Măgurele Summer School for Computing in a Rapidly Evolving Society: Parallel Algorithms and Optimizations

Quantum algorithms on IBM's quantum platform

by Stefan Ataman

Extreme Light Infrastructure - Nuclear Physics (ELI-NP)

July 11, 2025

The (small) quantum part of this summer school:

10/07/2025:

- 9h00 – 10h30 The future is quantum by Radu Ionicioiu. ✓
- 11h00 – 12h30 Quantum information: foundations and entanglement by Radu Ionicioiu. ✓
- 14h00 – 15h30 One and two-qubit quantum gates, quantum circuits & measurement by S. A. ✓
- 16h00 – 17h30 One and two-qubit applications on IBM's quantum computers by S. A. ✓

11/07/2025:

- 9h00–10h30 Quantum information: protocols and applications by Radu Ionicioiu. ✓
- 11h00–12h30 Quantum algorithms on IBM's quantum computers (Deutsch, Bernstein-Vazirani, Grover) by S. A.

Outlook today

Quantum computation

Some history.

Quantum computation – algorithms

The Deutsch algorithm + IBMqe implementation
The Deutsch-Jozsa algorithm + IBMqe implementation
The Bernstein-Vazirani algorithm + IBMqe implementation
The Grover algorithm + IBMqe implementation
The quantum Fourier transform + IBMqe implementation

Table of Contents

The early days

Paul Benioff - The computer as a physical system - 1980

The first quantum mechanical description of the Turing machine

Richard Feynman ("Simulating physics with computers" – 1982)

"...trying to find a computer simulation of physics seems to me to be an excellent program to follow out...the real use of it would be with quantum mechanics... **Nature isn't classical...and if you want to make a simulation of Nature, you'd better make it quantum mechanical**, and by golly it's a wonderful problem, because it doesn't look so easy. "

Benioff, P. *The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines.*, J Stat Phys 22, 563 (1980)
Feynman, R.P, *Simulating physics with computers*, Int J Theor Phys 21, 467 (1982)

Quantum algorithms

Algorithm type	Speedup	Name	Year
Deutsch problem	(Exponential)	Deutsch algorithm	1985
Deutsch,Jozsa	Exponential	Deutsch-Jozsa algorithm	1992
Bernstein-Vazirani problem	$\mathcal{O}(N)$	Bernstein-Vazirani alg.	1992
Simon's problem	Exponential	Simon's algorithm	1994
Integer factoring	Exponential	**Shor's** algorithm	1996
Optimisation / combinatorial search	$\mathcal{O}\left(\sqrt{N}\right)$	**Grover's algorithm**	1996
Quantum heuristics	Unknown	QAOA	2014
High-dimensional linear algebra	Exponential?	HHL	2018
Integer factoring – Shor on steroids	Exponential	Regev's algorithm	2023

David Deutsch, *Quantum theory, the Church-Turing principle and the universal quantum computer*, Proc. R. Soc. Lond. A400 97 (1985)
David Deutschand Richard Jozsa, *Rapid solution of problems by quantum computation*, Proc. R. Soc. Lond. A439 553 (1992)
Ethan Bernstein and Umesh Vazirani, *Quantum Complexity Theory*, SIAM Journal on Computing 26 1411 (1992)
Lov K. Grover, *A fast quantum mechanical algorithm for database search*, (STOC '96). 212 (1996)
Oded Regev *An Efficient Quantum Factoring Algorithm*, arXiv:2308.06572 [quant-ph] (2023)

See also: https://quantumalgorithmzoo.org

Table of Contents

The Deutsch algorithm

First quantum algorithm to outperform a classical one. David Deutsch (1985)

A function $f(x)$, $x \in \{0,1\}$ and $f(x) \in \{0,1\}$ is

- either constant ($f(0) = f(1)$)
- or balanced ($f(0) \neq f(1)$).

How many interrogations are needed to find out if $f(x)$ is constant or balanced?

Classical: two function (oracle) interrogations.

First input "0". See the function's (or oracle's) answer. Then input "1" and see what output you get.

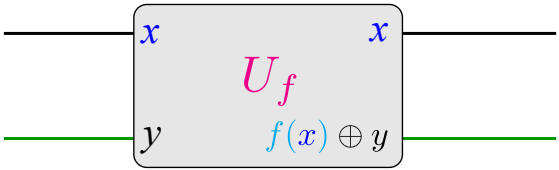
Quantum:

Could we do better than that?

The Deutsch algorithm

First quantum algorithm to outperform a classical one. David Deutsch (1985)

A function $f(x)$, $x \in \{0,1\}$ and $f(x) \in \{0,1\}$ is:
– either constant ($f(0) = f(1)$)
– or balanced ($f(0) \neq f(1)$).
How many interrogations are needed to find out if $f(x)$ is constant or balanced?



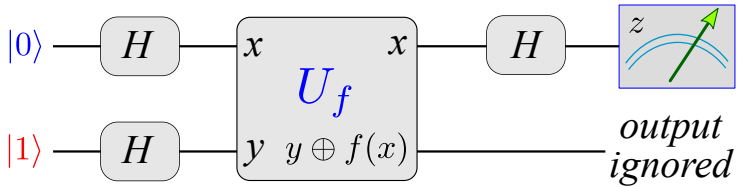
Classical:
two function (oracle) interrogations.

Deutsch's algorithm:
one interrogation.

The Deutsch algorithm

First quantum algorithm to outperform a classical one. David Deutsch (1985)

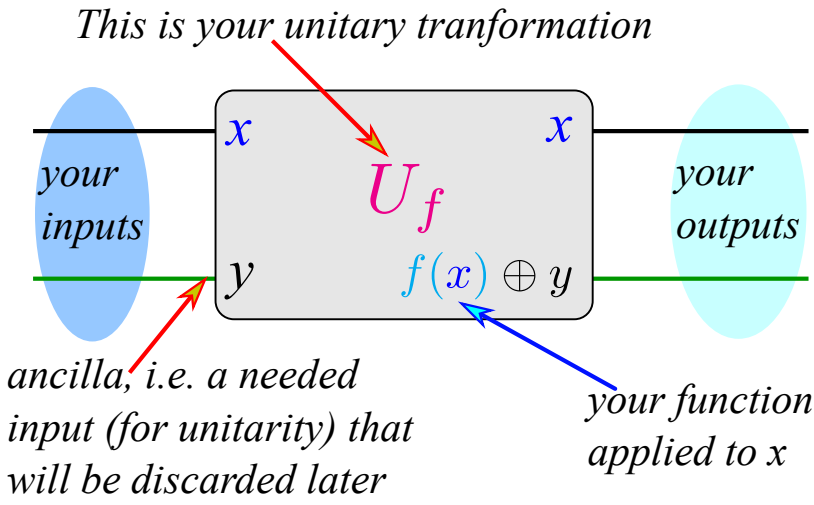
A function $f(x)$, $x \in \{0,1\}$ and $f(x) \in \{0,1\}$ is either constant ($f(0) = f(1)$) or balanced ($f(0) \neq f(1)$). How many interrogations are needed to find out if $f(x)$ is constant or balanced?



Classical: two function (oracle) interrogations.

Quantum: Deutsch algorithm, one interrogation.

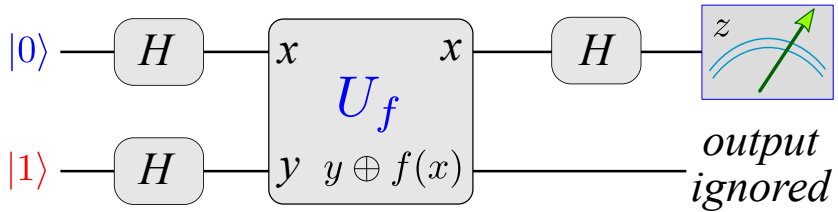
Quantum algorithms: ancillas are usually needed



The Deutsch algorithm

First quantum algorithm to outperform a classical one. David Deutsch (1985)

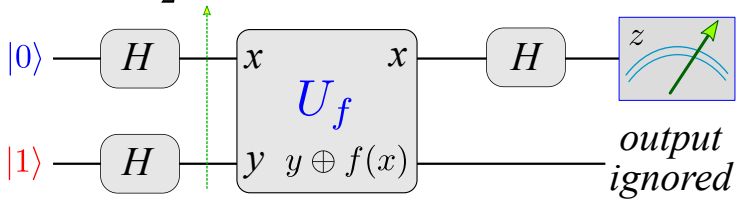
A function $f(x)$, $x \in \{0,1\}$ and $f(x) \in \{0,1\}$ is either constant ($f(0) = f(1)$) or balanced ($f(0) \neq f(1)$).



By the way, I hope you figured out by now that $\oplus$ denotes addition modulo 2.
Let's discuss this quantum circuit in a step by step manner.

The Deutsch algorithm

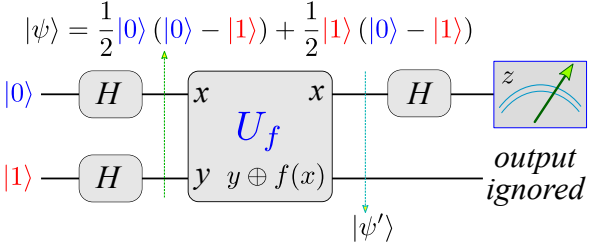
$$|\psi\rangle = \frac{1}{2}\left(|0\rangle + |1\rangle\right)\left(|0\rangle - |1\rangle\right)$$



The state vector **before the oracle** is

$$|\psi\rangle = \frac{1}{2}\underbrace{\left(|0\rangle + |1\rangle\right)}_{\text{top qubit}}\underbrace{\left(|0\rangle - |1\rangle\right)}_{\text{bottom qubit}} \quad \text{so expanded} \quad |\psi\rangle = \frac{1}{2}|0\rangle\left(|0\rangle - |1\rangle\right) + \frac{1}{2}|1\rangle\left(|0\rangle - |1\rangle\right)$$

The Deutsch algorithm



The oracle applies its transformation (meaning $y \oplus f(x)$) and we have (after the oracle)

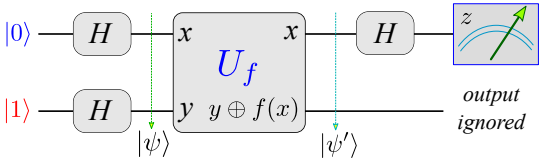
$$|\psi'\rangle = \frac{1}{2}\underbrace{|0\rangle}_{\text{control}}\left(|\underbrace{0 \oplus f(0)}_{\text{oracle}}\rangle - |\underbrace{1 \oplus f(0)}_{\text{oracle}}\rangle\right) + \frac{1}{2}\underbrace{|1\rangle}_{\text{control}}\left(|\underbrace{0 \oplus f(1)}_{\text{oracle}}\rangle - |\underbrace{1 \oplus f(1)}_{\text{oracle}}\rangle\right)$$

We now discuss the two parts of this wavevector separately.

The Deutsch algorithm

The state after the oracle is

$$|\psi'\rangle = \frac{1}{2}\underbrace{|0\rangle}_{\text{control}}\left(|\underbrace{0 \oplus f(0)}_{\text{oracle}}\rangle - |\underbrace{1 \oplus f(0)}_{\text{oracle}}\rangle\right) + \frac{1}{2}\underbrace{|1\rangle}_{\text{control}}\left(|0 \oplus f(1)\rangle - |1 \oplus f(1)\rangle\right)$$

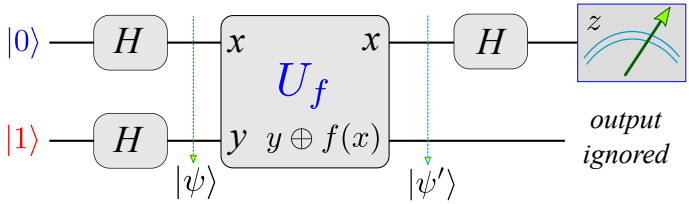


Recall that f is a very simple function with just two possible input/output values.

If $f(0) = 0$ then $0 \oplus f(0) = 0$ and $1 \oplus f(0) = 1$, thus the first part of $|\psi'\rangle$ is

$$\frac{1}{2}\underbrace{|0\rangle}_{\text{control}}\left(|0 \oplus f(0)\rangle - |1 \oplus f(0)\rangle\right) = \frac{1}{2}|0\rangle\left(|0\rangle - |1\rangle\right) = \frac{1}{2}\underbrace{(-1)^{f(0)}}_{=1}|0\rangle\left(|0\rangle - |1\rangle\right)$$

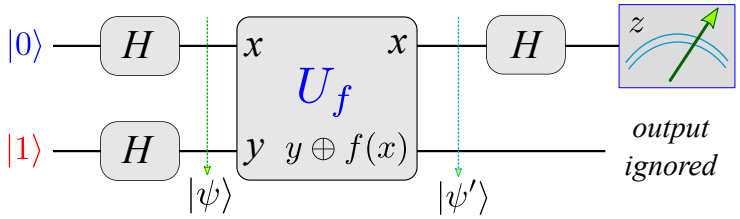
The Deutsch algorithm



If $f(0) = 1$ then $0 \oplus f(0) = 1$ and $1 \oplus f(0) = 0$. Thus, the first part of $|\psi'\rangle$ is:

$$\frac{1}{2} \underbrace{|0\rangle}_{\text{control}} \left(|0 \oplus f(0)\rangle - |1 \oplus f(0)\rangle\right) = \frac{1}{2}|0\rangle \left(|1\rangle - |0\rangle\right) = \frac{1}{2} \underbrace{(-1)^{f(0)}}_{=-1} |0\rangle \left(|0\rangle - |1\rangle\right)$$

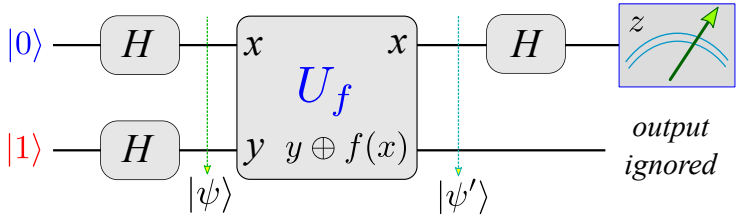
The Deutsch algorithm



Similarly if $f(1) = 0$ then $1 \oplus f(1) = 1$ and $1 \oplus f(1) = 0$, thus

$$\frac{1}{2}|1\rangle\left(|0 \oplus f(1)\rangle - |1 \oplus f(1)\rangle\right) = \frac{1}{2}|1\rangle\left(-|0\rangle + |1\rangle\right) = \frac{1}{2}(-1)^{f(1)}|1\rangle\left(|0\rangle - |1\rangle\right)$$

The Deutsch algorithm



To wrap it all up, we have the state

$$|\psi'\rangle = \frac{1}{2}\left((-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle\right)(|0\rangle - |1\rangle)$$

An interesting way of writing our quantum state. But does it advance us?

The Deutsch algorithm

However, $f(0) \oplus f(0) = 0$. (Can you see why?) Thus, we can add a factor

$$(-1)^{f(0)\oplus f(0)} = (-1)^{f(0)} \cdot (-1)^{f(0)} = 1$$

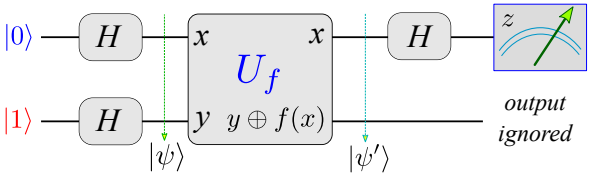
in front of $|1\rangle$. (Why not? Let' be crazy!) So we have the state:

$$|\psi'\rangle = (-1)^{f(0)} \frac{1}{2} \left(|0\rangle + (-1)^{f(0)\oplus f(1)} |1\rangle \right) (|0\rangle - |1\rangle)$$

but **a global phase is irrelevant**, thus we throw out $(-1)^{f(0)}$ and have:

$$|\psi'\rangle = \frac{1}{2} \left(|0\rangle + (-1)^{f(0)\oplus f(1)} |1\rangle \right) (|0\rangle - |1\rangle)$$

Stop! Did you notice something unusual?



We started with

$$|\psi'\rangle = \frac{1}{2}|0\rangle\Big(|\underbrace{0 \oplus f(0)}_{\text{oracle}}\rangle - |\underbrace{1 \oplus f(0)}_{\text{oracle}}\rangle\Big) + \frac{1}{2}|1\rangle\Big(|\underbrace{0 \oplus f(1)}_{\text{oracle}}\rangle - |\underbrace{1 \oplus f(1)}_{\text{oracle}}\rangle\Big)$$

and we ended up with:

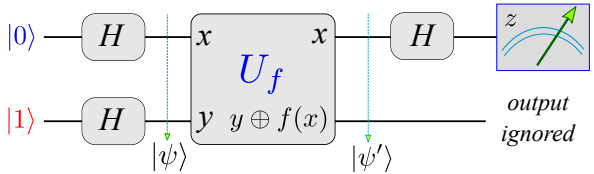
$$|\psi'\rangle = \frac{1}{2}\Big(|0\rangle + (-1)^{f(0)\oplus f(1)}|1\rangle\Big)(|0\rangle - |1\rangle)$$

The phase kickback mechanism

How the "magic" happens:
The state before the oracle when the top qubit is $|0\rangle$ is

$$|\psi\rangle\,|_{|0\rangle} \sim |0\rangle\rangle\,(|0\rangle - |1\rangle\rangle) \sim |0\rangle\,|-\rangle$$

so after the oracle we have



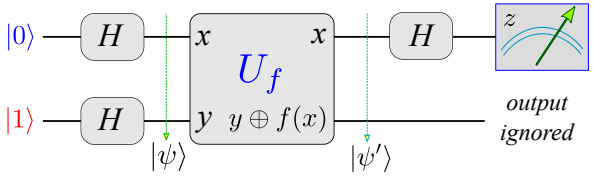
$$|0\rangle\,|-\rangle \longrightarrow |0\rangle\,e^{\phi_0}\,|-\rangle = e^{\phi_0}\,|0\rangle\,|-\rangle$$

The state before the oracle when the top qubit is $|1\rangle$ is $|\psi\rangle\,|_{|1\rangle} \sim |1\rangle\rangle\,(|0\rangle - |1\rangle) \sim |1\rangle\,|-\rangle$ so after the oracle we have

$$|1\rangle\,|-\rangle \longrightarrow |1\rangle\,e^{\phi_1}\,|-\rangle = e^{\phi_1}\,|1\rangle\,|-\rangle$$

The phase kickback mechanism

We wrap it up and have **after the oracle**:



$$\begin{cases} |0\rangle\,|-\rangle \longrightarrow |0\rangle\, e^{\phi_0}\,|-\rangle = e^{\phi_0}\,|0\rangle\,|-\rangle \\ |1\rangle\,|-\rangle \longrightarrow |1\rangle\, e^{\phi_1}\,|-\rangle = e^{\phi_1}\,|1\rangle\,|-\rangle \end{cases}$$

and since we actually have the superposition $|0\rangle + |1\rangle$ in the upper qubit we have

$$\underbrace{(|0\rangle + |1\rangle)}_{\text{upper qubit}}\,|-\rangle \longrightarrow e^{\phi_0}\,|0\rangle\,|-\rangle + e^{\phi_1}\,|1\rangle\,|-\rangle = e^{\phi_0}\left(|0\rangle + \underbrace{e^{(\phi_1-\phi_0)}}_{\text{phase kickback}}\,|1\rangle\right)|-\rangle$$

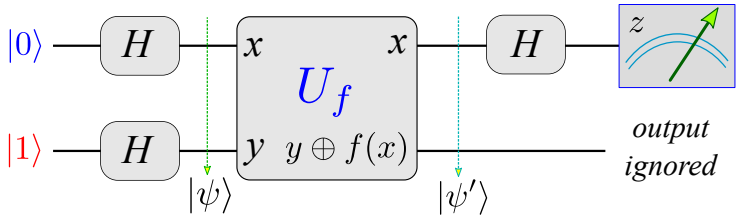
Back to the Deutsch algorithm

So after U_f we have

$$|\psi'\rangle = \frac{1}{2}\left(|0\rangle + (-1)^{f(0)\oplus f(1)}|1\rangle\right)(|0\rangle - |1\rangle)$$

and applying the last Hadamard gate we arrive at

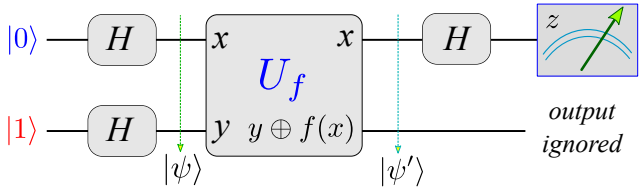
$$|\psi_{out}\rangle = \frac{1}{2}\left(\underbrace{H|0\rangle}_{=?} + (-1)^{f(0)\oplus f(1)}\underbrace{H|1\rangle}_{=?}\right)(|0\rangle - |1\rangle)$$



The Deutsch algorithm

So after U_f and applying the last Hadamard gate we arrive at

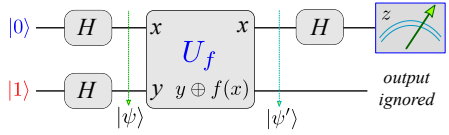
$$|\psi_{out}\rangle = \frac{1}{2}\Big(\underbrace{H\,|0\rangle}_{=\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)} + (-1)^{f(0)\oplus f(1)} \underbrace{H\,|1\rangle}_{=\frac{1}{\sqrt{2}}(|0\rangle-|1\rangle)} \Big)\,(|0\rangle - |1\rangle)$$



or, expanded:

$$|\psi_{out}\rangle = \frac{1}{2}\Big(|0\rangle + |1\rangle + (-1)^{f(0)\oplus f(1)}\,(|0\rangle - |1\rangle) \Big)\frac{1}{\sqrt{2}}\,(|0\rangle - |1\rangle)$$

The Deutsch algorithm



So after U_f and applying the last Hadamard gate we arrive at

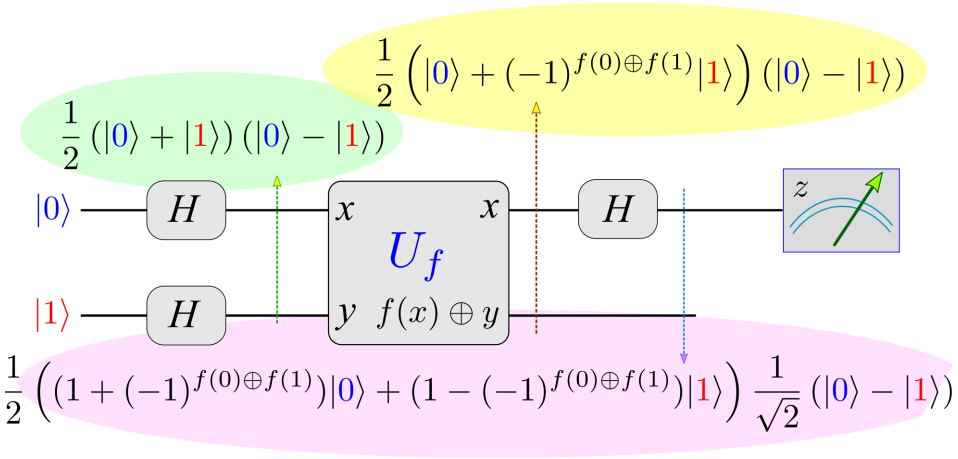
$$|\psi_{out}\rangle = \frac{1}{2}\left((1+(-1)^{f(0)\oplus f(1)})|0\rangle + (1-(-1)^{f(0)\oplus f(1)})|1\rangle\right)\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

Please note that the state after the oracle **is separable**. We have

$$|\psi_{out}\rangle = \underbrace{\frac{1}{2}\left((1+(-1)^{f(0)\oplus f(1)})|0\rangle + (1-(-1)^{f(0)\oplus f(1)})|1\rangle\right)}_{\text{upper qubit}} \otimes \underbrace{\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)}_{\text{lower qubit}}$$

We can thus **ignore the lower qubit**.

The Deutsch algorithm



The Deutsch algorithm

The last step is the **measurement of the upper qubit**. The relevant state is:

$$|\psi_{out}\rangle\,|_{\text{upper qubit only}} = \frac{1}{2}\left[\left(1+(-1)^{f(0)\oplus f(1)}\right)|0\rangle + \left(1-(-1)^{f(0)\oplus f(1)}\right)|1\rangle\right]$$

If the function is constant we have $f(0)\oplus f(1)=0$ thus replacing in the above equation

$$|\psi_{out}\rangle\,|_{\text{upper qubit only}} = \frac{1}{2}\left[(1+(-1)^{0})|0\rangle + (1-(-1)^{0})|1\rangle\right]$$

however:

$$(1+(-1)^{0})=2 \quad \text{and} \quad (1-(-1)^{0})=0$$

so we end up with

$$|\psi_{out}\rangle\,|_{\text{upper qubit only}} = \frac{1}{2}\left(\underbrace{\left(1+(-1)^{f(0)\oplus f(1)}\right)}_{=2}|0\rangle + \underbrace{(1-(-1)^{f(0)\oplus f(1)})}_{=0}|1\rangle\right) = |0\rangle$$

The Deutsch algorithm

If the function is balanced we have $f(0) \oplus f(1) = 1$ thus

$$|\psi_{out}\rangle\,|_{\text{upper qubit only}} = \frac{1}{2}\left(\left(1+(-1)^1\right)|0\rangle + \left(1-(-1)^1\right)|1\rangle\right)$$

however:

$$(1+(-1)^1) = 0 \quad \text{and} \quad (1-(-1)^1) = 2$$

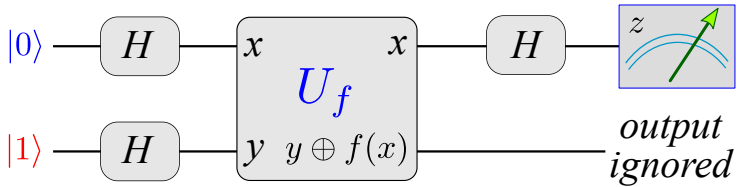
so

$$|\psi_{out}\rangle\,|_{\text{upper qubit only}} = \frac{1}{2}\left(\underbrace{\left(1+(-1)^{f(0)\oplus f(1)}\right)}_{=0}|0\rangle + \underbrace{\left(1-(-1)^{f(0)\oplus f(1)}\right)}_{=2}|1\rangle\right) = |1\rangle$$

The Deutsch algorithm

Conclusion:

We have **a deterministic answer**, after **a single** interrogation.



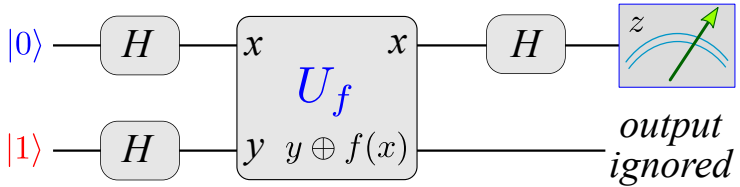
IBM Quantum Experience

Next – we implement it!

Table of Contents

The Deutsch algorithm - IBMqe implementation

Let's implement Deutsch's algorithm in IBM's Quantum Learning platform.



Recall:

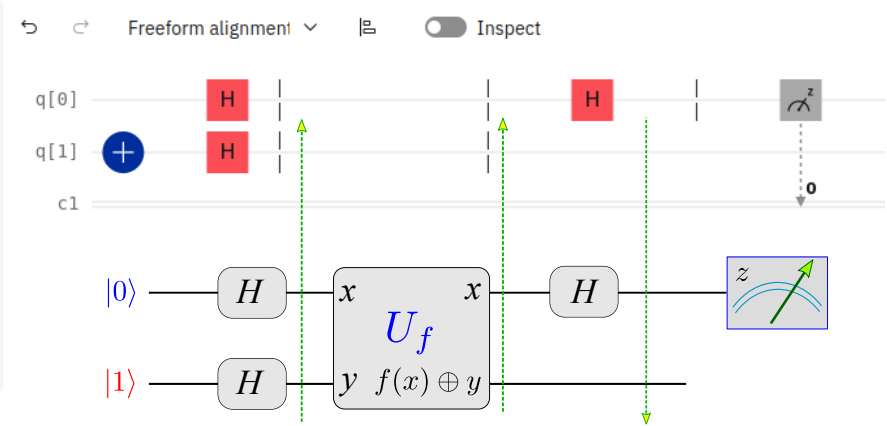
You have 1 qubit for the input/output x.
You need an extra 1 qubit (tha ancilla) for the encoding of the function $f(x)$.

Suggestion

Start with a constant function $f(x)$.
Then, consider the balanced case, too.

The Deutsch algorithm - IBM quantum implementation

Say $f(x) = 0$. So $y \oplus f(x) = y$. We have the IBM quantum implementation:



The Deutsch algorithm - IBM quantum implementation

Say $f(x)$ is balanced. And let's assume $f(x) = x$. In this case $y \oplus f(x) = y \oplus x$.
Wait a minute! We have the transformation $|xy\rangle \rightarrow |xy \oplus x\rangle$. This is a CNOT!

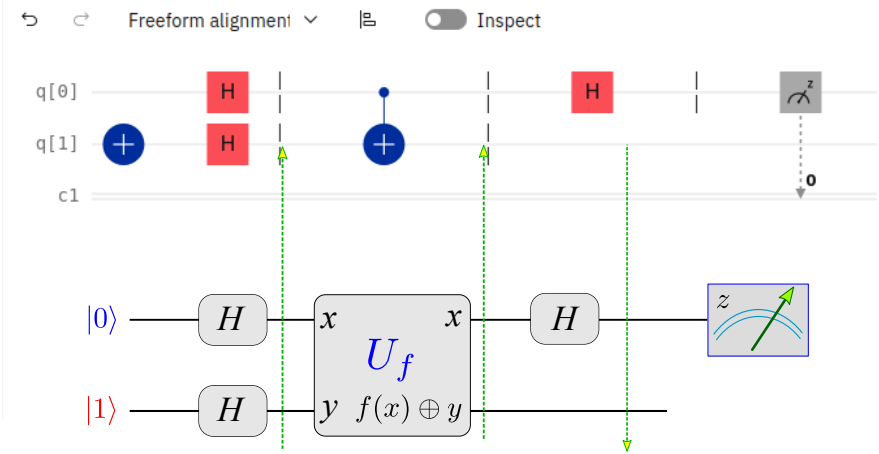
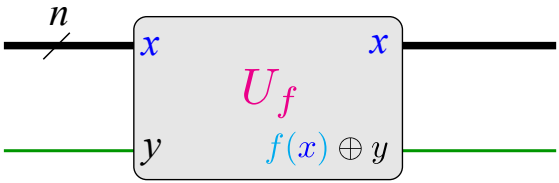


Table of Contents

The Deutsch-Jozsa algorithm

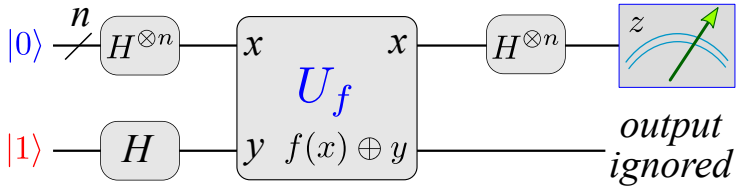
The Deutsch-Jozsa algorithm is the logical extension of the Deutsch algorithm to n qubits.



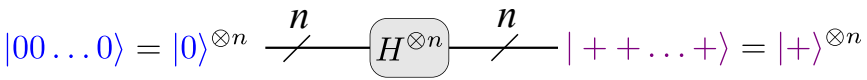
This time the input is on n bits (qubits), $x = \{x_0, x_1, \ldots x_{n-1}\}$.
So we have a function (oracle) $f(x)$, $x \in \{0,1\}^{\otimes n}$ and $f(x) \in \{0,1\}$ that is:
– either constant ($f(0) = f(1) = \ldots f(2^n - 1)$)
– or balanced ($2^n/2$ inputs yields $f(x) = 0$ and the other $2^n/2$ inputs yields $f(x) = 1$).

The Deutsch-Jozsa algorithm

As states, the Deutsch-Jozsa algorithm is the logical extension of the Deutsch algorithm to n qubits. So is the implementation, too:



Hadamard gates on n qubits



We have n qubits

$$|00\ldots0\rangle$$

and we apply n Hadamard gates. This takes us to

$$\frac{1}{\sqrt{2^n}}\left(|0\rangle+|1\rangle\right)\left(|0\rangle+|1\rangle\right)\ldots\left(|0\rangle+|1\rangle\right)$$

and this is actually

$$\frac{1}{\sqrt{2^n}}\left(|00\ldots0\rangle+|0\ldots01\rangle+|0\ldots010\rangle+\ldots+|11\ldots1\rangle\right)$$

Hadamard gates on n qubits – let's introduce a new notation

$$\frac{1}{\sqrt{2^n}}\left(|00\ldots0\rangle + |0\ldots01\rangle + |0\ldots010\rangle + \ldots + |11\ldots1\rangle\right)$$

Now here we have all possible combinations of 0s and 1s on n qubits. If we denote:

$$\begin{aligned}
|0\rangle &= |x=0\rangle = |0\ldots0000\rangle,\\
|1\rangle &= |x=1\rangle = |0\ldots0001\rangle,\\
|2\rangle &= |x=2\rangle = |0\ldots0010\rangle,\\
|3\rangle &= |x=3\rangle = |0\ldots0011\rangle,\\
|4\rangle &= |x=4\rangle = |0\ldots0100\rangle,\\
|5\rangle &= |x=5\rangle = |0\ldots0101\rangle,\\
&\ldots
\end{aligned}$$

Hadamard gates on n qubits – let's introduce a new notation

So we had

$$\frac{1}{\sqrt{2^n}}\left(|00\ldots0\rangle + |0\ldots01\rangle + |0\ldots010\rangle + \ldots + |11\ldots1\rangle\right)$$

and we can write our state before the oracle as (ignoring the lower qubit)

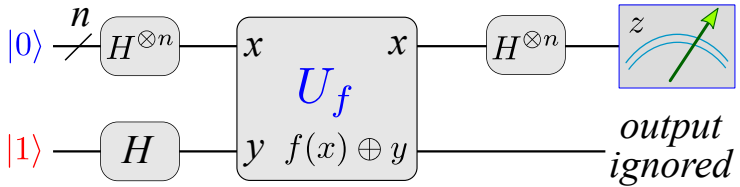
$$\frac{1}{\sqrt{2^n}}\left(|00\ldots0\rangle + |0\ldots01\rangle + |0\ldots010\rangle + \ldots + |11\ldots1\rangle\right) = \frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}|x\rangle$$

This notation **will be crucial** in the following.

The Deutsch-Jozsa algorithm

The state before the oracle is

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



and after the oracle we have

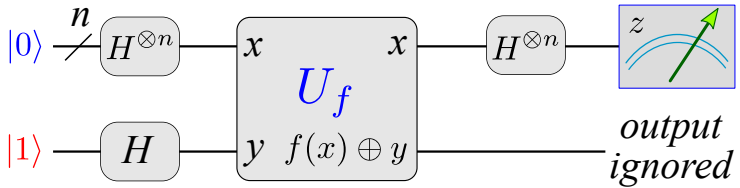
$$|\psi'\rangle = \frac{1}{\sqrt{2^{n+1}}} \sum_{x=0}^{2^n-1} |x\rangle \left(|0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle\right) = \frac{1}{\sqrt{2^{n+1}}} \sum_{x=0}^{2^n-1} (-1)^{f(x)} |x\rangle \left(|0\rangle - |1\rangle\right)$$

This state, too, is separable, we thus ignore the last qubit.

The Deutsch-Jozsa algorithm

With the last qubit ignored we have

$$|\psi'\rangle\,|_{\text{upper qubits only}} = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} (-1)^{f(x)} |x\rangle$$



We have to figure out the state after the last Hadamard transformations. Let's see.

The Deutsch-Jozsa algorithm

I claim that the effect of the last Hadamard gates can be written as

$$H^{\otimes n}|x\rangle \longrightarrow \frac{1}{\sqrt{2^n}} \sum_{w=0}^{2^n-1} (-1)^{x\cdot w}|w\rangle$$

Is this formula correct? Let's see. If $|x\rangle = |0\rangle$ we have

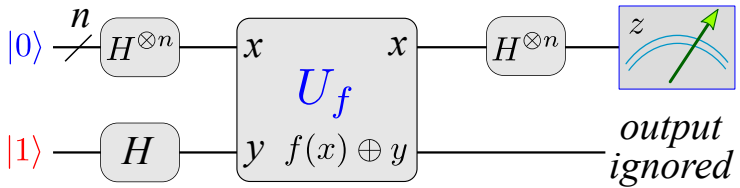
$$H^{\otimes n}|0\rangle \longrightarrow \frac{1}{\sqrt{2^n}} \sum_{w=0}^{2^n-1} (-1)^{0\cdot w}|w\rangle = \frac{1}{\sqrt{2^n}} \sum_{w=0}^{2^n-1} |w\rangle = \frac{1}{\sqrt{2^n}} (|0\rangle + |1\rangle + |2\rangle + |3\rangle + \ldots)$$

Makes sense. If $|x\rangle = |1\rangle$ we have

$$H^{\otimes n}|1\rangle \longrightarrow \frac{1}{\sqrt{2^n}} \sum_{w=0}^{2^n-1} (-1)^{1\cdot w}|w\rangle = \frac{1}{\sqrt{2^n}} (|0\rangle - |1\rangle + |2\rangle - |3\rangle + \ldots)$$

Also makes sense. (You can verify the formula for yourself.)

The Deutsch-Jozsa algorithm



So the state after the Hadamard gates is

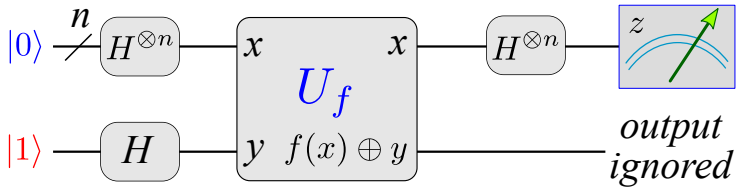
$$|\psi_{out}\rangle = \frac{1}{2^n}\sum_{x=0}^{2^n-1}(-1)^{f(x)}\sum_{w=0}^{2^n-1}(-1)^{x\cdot w}|w\rangle = \frac{1}{2^n}\sum_{w=0}^{2^n-1}\left(\sum_{x=0}^{2^n-1}(-1)^{f(x)}(-1)^{x\cdot w}\right)|w\rangle$$

A pretty horrible state, one must admit.

The Deutsch-Jozsa algorithm

So the state after the Hadamard gates is

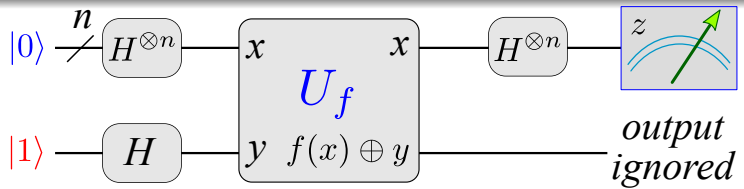
$$|\psi_{out}\rangle = \frac{1}{2^n}\sum_{w=0}^{2^n-1}\left(\sum_{x=0}^{2^n-1}(-1)^{f(x)}(-1)^{x\cdot w}\right)|w\rangle$$



We are **interested only in the $w = 0$ case** we thus have the probability to measure $|w = 0\rangle$,

$$p_0 = |\langle 0|\psi_{out}\rangle|^2 = \frac{1}{2^{2n}}\left|\sum_{x=0}^{2^n-1}(-1)^{f(x)}\right|^2$$

The Deutsch-Jozsa algorithm



Now we have this probability:

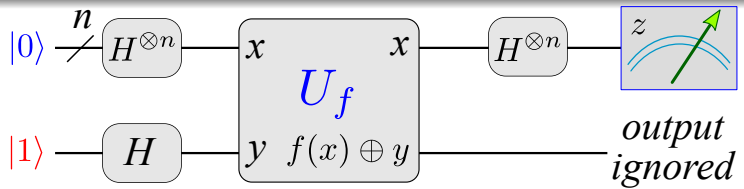
$$p_0 = \frac{1}{2^{2n}} \left| \sum_{x=0}^{2^n-1} (-1)^{f(x)} \right|^2$$

Say $f(x)$ is **constant**. And let's say $\boxed{f(x) = 0}$. Then

$$\sum_{x=0}^{2^n-1} (-1)^{f(x)} = \sum_{x=0}^{2^n-1} (-1)^0 = 1 + 1 + \ldots = 2^n$$

so $p_0 = \frac{1}{2^{2n}} |2^n|^2 = \frac{1}{2^{2n}} 2^{2n}$ and we have $\boxed{p_0 = 1}$.

The Deutsch-Jozsa algorithm



Now we have this probability:

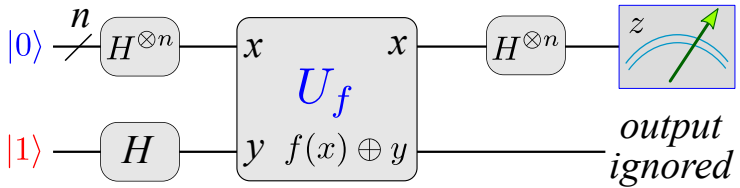
$$p_0 = \frac{1}{2^{2n}} \left| \sum_{x=0}^{2^n-1} (-1)^{f(x)} \right|^2$$

Say $f(x)$ is **constant**. And let's say $\boxed{f(x) = 1}$. Then

$$\sum_{x=0}^{2^n-1} (-1)^{f(x)} = \sum_{x=0}^{2^n-1} (-1)^1 = -2^n$$

so mod-squared it is 2^{2n} and divided by 2^{2n} we have $\boxed{p_0 = 1}$.

The Deutsch-Jozsa algorithm



Now we have this probability:

$$p_0 = \frac{1}{2^{2n}} \left| \sum_{x=0}^{2^n-1} (-1)^{f(x)} \right|^2$$

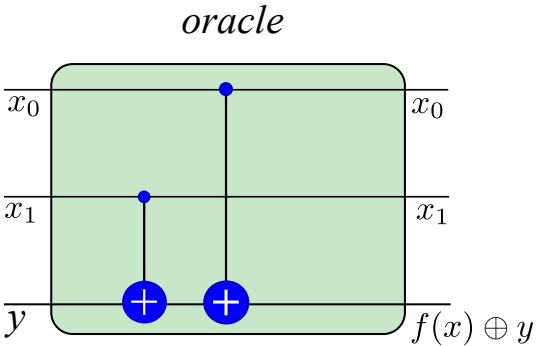
Say $f(x)$ is **balanced**. Then

$$\sum_{x=0}^{2^n-1} (-1)^{f(x)} = 1 + (-1) + 1 + (-1) + \ldots = 0$$

so mod-squared it is still 0 and we have $\boxed{p_0 = 0}$.

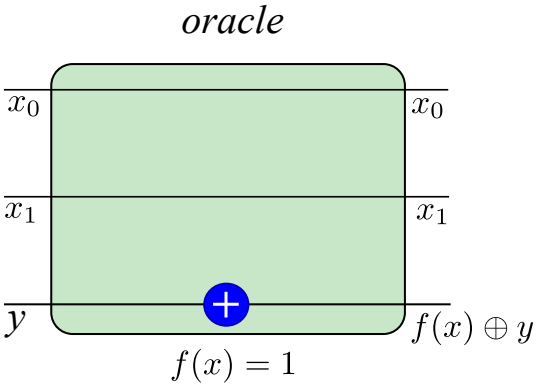
The Deutsch-Jozsa oracle

Implementing a balanced oracle:

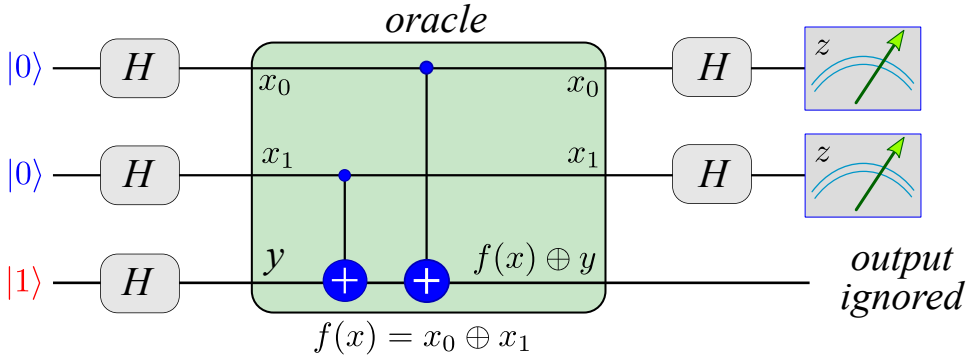


The Deutsch-Jozsa oracle

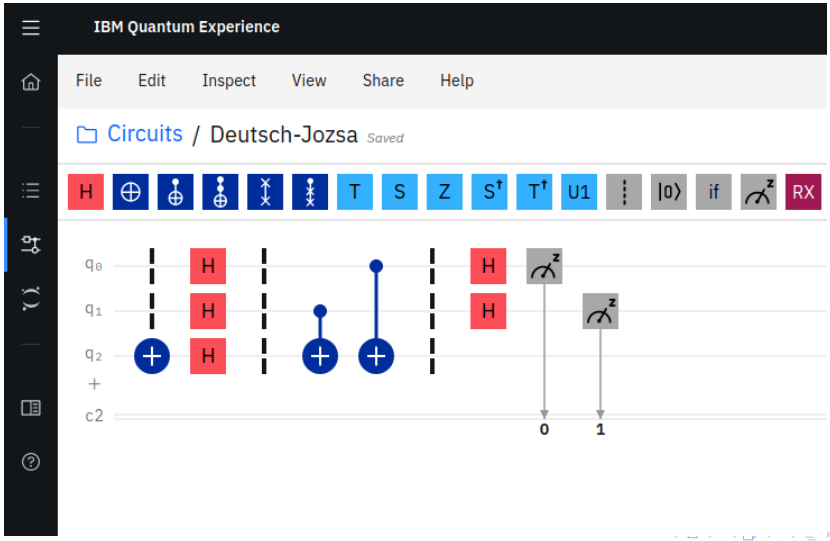
Implementing a constant oracle:



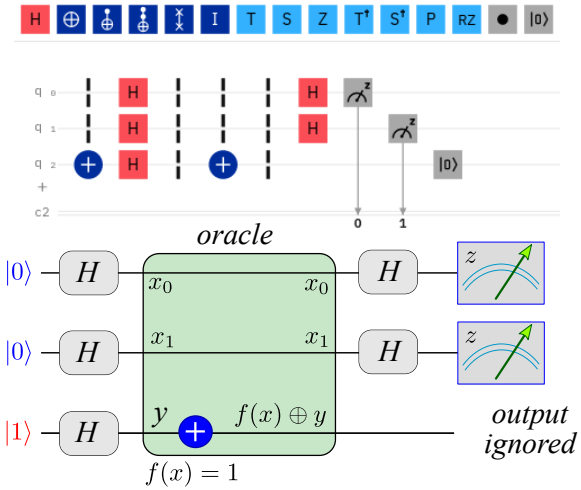
The Deutsch-Jozsa 2 qubit implementation - balanced oracle



The Deutsch-Jozsa 2 qubit implementation - balanced oracle



The Deutsch-Jozsa 2 qubit implementation - constant oracle



The Deutsch-Jozsa 2 qubit implementation - constant oracle

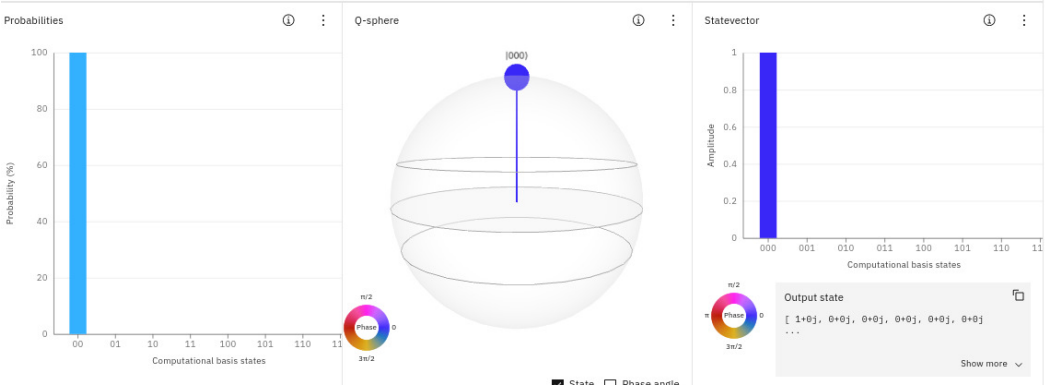
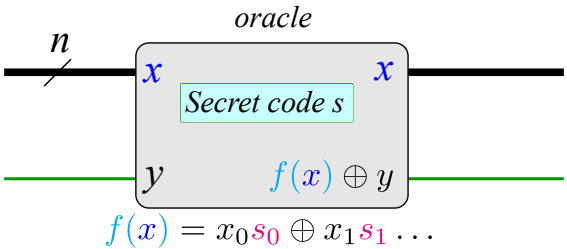


Table of Contents

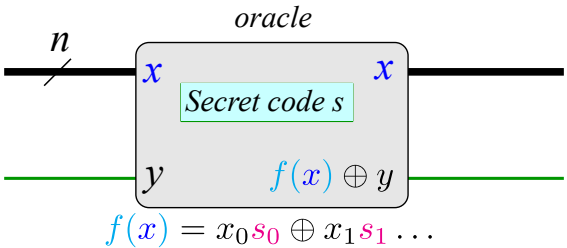
The Bernstein-Vazirani algorithm

An oracle (again) implements a function $f : \{0,1\}^n \to \{0,1\}$.

$$f(x) = x \cdot s = x_0 s_0 + x_1 s_1 + \ldots x_{n-1} s_{n-1} \text{ mod } 2$$



The Bernstein-Vazirani algorithm



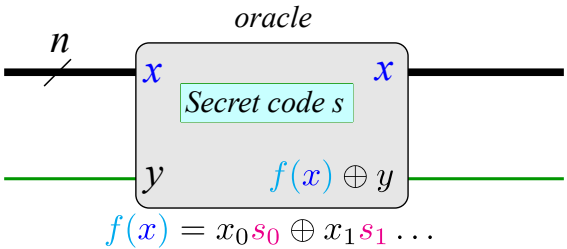
Question:

How many queries are necessary to find out the "secret" code s?

Classical solution

n queries

The Bernstein-Vazirani algorithm



Question:

How many queries are necessary to find out the "secret" code s?

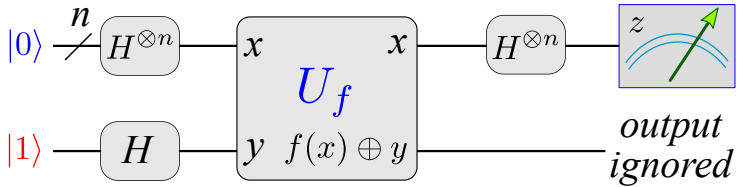
Quantum solution

1 query

The Bernstein-Vazirani algorithm

Similar to the Deutsch-Jozsa algorithm, the state after the Hadamard gates is:

$$|\psi_{out}\rangle = \frac{1}{2^n}\sum_{x=0}^{2^n-1}\sum_{w=0}^{2^n-1}(-1)^{f(x)}(-1)^{x\cdot w}|w\rangle = \frac{1}{2^n}\sum_{w=0}^{2^n-1}\left(\sum_{x=0}^{2^n-1}(-1)^{f(x)}(-1)^{x\cdot w}\right)|w\rangle$$

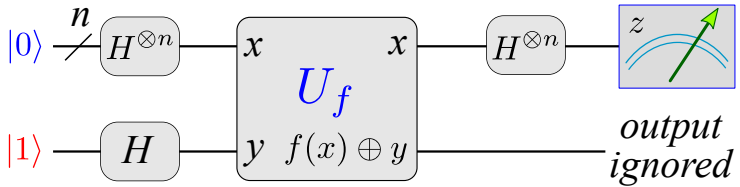


(We've seen this in the previous slides.)

The Bernstein-Vazirani algorithm

So the state after the (last) Hadamard gates is:

$$|\psi_{out}\rangle = \frac{1}{2^n}\sum_{w=0}^{2^n-1}\left(\sum_{x=0}^{2^n-1}(-1)^{f(x)}(-1)^{x\cdot w}\right)|w\rangle$$



However this time $f(x) = x \cdot s$, thus

$$|\psi_{out}\rangle = \frac{1}{2^n}\sum_{w=0}^{2^n-1}\left(\sum_{x=0}^{2^n-1}(-1)^{(s+w)\cdot x}\right)|w\rangle = \frac{1}{2^n}\sum_{w=0}^{2^n-1}\left(\sum_{x=0}^{2^n-1}(-1)^{(s\oplus w)\cdot x}\right)|w\rangle$$

The Bernstein-Vazirani algorithm

So we have

$$|\psi_{out}\rangle = \frac{1}{2^n}\sum_{w=0}^{2^n-1}\left(\sum_{x=0}^{2^n-1}(-1)^{(s\oplus w)\cdot x}\right)|w\rangle$$

Now if $s = w$ (i. e. $s_0 = w_0,\ s_1 = w_1 \ldots$) then (due to constructive interference)

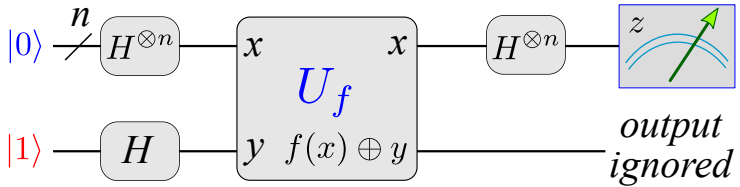
$$\sum_{x=0}^{2^n-1}(-1)^{(s\oplus w)\cdot x} = \sum_{x=0}^{2^n-1}(-1)^0 = 1 + 1 + \ldots = 2^n$$

If $s \neq w$ then we have (destructive interference)

$$\sum_{x=0}^{2^n-1}(-1)^{(s\oplus w)\cdot x} = \sum_{x=0}^{2^n-1}(-1)^{1\cdot x} = 1 - 1 + 1 - 1 + \ldots = 0$$

The Bernstein-Vazirani algorithm

$$|\psi_{out}\rangle = \frac{1}{2^n}\sum_{w=0}^{2^n-1}\left(\sum_{x=0}^{2^n-1}(-1)^{(s\oplus w)\cdot x}\right)|w\rangle = |s\rangle$$

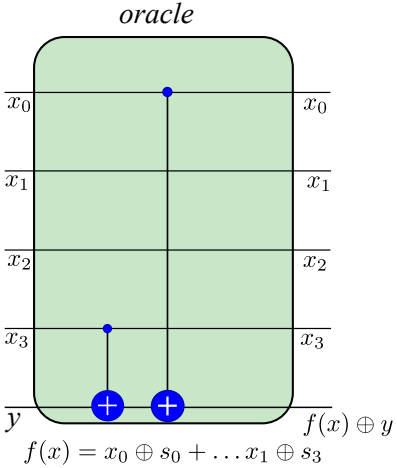


Yes, deterministic

In 1 query.

The Bernstein-Vazirani oracle

Implementing a 4-qubit oracle:



Here the "secret" code is $s = 1001$.

The Bernstein-Vazirani algorithm - 4 qubit

Implementing a 4-qubit version:

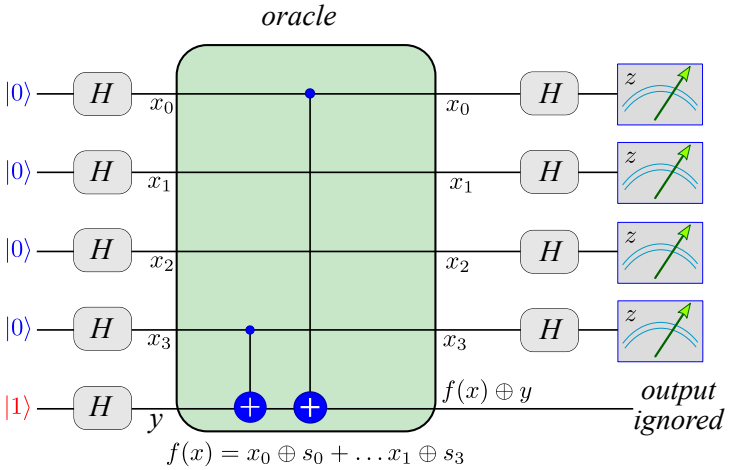


Table of Contents

The Grover search algorithm

Suppose you have an unstructured database. One element is "different" and we want to find it. The database has $N = 2^n$ entries.

How many queries are needed

to find this "different" element?

Classical solution

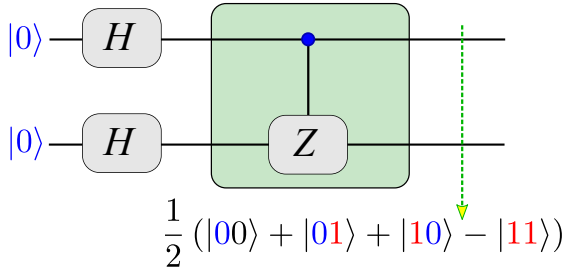
You need $\mathcal{O}(N)$ queries.

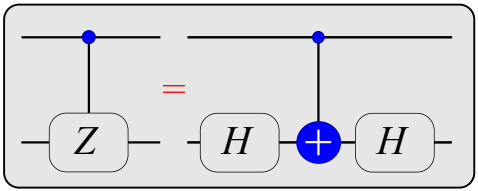
Quantum solution - Grovers's algorithm

You need $\mathcal{O}(\sqrt{N})$ queries.

The Grover oracle

First thing first. Create one "marked" element. Simple two-qubit example:

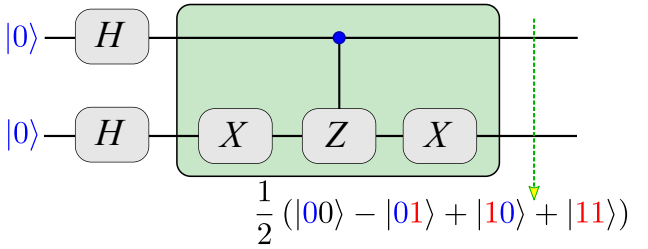




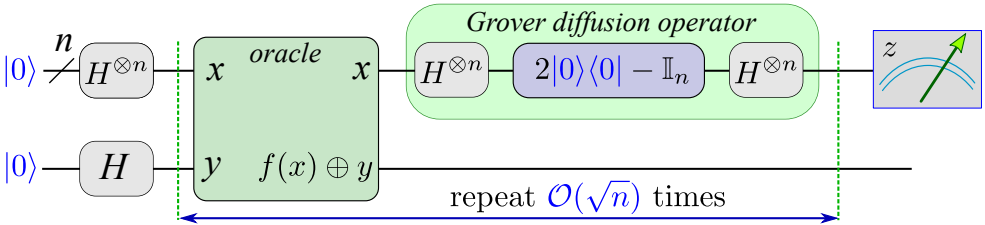
The different element is obviously $|11\rangle$.

The Grover oracle

One can choose which element is marked by the Grover oracle:



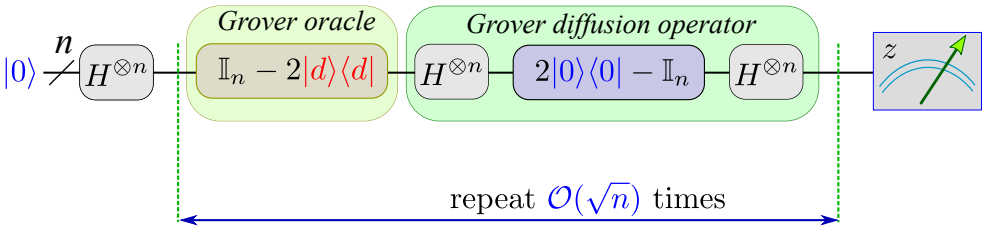
The Grover algorithm



The Grover algorithm

I claim that the action of the oracle can be replaced by the unitary operator:

$$\hat{U}_d = \mathbb{I}^n - 2|d\rangle\langle d|$$



Recall, we had a bunch of qubits, all "identical" except one, the "marked" qubit.

The Grover algorithm

Let's call $|s\rangle$ the equal superposition of all states

$$|s\rangle = H^{\otimes n}|0\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle$$

i. e. the quantum state after the n Hadamard gates. (Lower qubit – the ancilla – not included.)
Now if we apply the Grover oracle

$$\hat{U}_d = \mathbb{I}^n - 2|d\rangle\langle d|$$

on this state, we have (detailed calculation for $\langle d|s\rangle$ on the next slide)

$$\hat{U}_d|s\rangle = \mathbb{I}^n|s\rangle - 2|d\rangle\langle d|s\rangle = |s\rangle - 2\underbrace{\langle d|s\rangle}_{=\frac{1}{\sqrt{2^n}}}|d\rangle = |s\rangle - \frac{2}{\sqrt{2^n}}|d\rangle$$

The Grover algorithm

We used the fact that the scalar product is

$$\langle d|s\rangle = \frac{1}{\sqrt{2^n}}\langle d|\sum_{x=0}^{2^n-1}|x\rangle = \frac{1}{\sqrt{2^n}}\left(\underbrace{\langle d||0\rangle}_{=0} + \ldots + \underbrace{\langle d||d\rangle}_{=1} + \ldots + \underbrace{\langle d||2^n-1\rangle}_{=0}\right) = \frac{1}{\sqrt{2^n}}$$

So we have

$$\hat{U}_d|s\rangle = |s\rangle - \frac{2}{\sqrt{2^n}}|d\rangle = \frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}|x\rangle - \frac{2}{\sqrt{2^n}}|d\rangle$$

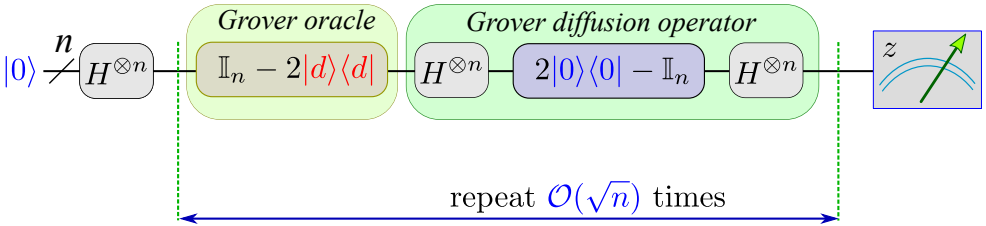
and indeed, this no other than

$$\hat{U}_d\,|s\rangle = \frac{1}{\sqrt{2^n}}\left(|0\rangle + |1\rangle + \ldots \underbrace{-|d\rangle}_{\text{“marked”}} + \ldots + |2^n-2\rangle + |2^n-1\rangle\right)$$

The Grover diffusion operator

Now the so-called Grover diffusion operator is another unitary operator defined as

$$\hat{U}_s = 2|s\rangle\langle s| - \mathbb{I}^n$$



We apply $\hat{U}_s$ to the quantum state $\hat{U}_d|s\rangle$ (computed already) and have

$$\hat{U}_s\hat{U}_d|s\rangle = (2|s\rangle\langle s| - \mathbb{I}^n)\left(|s\rangle - 2\frac{1}{\sqrt{2^n}}|d\rangle\right)$$

The Grover algorithm

So applying the Grover diffusion operator leads to

$$\hat{U}_s\hat{U}_d\,|s\rangle = 2|s\rangle \underbrace{\langle s|\,|s\rangle}_{=1} - \underbrace{\mathbb{I}^n\,|s\rangle}_{=|s\rangle} - 4\frac{1}{\sqrt{2^n}}|s\rangle \underbrace{\langle s|d\rangle}_{=\frac{1}{\sqrt{2^n}}} + 2\frac{1}{\sqrt{2^n}}|d\rangle$$

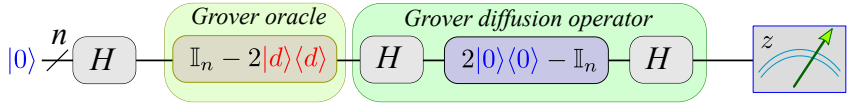
therefore we have

$$\hat{U}_s\hat{U}_d\,|s\rangle = |s\rangle - 4\frac{1}{\sqrt{2^n}}\frac{1}{\sqrt{2^n}}|s\rangle + \frac{2}{\sqrt{2^n}}|d\rangle = \left(1-\frac{4}{2^n}\right)|s\rangle + \frac{2}{\sqrt{2^n}}|d\rangle$$

For 2 qubits one step is enough: $n = 2$ implies

$$\hat{U}_s\hat{U}_d\,|s\rangle = \left(1-\frac{4}{2^n}\right)|s\rangle + \frac{2}{\sqrt{2^n}}|d\rangle = |d\rangle$$

The Grover algorithm - 2 qubits



Start with $|0\rangle$. Democratically, create an equal superposition

$$|s\rangle = H^{\otimes n}|0\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle = \frac{1}{\sqrt{4}} \sum_{x=0}^{3} |x\rangle$$

Apply now the Grover oracle. If will "mark" the element that is different;

$$\hat{U}_d|s\rangle = |s\rangle - \frac{2}{\sqrt{2^n}}|d\rangle = \frac{1}{2} \sum_{x=0}^{2^n-1} |x\rangle - |d\rangle$$

Apply now the Grover diffusion operator:

$$\hat{U}_s\hat{U}_d\,|s\rangle = \left(1 - \frac{4}{2^n}\right)|s\rangle + \frac{2}{\sqrt{2^n}}|d\rangle = |d\rangle$$

The Grover algorithm

Now if $n > 2$ we need to continue. We recall that we had the state:

$$\hat{U}_s\hat{U}_d\left|s\right\rangle = \left(1 - \frac{4}{2^n}\right)\left|s\right\rangle + \frac{2}{\sqrt{2^n}}\left|d\right\rangle = \frac{N-4}{N}\left|s\right\rangle + \frac{2}{\sqrt{N}}\left|d\right\rangle$$

and we use the simpler notation $N = 2^n$ from now on. Applying again the Grover oracle we have

$$\hat{U}_d\hat{U}_s\hat{U}_d\left|s\right\rangle = \left(\mathbb{I}^n - 2\left|d\right\rangle\left\langle d\right|\right)\left(\frac{N-4}{N}\left|s\right\rangle + \frac{2}{\sqrt{N}}\left|d\right\rangle\right)$$

and therefore

$$\hat{U}_d\hat{U}_s\hat{U}_d\left|s\right\rangle = \frac{N-4}{N}\left|s\right\rangle - 2\frac{N-4}{N}\underbrace{\left\langle d\right|\left.s\right\rangle}_{=\frac{1}{\sqrt{N}}}\left|d\right\rangle + \frac{2}{\sqrt{N}}\left(\mathbb{I}^n - 2\left|d\right\rangle\left\langle d\right|\right)\left|d\right\rangle$$

and finally

$$\hat{U}_d\hat{U}_s\hat{U}_d\left|s\right\rangle = \frac{N-4}{N}\left|s\right\rangle - 4\frac{N-2}{N}\frac{1}{\sqrt{N}}\left|d\right\rangle$$

The Grover algorithm

We have

$$\hat{U}_d\hat{U}_s\hat{U}_d\left|s\right\rangle = \frac{N-4}{N}\left|s\right\rangle - 4\frac{N-2}{N}\frac{1}{\sqrt{N}}\left|d\right\rangle$$

and now we apply again the Grover diffusion operator

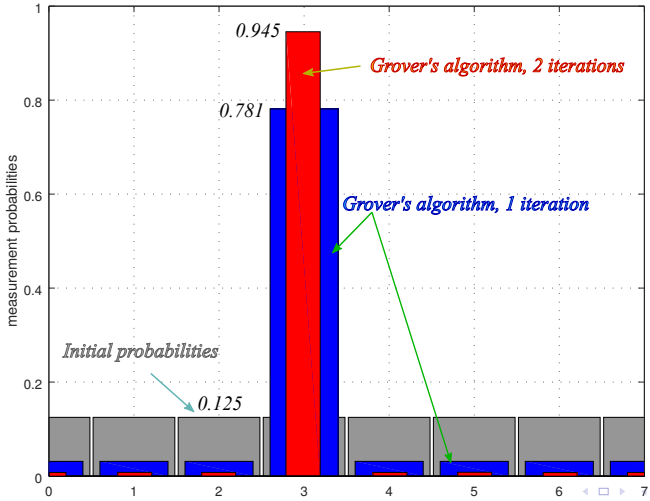
$$\hat{U}_s\hat{U}_d\hat{U}_s\hat{U}_d\left|s\right\rangle = \left(2\left|s\right\rangle\left\langle s\right| - \mathbb{I}^n\right)\left(\frac{N-4}{N}\left|s\right\rangle - 4\frac{N-2}{N}\frac{1}{\sqrt{N}}\left|d\right\rangle\right)$$

thus we end up with

$$\hat{U}_s\hat{U}_d\hat{U}_s\hat{U}_d\left|s\right\rangle = \frac{N^2-12N+16}{N^2}\left|s\right\rangle + \frac{4\left(N-2\right)}{N}\frac{1}{\sqrt{N}}\left|d\right\rangle$$

Grover's algorithm example - 3 qubits

We have 3 qubits (thus $2^3 = 8$ states) and the marked element is $d = 3$.



The Grover algorithm

We can add one more step (if needed). We start form

$$\hat{U}_s\hat{U}_d\hat{U}_s\hat{U}_d\,|s\rangle = \frac{N^2-12N+16}{N^2}|s\rangle + 4\frac{N-2}{N}\frac{1}{\sqrt{N}}|d\rangle$$

and applying the Grover oracle we have

$$\hat{U}_d\hat{U}_s\hat{U}_d\hat{U}_s\hat{U}_d\,|s\rangle = (\mathbb{I}^n - 2|d\rangle\langle d|)\left(\frac{N^2-12N+16}{N^2}|s\rangle + 4\frac{N-2}{N}\frac{1}{\sqrt{N}}|d\rangle\right)$$

or, expanded

$$\hat{U}_d\hat{U}_s\hat{U}_d\hat{U}_s\hat{U}_d\,|s\rangle = \frac{N^2-12N+16}{N^2}|s\rangle + \frac{-6N^2+32N-32}{N^2\sqrt{N}}|d\rangle$$

The Grover algorithm

We now apply the Grover diffusion operator

$$\hat{U}_s\hat{U}_d\hat{U}_s\hat{U}_d\hat{U}_s\hat{U}_d\,|s\rangle = \frac{N^2-12N+16}{N^2}|s\rangle + 2\frac{-6N^2+32N-32}{N^3}|s\rangle + \frac{6N^2-32N+32}{N^2\sqrt{N}}|d\rangle$$

and finally we end up with

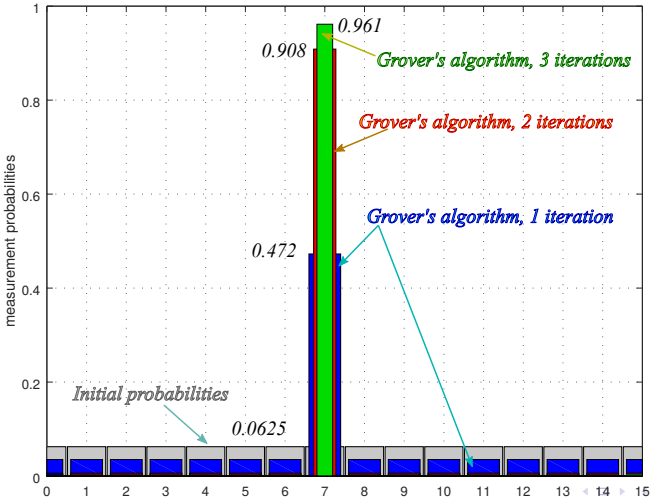
$$\hat{U}_s\hat{U}_d\hat{U}_s\hat{U}_d\hat{U}_s\hat{U}_d\,|s\rangle = \frac{N^3-24N^2+80N-64}{N^3}|s\rangle + \frac{6N^2-32N+32}{N^2\sqrt{N}}|d\rangle$$

Remark:

What Grover's algorithm does is to "amplify" the marked element's amplitude until it gets closer and closer to 1.

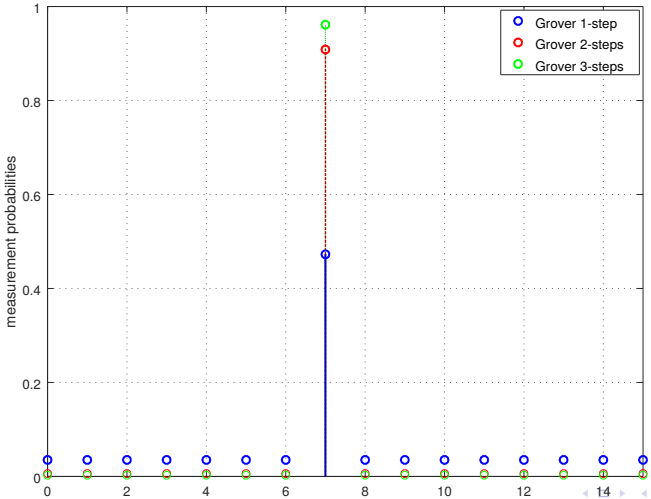
The Grover algorithm

We have 4 qubits (thus $2^4 = 16$ states) and the marked element is $d = 7$.



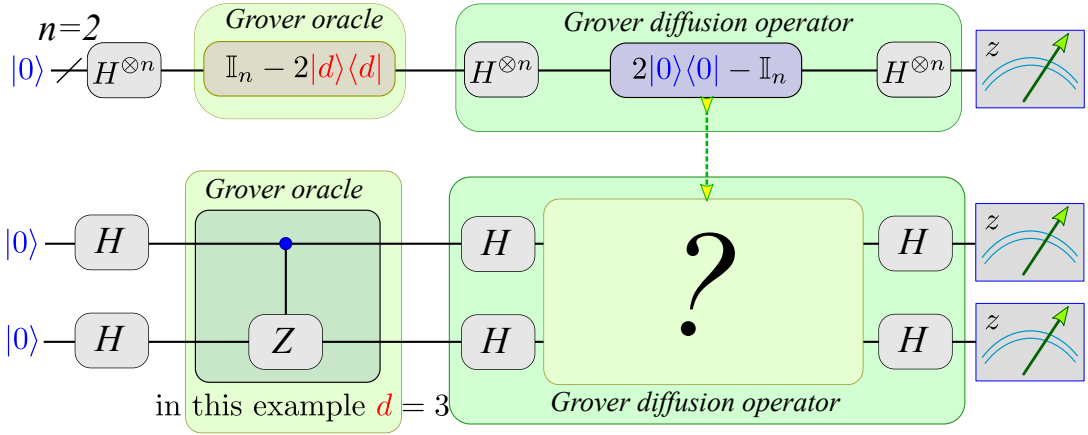
The Grover algorithm

4 qubits, 1, 2 and 3 Grover algorithm iterations



The Grover algorithm - 2 qubits

From theory to implementation:



The Grover algorithm - 2 qubits

Recall the Grover diffusion operator,

$$\hat{U}_s = 2|s\rangle\langle s| - \mathbb{I}^n = H^{\otimes n}|0\rangle\langle 0|H^{\otimes n} - \mathbb{I}^n = H^{\otimes n}(|0\rangle\langle 0| - \mathbb{I}^n)H^{\otimes n}$$

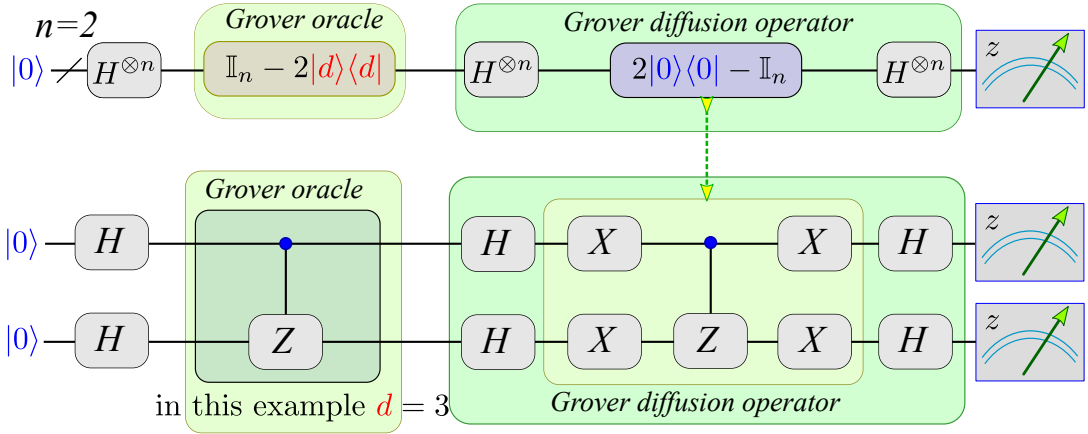
Now what is $|0\rangle\langle 0| - \mathbb{I}^n$ doing in a two-qubit case? Recall, now we have $n = 0, 1, 2, 3$.
In other words, we input $|00\rangle$, we write $|0\rangle$, we input $|01\rangle$, we write $|1\rangle$, input $|10\rangle$, we write $|2\rangle$ and finally input $|11\rangle$, we write $|3\rangle$.
It is easy to check that

$$\begin{cases} |00\rangle = |0\rangle \to \quad |0\rangle = \quad |00\rangle \\ |01\rangle = |1\rangle \to -|1\rangle = -|01\rangle \\ |10\rangle = |2\rangle \to -|2\rangle = -|10\rangle \\ |11\rangle = |3\rangle \to -|3\rangle = -|11\rangle \end{cases}$$

Now this would be a control-z if we would invert all zeros to ones and viceversa. X-gates will do the job.

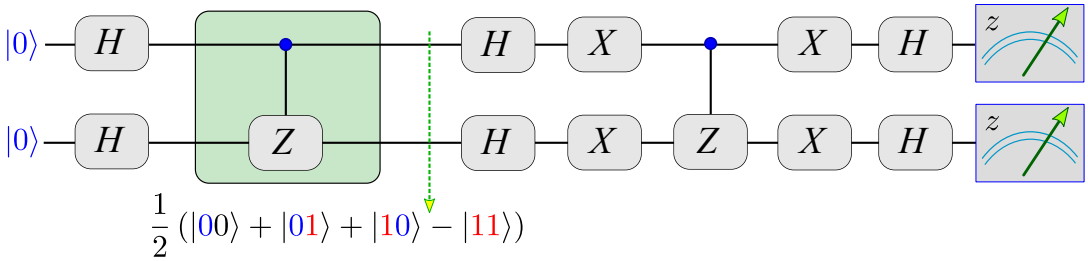
The Grover algorithm - 2 qubits

Finally, from theory to implementation:



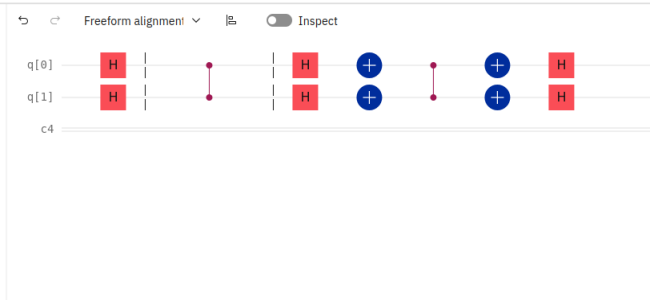
The Grover algorithm

Assuming to have a control-Z gate, a two qubit Grover's algorithm implementation is straightforward:



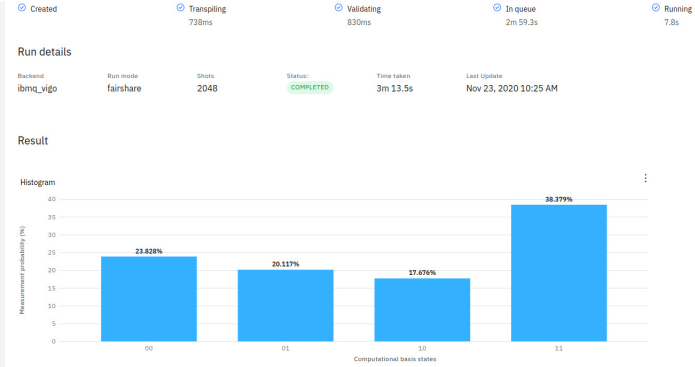
The Grover algorithm

Assuming to have a control-Z gate, a two qubit Grover's algorithm implementation is straightforward:

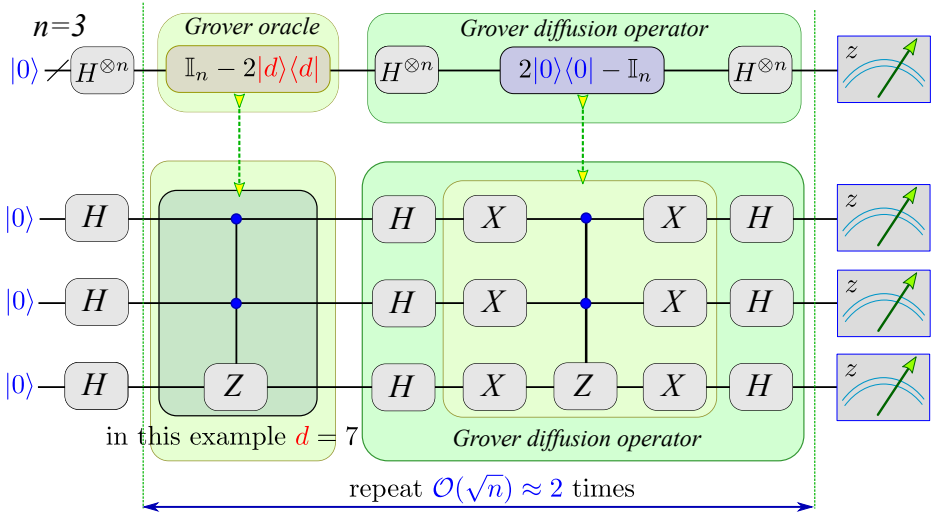


The Grover algorithm

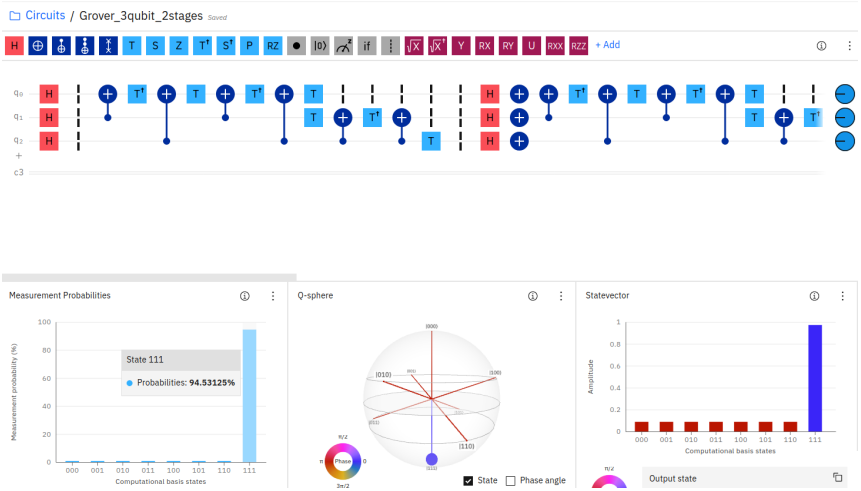
Running a 2-qubit Grover's algorithm on `ibmq_vigo`:



The Grover algorithm - 3 qubits

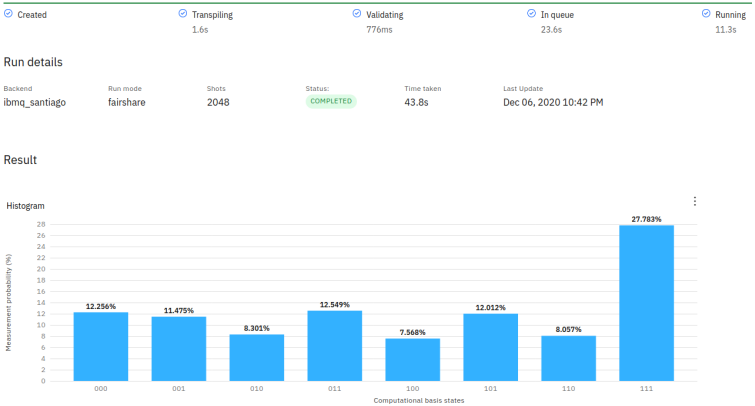


The Grover algorithm - 3 qubits



The Grover algorithm - 3 qubits, 1 stage

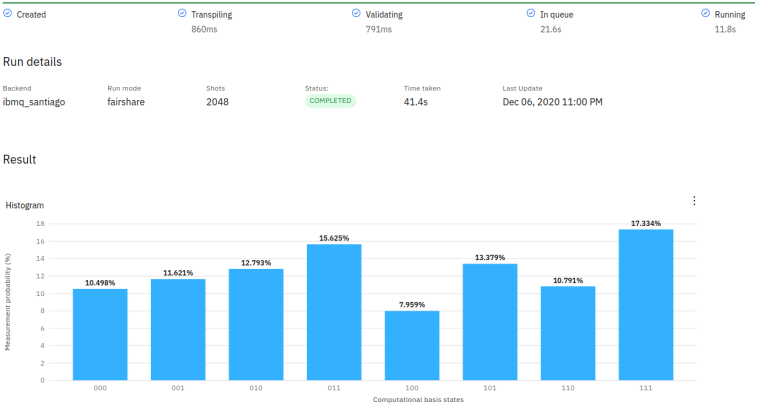
Running a 3-qubit and 1-stage Grover's algorithm on `ibmq_santiago`:



Conclusion: not bad!

The Grover algorithm - 3 qubits, 2 stages

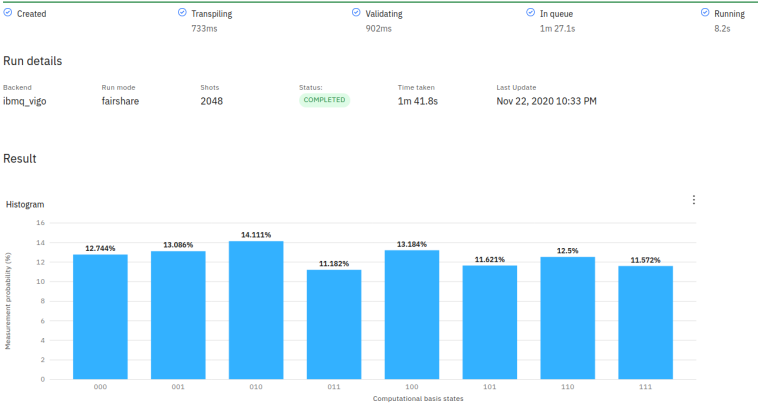
Running a 3-qubit and 2-stage Grover's algorithm on `ibmq_santiago`:



Conclusion: not great (but not terrible)!

The Grover algorithm - 3 qubits, 2 stages

Running a 3-qubit and 2-stage Grover's algorithm on `ibmq_vigo`:



Conclusion: a disaster!

Table of Contents

The quantum Fourier transform

We define the quantum Fourier transform of a state $|x\rangle$ by

$$\hat{U}_{\text{QFT}} |k\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} e^{i\frac{2\pi}{N}xk} |x\rangle$$

and sometimes we denote (recall $N = 2^n$ where n is the number of qubits)

$$\omega_N = e^{i\frac{2\pi}{N}}$$

so the quantum Fourier transform can be written as

$$\hat{U}_{\text{QFT}} |k\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} \omega_N^{xk} |x\rangle$$

Recall: ω_N is the N-th root of unity, *i. e.* the solution to $z^N = 1$, with $z \in \mathbb{C}$.

The quantum Fourier transform

The unitary operator $\hat{U}_{\mathsf{QFT}}$ is given by

$$\hat{U}_{\mathsf{QFT}} = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} e^{i\frac{2\pi}{N}xy} \left|x\right\rangle \left\langle y\right|$$

and it is obviously a $N \times N$ matrix where $N = 2^n$, n being the number of qubits considered. Expanded:

$$\hat{U}_{\mathsf{QFT}} = \frac{1}{\sqrt{N}} \begin{pmatrix} e^{i\frac{2\pi}{N}0\cdot 0} & e^{i\frac{2\pi}{N}1\cdot 0} & \dots & e^{i\frac{2\pi}{N}(N-1)\cdot 0} \\ e^{i\frac{2\pi}{N}0\cdot 1} & e^{i\frac{2\pi}{N}1\cdot 1} & \dots & e^{i\frac{2\pi}{N}(N-1)\cdot 1} \\ \vdots & \vdots & \ddots & \vdots \\ e^{i\frac{2\pi}{N}0\cdot (N-1)} & e^{i\frac{2\pi}{N}1\cdot (N-1)} & \dots & e^{i\frac{2\pi}{N}(N-1)\cdot (N-1)} \end{pmatrix}$$

The quantum Fourier transform

The QFT matrix becomes much clearer if we employ again the N-th root of unity

$$\omega_N = e^{i\frac{2\pi}{N}}$$

Indeed, we have:

$$\hat{U}_{\text{QFT}} = \frac{1}{\sqrt{N}} \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_N & \omega_N^2 & \dots & \omega_N^{N-1} \\ 1 & \omega_N^2 & \omega_N^4 & \dots & \omega_N^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_N^{N-1} & \omega_N^{2(N-1)} & \dots & \omega_N^{(N-1)(N-1)} \end{pmatrix}$$

The quantum Fourier transform - 1 qubit

The one qubit version the QFT is quite nice (recall $N = 2^n = 2$). Indeed:

$$\hat{U}_{\text{QFT}} = \frac{1}{\sqrt{2}} \begin{pmatrix} \omega_2^0 & \omega_2^0 \\ \omega_2^0 & \omega_2^1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & e^{i\pi 1 \cdot 1} \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

so we have:

$$\hat{U}_{\text{QFT}} |0\rangle = \frac{1}{\sqrt{2}} \sum_{x=0}^{1} e^{i\frac{2\pi}{2} x \cdot 0} |x\rangle = \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) = |+\rangle$$

and

$$\hat{U}_{\text{QFT}} |1\rangle = \frac{1}{\sqrt{2}} \sum_{x=0}^{1} e^{i\frac{2\pi}{2} x} |x\rangle = \frac{1}{\sqrt{2}} \left(|0\rangle + e^{i\pi} |1\rangle\right) = |-\rangle$$

Question: did this transformation remind you of something?

The quantum Fourier transform – 2 qubits

The two qubit version the QFT is ($N = 2^2 = 4$)

$$\hat{U}_{\text{QFT}} = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & e^{i\frac{\pi}{2}1\cdot 1} & e^{i\frac{\pi}{2}2\cdot 1} & e^{i\frac{\pi}{2}3\cdot 1} \\ 1 & e^{i\frac{\pi}{2}1\cdot 2} & e^{i\frac{\pi}{2}2\cdot 2} & e^{i\frac{\pi}{2}3\cdot 2} \\ 1 & e^{i\frac{\pi}{2}1\cdot 3} & e^{i\frac{\pi}{2}2\cdot 3} & e^{i\frac{\pi}{2}3\cdot 3} \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix}$$

So we have

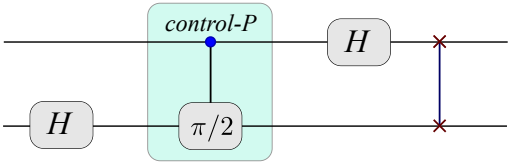
$$\hat{U}_{\text{QFT}} |0\rangle = \frac{1}{\sqrt{2^2}} \sum_{x=0}^{1} e^{i\frac{2\pi}{4}x\cdot 0} |x\rangle = \frac{1}{2}\left(|0\rangle + |1\rangle + |2\rangle + |3\rangle\right)$$

and

$$\hat{U}_{\text{QFT}} |1\rangle = \frac{1}{2} \sum_{x=0}^{1} e^{i\frac{2\pi}{2}x} |x\rangle = \frac{1}{2}\left(|0\rangle + i\,|1\rangle - |2\rangle - i\,|3\rangle\right)$$

The quantum Fourier transform – 2 qubit implementation

Actual circuit implementation:

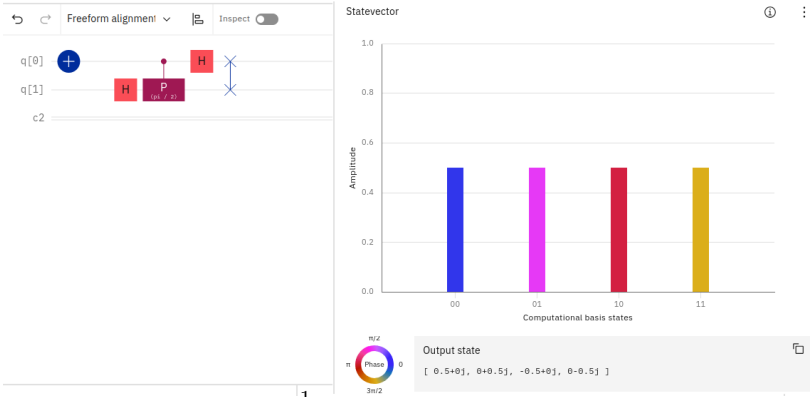


Matrix expression for control-P:

$$\hat{P}_{\text{ctrl}} = \mathbb{I}_2 \otimes \underbrace{|0\rangle\langle 0|}_{=\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}} + \underbrace{P(\phi)}_{=\begin{pmatrix} 0 & 0 \\ 0 & e^{i\phi} \end{pmatrix}} \otimes \underbrace{|1\rangle\langle 1|}_{=\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}} = \begin{pmatrix} \mathbb{I}_2 & \mathbb{O}_2 \\ \mathbb{O}_2 & P(\phi) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\phi} \end{pmatrix}$$

and we need the angle $\phi = \frac{2\pi}{2^n} = \frac{\pi}{2}$ (recall $n = 2$).

The quantum Fourier transform – 2 qubit IBM implementation

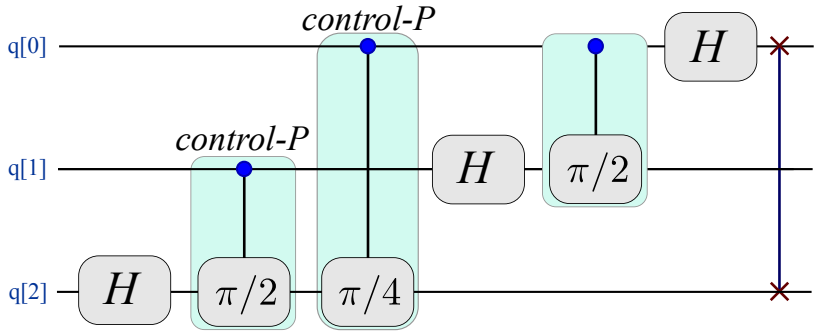


Recall: $\hat{U}_{\text{QFT}}\,|1\rangle = \dfrac{1}{2}\sum_{x=0}^{1} e^{i\frac{2\pi}{2}x}\,|x\rangle = \dfrac{1}{2}\left(|0\rangle + i\,|1\rangle - |2\rangle - i\,|3\rangle\right)$

How about the QFT for 3 qubits?

The quantum Fourier transform – 3 qubit implementation

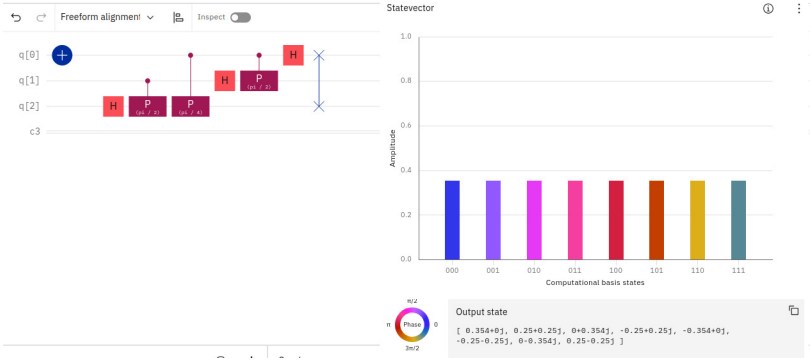
Actual circuit implementation:



The generalization is now straightforward.

The quantum Fourier transform – 3 qubit implementation

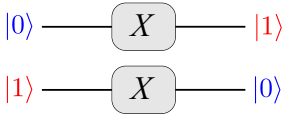
IBM Quantum platform implementation:

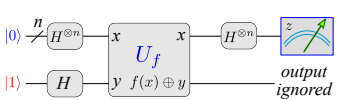


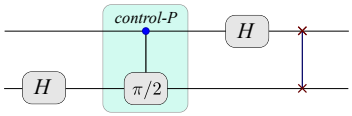
That's it for today.

Thank you for your attention!

Questions are welcome.

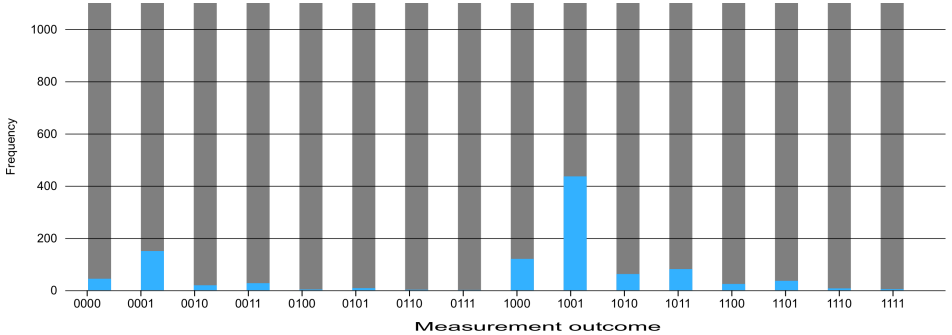






The Bernstein-Vazirani algorithm 4 qubits - IBM QE

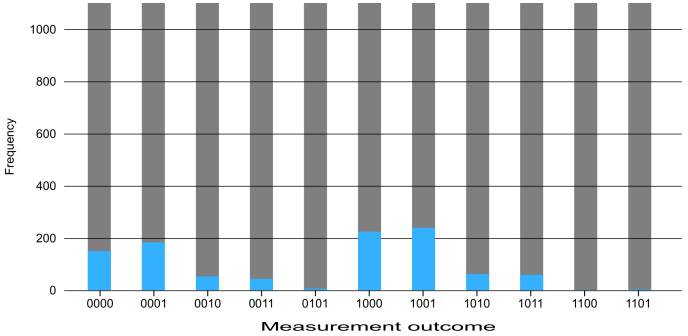
Running the code on a real quantum computer `ibmq_bogota` (16/01/2022):



Success rate 42.5%.

The Bernstein-Vazirani algorithm 4 qubits - IBM QE

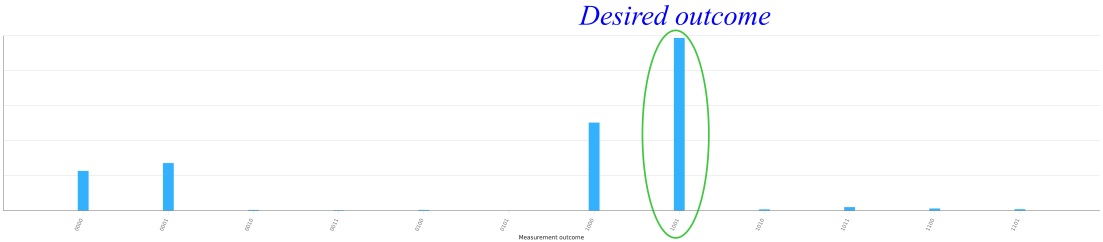
Running the code on a real quantum computer `ibmq_quito` (17/01/2022):



Success rate 23.34%. Quite low.

The Bernstein-Vazirani algorithm 4 qubits - IBM QE

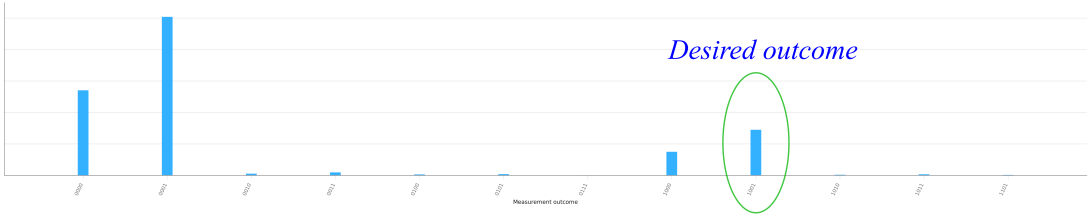
Running the code on a real quantum computer `ibmq_lima` (21/11/2022):



Success rate 48.19%. Quite OK.

The Bernstein-Vazirani algorithm 4 qubits - IBM QE

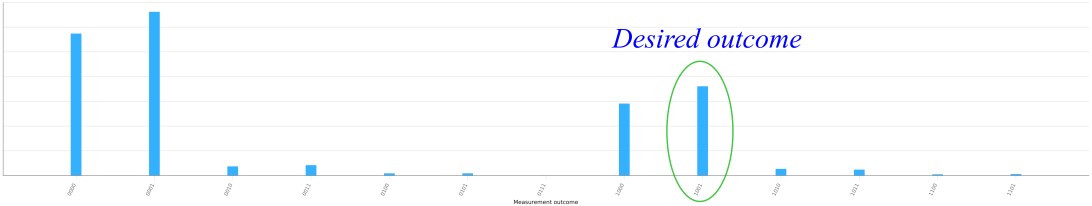
Running the code on a real quantum computer: `ibmq_manila` (21/11/2022):



Success rate 14.16%.

The Bernstein-Vazirani algorithm 4 qubits - IBM QE

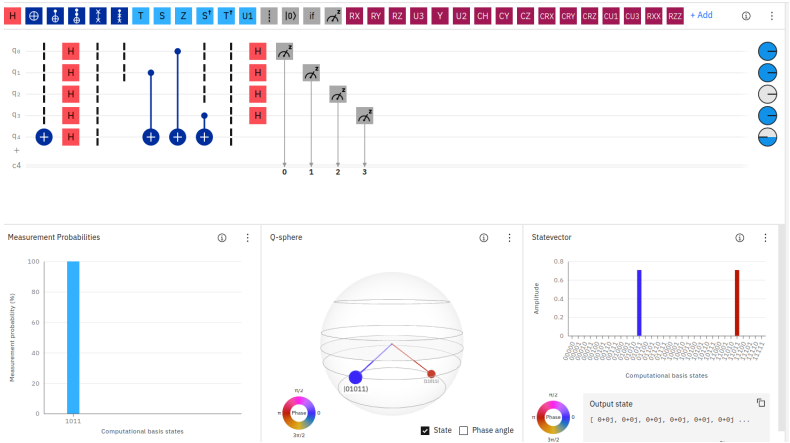
Running the code on a real quantum computer: `ibmq_quito` (21/11/2022):



Success rate 17.62%.

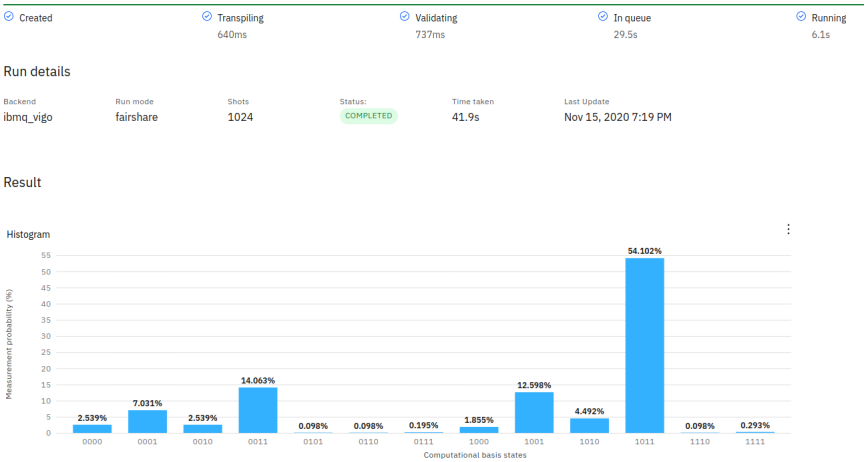
The Bernstein-Vazirani algorithm - IBM QE

Implementing a 4-qubit version: (Ok, I changed the oracle. Here $s = 1011$.)



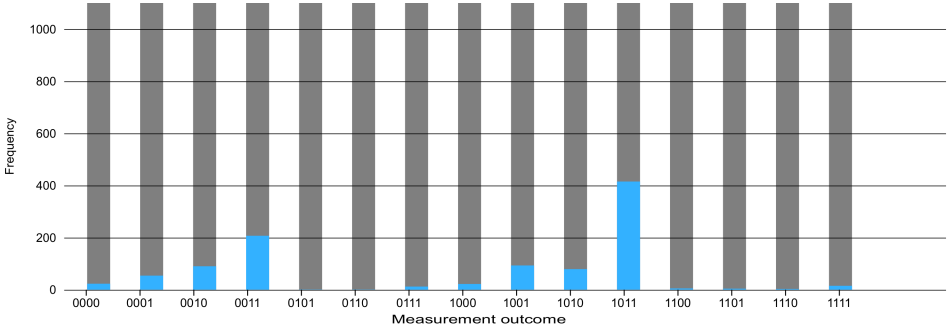
The Bernstein-Vazirani algorithm 4 qubits - IBM QE

Running the code on a real quantum computer `ibmq_vigo` (15/11/2020):



The Bernstein-Vazirani algorithm 4 qubits - IBM QE

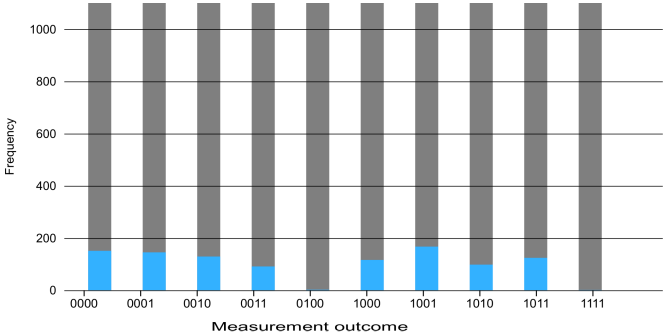
Running the code on a real quantum computer `ibmq_bogota` (16/01/2022):



Success probability 40.5%.

The Bernstein-Vazirani algorithm 4 qubits - IBM QE

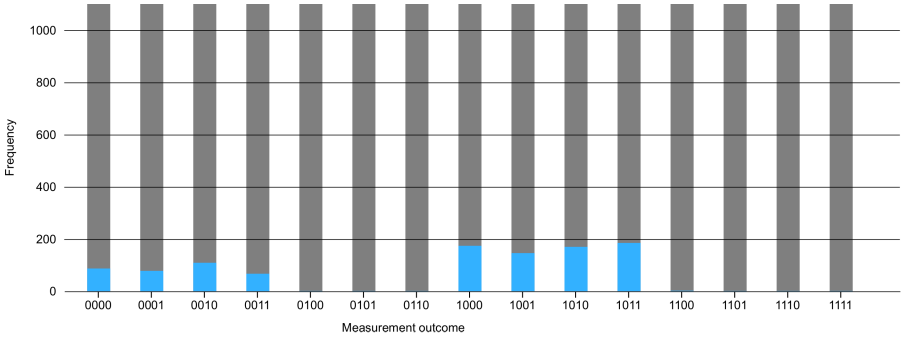
Running the code on a real quantum computer `ibmq_lima` (16/01/2022):



Success rate 16.3%. (This quantum computer has some really noisy qubits.)

The Bernstein-Vazirani algorithm 4 qubits - IBM QE

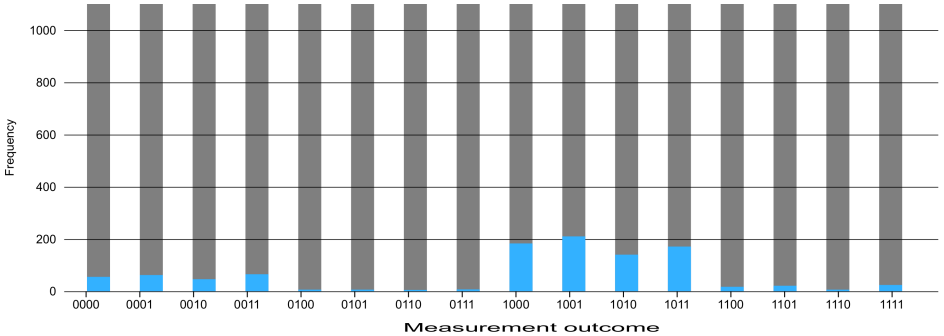
Running the code on a real quantum computer `ibmq_belem` (16/01/2022):



Success rate 18%. (This quantum computer has some really noisy qubits.)

The Bernstein-Vazirani algorithm 4 qubits - IBM QE

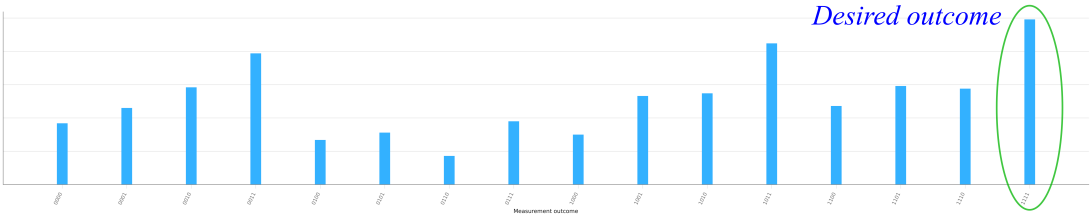
Running the code on a real quantum computer `ibmq_manila` (16/01/2022):



Success rate 16.7%.

The Bernstein-Vazirani algorithm 4 qubits - IBM QE

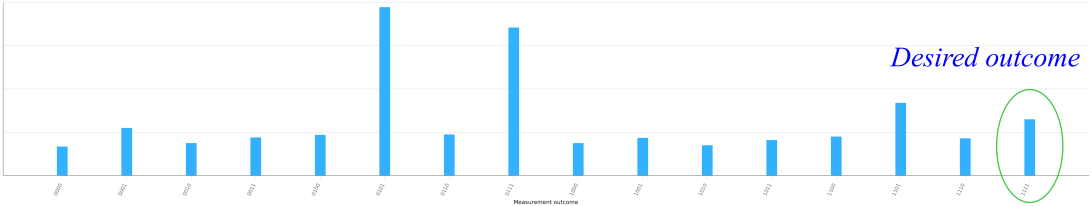
Running the code on a real quantum computer: `ibmq_belem` (21/11/2022):



Success rate 12.1%.

The Bernstein-Vazirani algorithm 4 qubits - IBM QE

Running the code on a real quantum computer: `ibmq_lima` (21/11/2022):



Lousy.