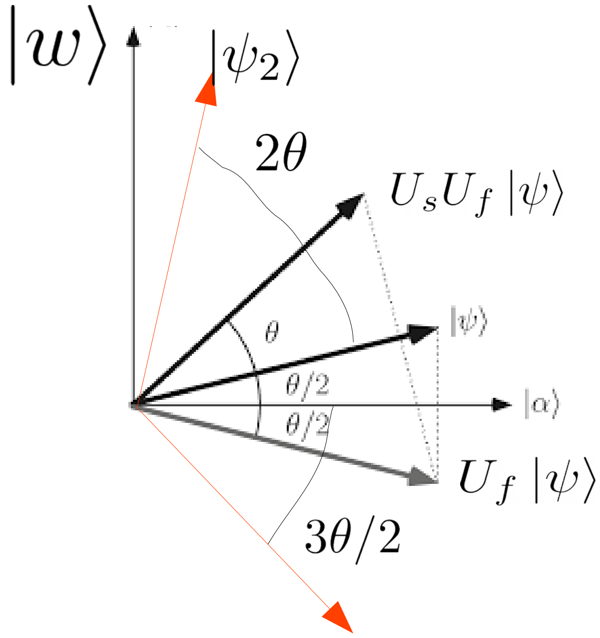
$$|\psi_1\rangle = U_s U_f |\psi\rangle = \cos(3\theta/2) |\alpha\rangle + \sin(3\theta/2) |w\rangle$$

The algorithm brings the quantum state towards the solution

Grover's algorithm (1996)

Performance

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



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

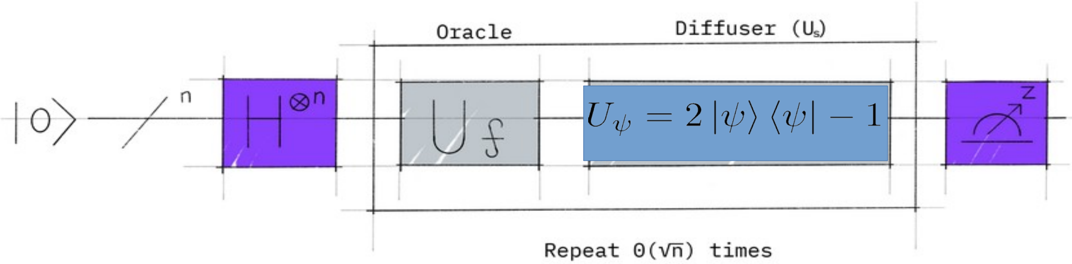
Quadratic speedup !

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)

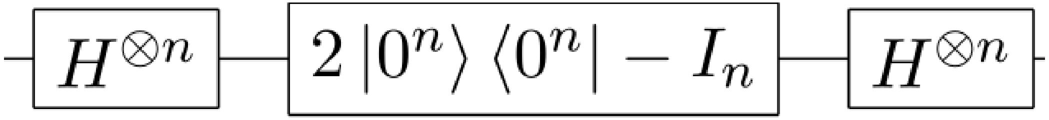
Implementation



Efficient algorithm for the symmetric reflection ?

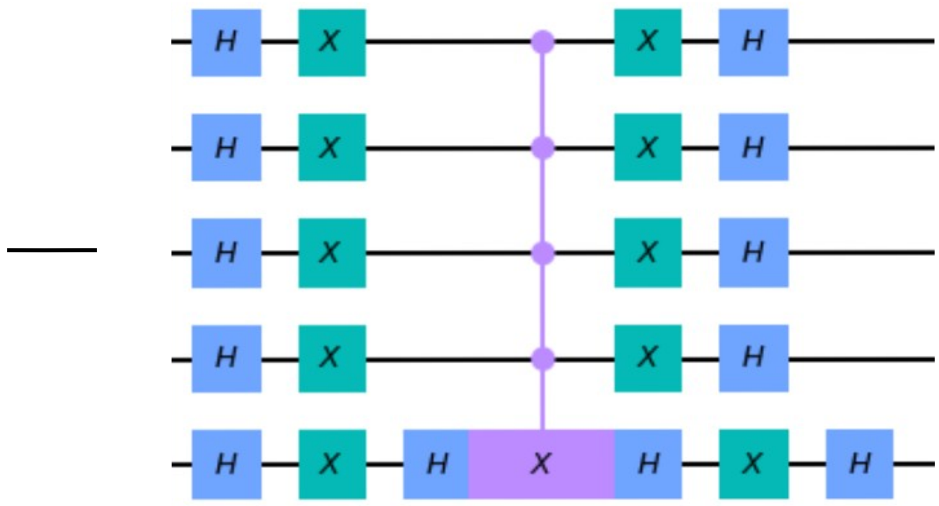
$U_\psi = 2|\psi\rangle\langle\psi| - 1$ $\quad\quad$ $|\psi\rangle = H^{\otimes n}|000\ldots0\rangle$ (ex: write it down for two qubits)

$\longrightarrow$



Grover's algorithm (1996)

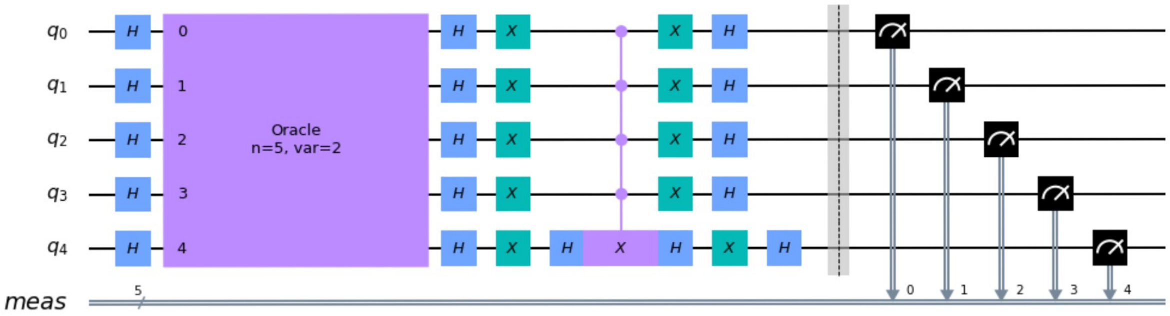
$$\rightarrow \boxed{H^{\otimes n}} - \boxed{2\,|0^n\rangle\langle 0^n| - I_n} - \boxed{H^{\otimes n}}$$

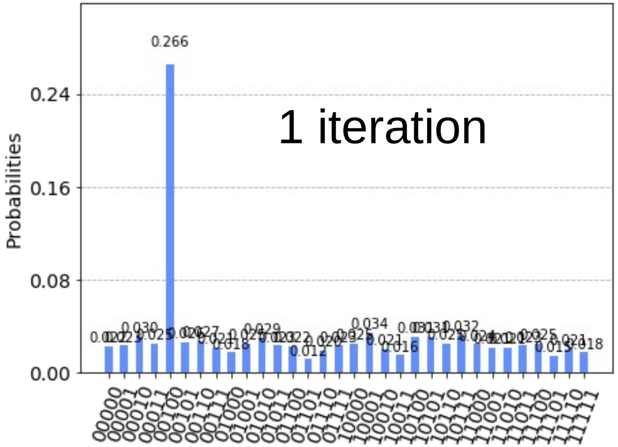


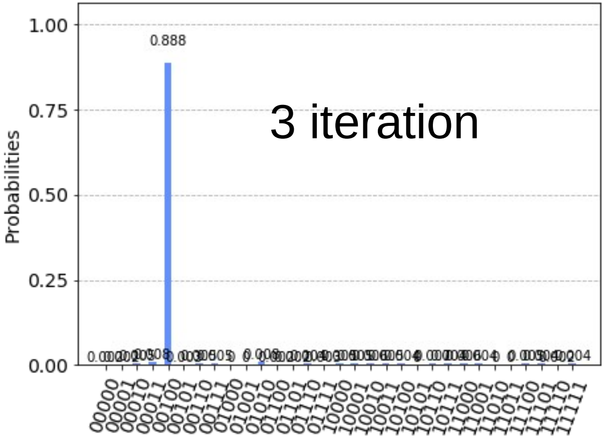
N qubit Toffoli gate (efficient implementations, see Nielsen's book for instance)

Grover’s algorithm (1996)

Illustration with Qiskit’s Aer simulator





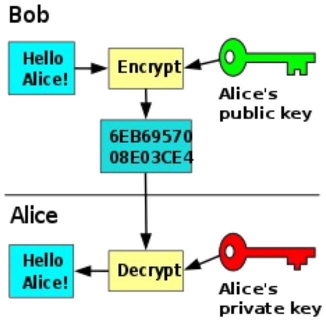


Shor’s algorithm (1995)

Factorization Problem: Given N=ab, ab coprimes, find a and b

Classical algorithm: sub-exponential in Log(N)

Quantum algorithm: polynomial in Log(N) –> Exponential speedup
→ Can break RSA cryptography...



Shor's algorithm (1995)

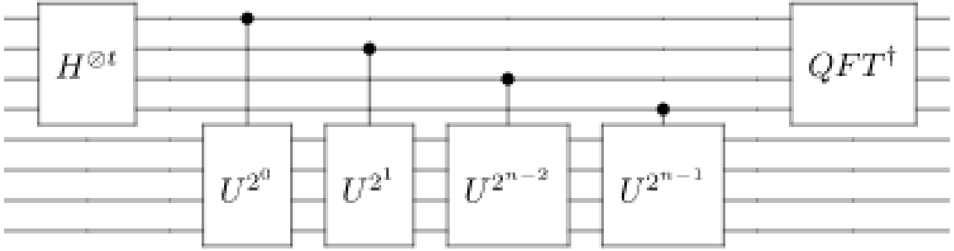
Classical part (non-trivial!): If N is a product of two co-primes and divides b^2-1, then gcd(N,b-1) and gcd(N,b+1) are non-trivial factors of N

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

Quantum Part: Goal → find b

- 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 divided a^r-1
- If r is even, b=$a^{r/2}$,therefore, N divides b^2-1

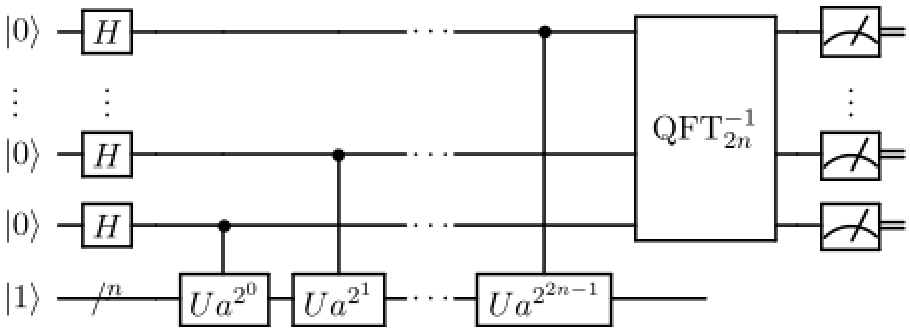
Shor's algorithm (1995)



Quantum subroutine : finding the period of $f(x) = a^x \bmod(N)$

- Choose q so that $Q=2^q>N^2$ and consider a 2q qubit quantum computer
- Prepare the first q qubits in a superposition state
- Prepare the full state in $$|\psi\rangle = \sum_x |x\rangle \otimes |a^x \bmod(N)\rangle$$
- Apply the quantum Fourier transform $$|\psi\rangle = \sum_{x,y} e^{2i\pi xy/Q} |y\rangle \otimes |a^x \bmod(N)\rangle$$

Shor's algorithm (1995)



Measurement: We access the spectrum of f, i.e we extract the periodicity r of the function

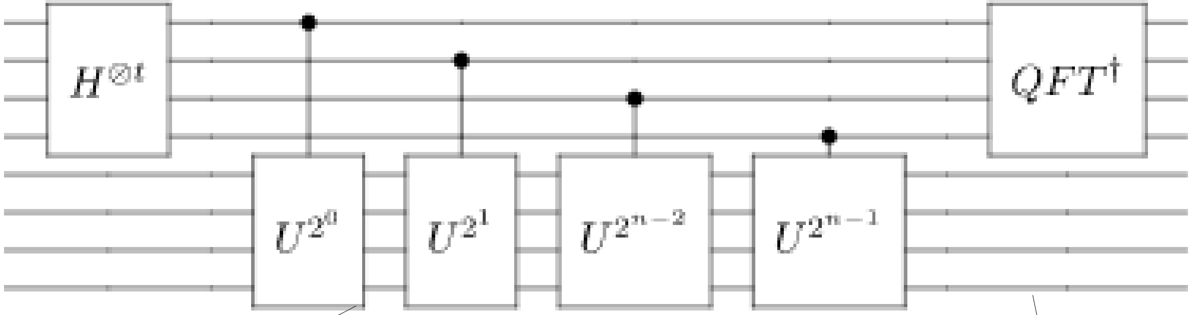
$$|\psi\rangle = \sum_{x,y} e^{2i\pi xy/Q}\,|y\rangle \otimes |a^x \mod(N)\rangle \qquad |\psi\rangle = \sum_y |y\rangle \otimes \left(\sum_x e^{2i\pi xy/Q}\,|f(x)\rangle\right) \qquad [f(x) = a^x \mod(N)]$$

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

Maximum for yr/Q integer

Shor's algorithm (1995)

Implementation aspects



Modular exponentiation
(quantum arithmetics in the ancilla space)

$$U^{2^j}\,|y\rangle = \left|a^{2^j} y \bmod(N)\right\rangle$$

Cost $O(q^3)$

Quantum Fourier Transform

Cost $O(q^2)$

nature physics

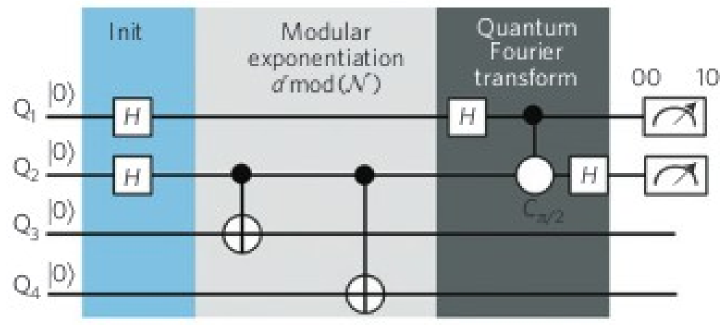
LETTERS

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

Computing prime factors with a Josephson phase qubit quantum processor

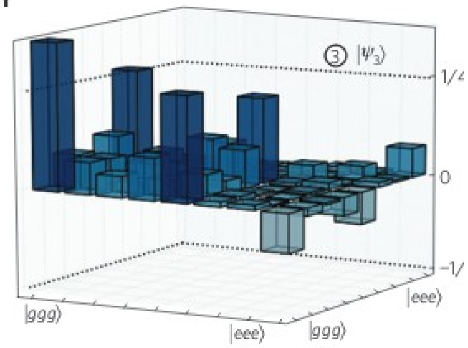
Erik Lucero, R. Barends, Y. Chen, J. Kelly, M. Mariantoni, A. Megrant, P. O'Malley, D. Sank, A. Vainsencher, J. Wenner, T. White, Y. Yin, A. N. Cleland and John M. Martinis*

a



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

f

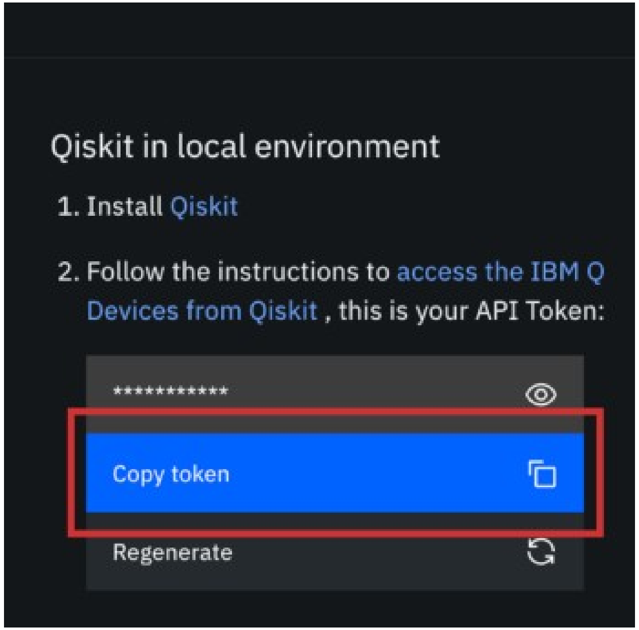


50 % success to factor 15

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

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



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)