

THIS IS CS4048!

GCR:wzj3vua

QUANTUM ALGORITHMS

CONTENT

Chapter 7 - Quantum Algorithms

Chapter 8 - Shor's Algorithm

Chapter 9 - Grover's Algorithm

From the book Quantum Computing - A gentle introduction

ORACLE

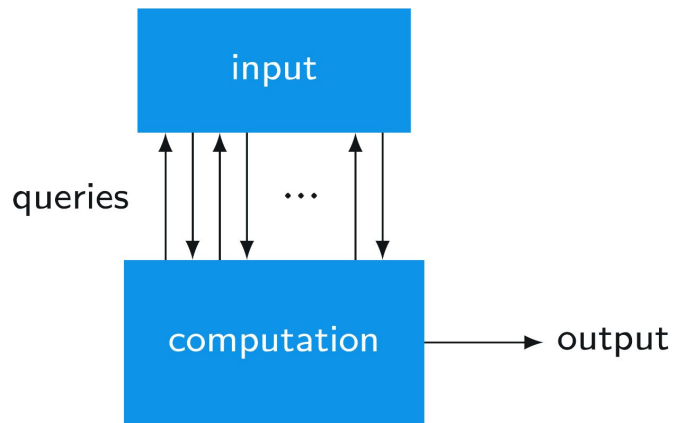
What comes in your mind?

ORACLE

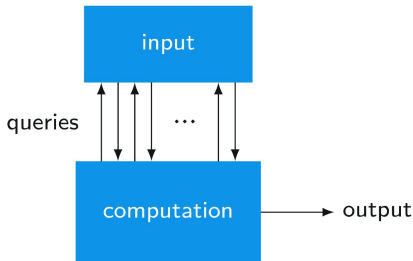
ORACLE

A device that we cannot look inside to which we can pass queries and which return answers.

Works like a black box



ORACLE

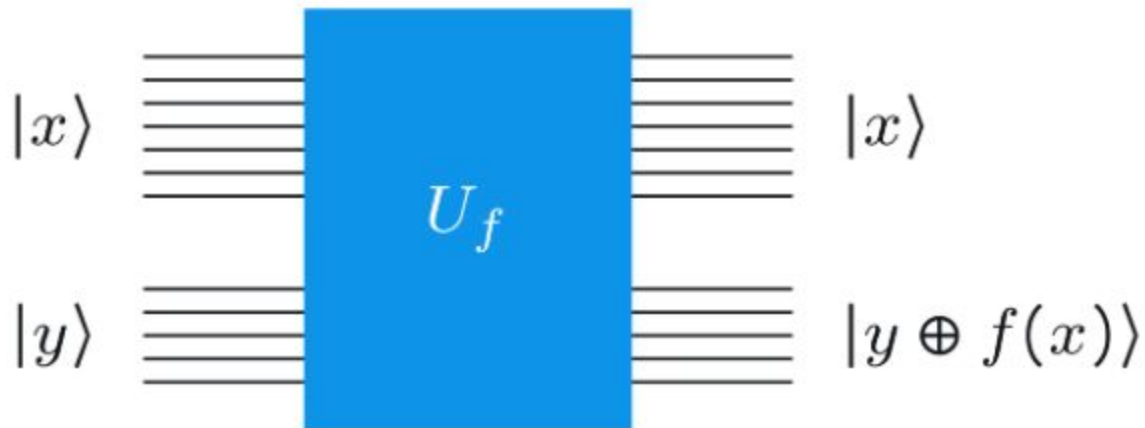


In the query model of computation, on the other hand, the entire input is not provided to the computation. Rather, the input is made available in the form of a function, which the computation accesses by making queries.

EXAMPLES OF QUERY PROBLEMS

- **OR.** The input function takes the form $f : \Sigma^n \rightarrow \Sigma$ (so $m = 1$ for this problem). The task is to output 1 if there exists a string $x \in \Sigma^n$ for which $f(x) = 1$, and to output 0 if there is no such string. If we think about the function f as representing a sequence of 2^n bits to which we have random access, the problem is to compute the OR of these bits.
- **Parity.** The input function again takes the form $f : \Sigma^n \rightarrow \Sigma$. The task is to determine whether the number of strings $x \in \Sigma^n$ for which $f(x) = 1$ is *even* or *odd*. To be precise, the required output is 0 if the set $\{x \in \Sigma^n : f(x) = 1\}$ has an even number of elements and 1 if it has an odd number of elements. If we think about the function f as representing a sequence of 2^n bits to which we have random access, the problem is to compute the parity (or XOR) of these bits.
- **Minimum.** The input function takes the form $f : \Sigma^n \rightarrow \Sigma^m$ for any choices of positive integers n and m . The required output is the string $y \in \{f(x) : x \in \Sigma^n\}$ that comes first in the lexicographic (i.e., dictionary) ordering of Σ^m . If we think about the function f as representing a sequence of 2^n integers encoded as strings of length m in binary notation to which we have random access, the problem is to compute the minimum of these integers.

QUERY GATES



DEUTCH'S ALGORITHM

Deutsch's algorithm solves the parity problem (described above) for the special case that $n=1$.

In the context of quantum computing this problem is sometimes referred to as Deutsch's problem

To be precise, the input is represented by a function $f:\Sigma\rightarrow\Sigma$ from one bit to one bit.

DEUTCH'S ALGORITHM

a	$f_1(a)$
0	0
1	0

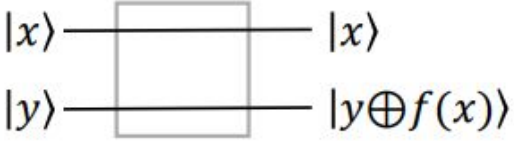
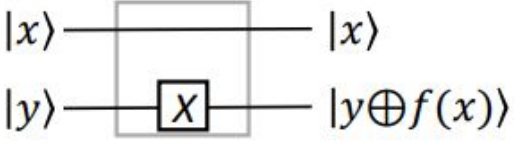
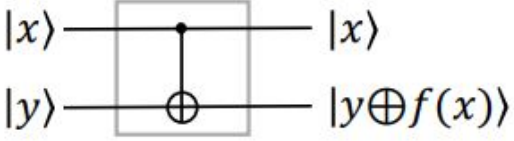
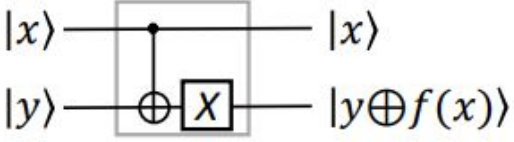
a	$f_2(a)$
0	0
1	1

a	$f_3(a)$
0	1
1	0

a	$f_4(a)$
0	1
1	1

Constant and balanced function on a single bit

If the input, x , is a single bit (as is the output), then we have four possible functions:

Function	x	$f(x)$	Type	Unitary
$f(x) = 0$	0 1	0 0	Constant	
$f(x) = 1$	0 1	1 1	Constant	
$f(x) = x$	0 1	0 1	Balanced	
$f(x) = x \oplus 1$	0 1	1 0	Balanced	

DEUTSCH PROBLEM

Deutsch's Problem :

Given a Boolean function $f : \mathbb{Z}_2 \rightarrow \mathbb{Z}_2$, determine whether f is constant.

DEUTSCH PROBLEM

Developed by David Deutsch in 1985

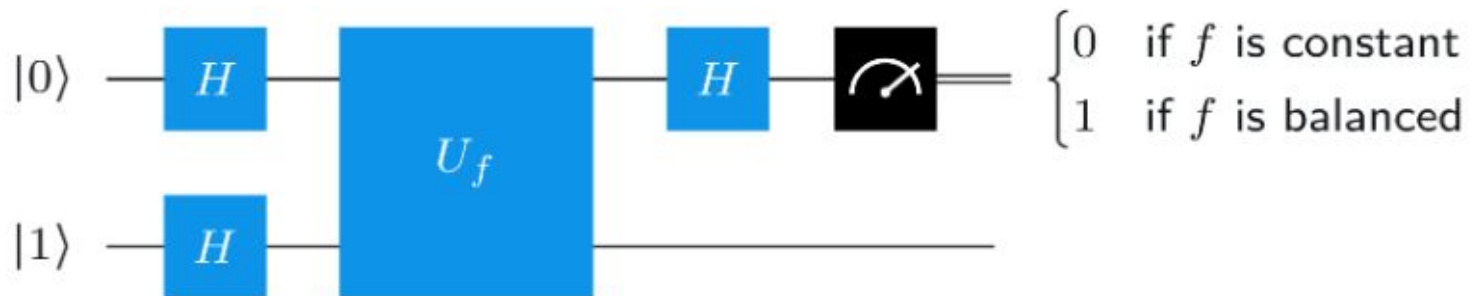
First result that showed that quantum computation could outperform classical computation

The problem Deutsch's algorithm solves is a black box problem

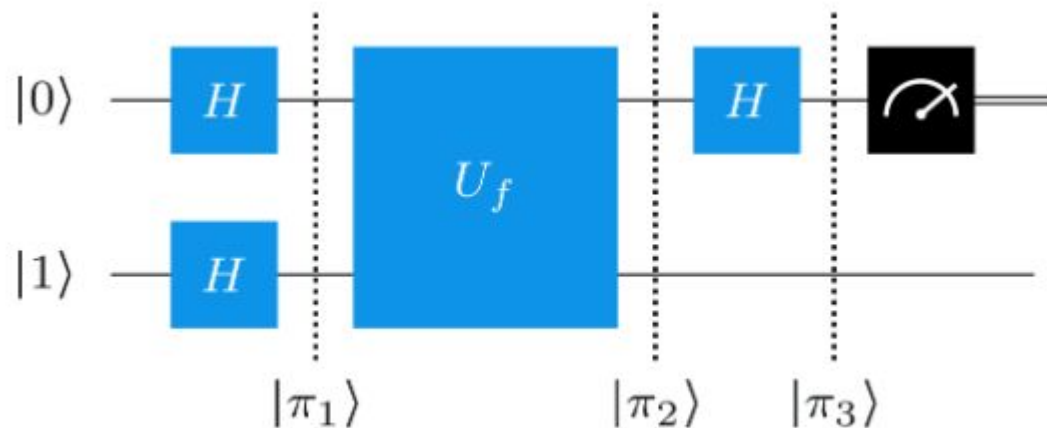
Deutsch showed that his quantum algorithm has better query complexity than any possible classical algorithm: it can solve the problem with fewer calls to the black box than is possible classically

DEUTSCH PROBLEM

Deutsch's algorithm solves Deutsch's problem using a single query, therefore providing a quantifiable advantage of quantum over classical computations. This may be a modest advantage – one query as opposed to two.

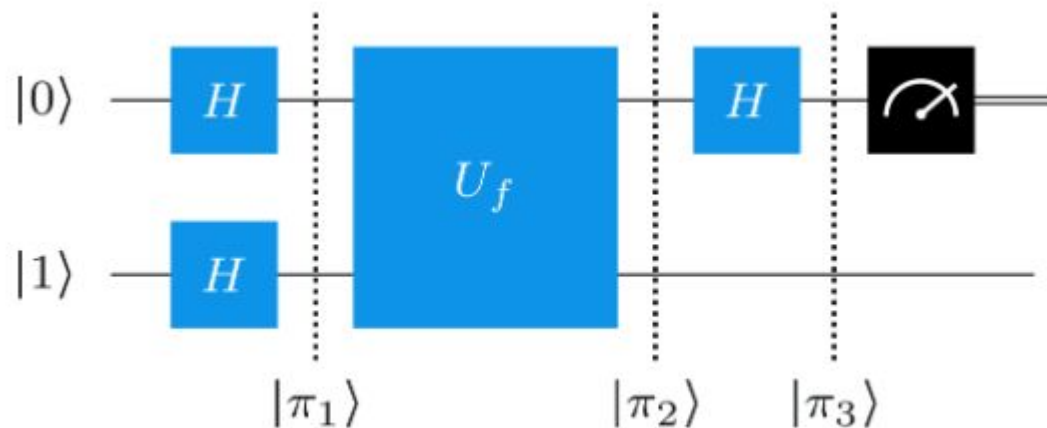


DEUTSCH PROBLEM



$$|\pi_1\rangle = |-\rangle|+\rangle = \frac{1}{2}(|0\rangle - |1\rangle)|0\rangle + \frac{1}{2}(|0\rangle - |1\rangle)|1\rangle.$$

DEUTSCH PROBLEM



$$|\pi_2\rangle = \frac{1}{2}(|0 \oplus f(0)\rangle - |1 \oplus f(0)\rangle)|0\rangle + \frac{1}{2}(|0 \oplus f(1)\rangle - |1 \oplus f(1)\rangle)|1\rangle.$$

DEUTSCH PROBLEM

$$|\pi_2\rangle = \frac{1}{2}(|0 \oplus f(0)\rangle - |1 \oplus f(0)\rangle)|0\rangle + \frac{1}{2}(|0 \oplus f(1)\rangle - |1 \oplus f(1)\rangle)|1\rangle.$$

$$|0 \oplus a\rangle - |1 \oplus a\rangle = (-1)^a(|0\rangle - |1\rangle)$$

$$|0 \oplus 0\rangle - |1 \oplus 0\rangle = |0\rangle - |1\rangle = (-1)^0(|0\rangle - |1\rangle)$$

$$|0 \oplus 1\rangle - |1 \oplus 1\rangle = |1\rangle - |0\rangle = (-1)^1(|0\rangle - |1\rangle)$$

$$\begin{aligned} |\pi_2\rangle &= \frac{1}{2}(-1)^{f(0)}(|0\rangle - |1\rangle)|0\rangle + \frac{1}{2}(-1)^{f(1)}(|0\rangle - |1\rangle)|1\rangle \\ &= |-\rangle \left(\frac{(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle}{\sqrt{2}} \right). \end{aligned}$$

DEUTSCH PROBLEM

$$\begin{aligned} |\pi_2\rangle &= \frac{1}{2}(-1)^{f(0)}(|0\rangle - |1\rangle)|0\rangle + \frac{1}{2}(-1)^{f(1)}(|0\rangle - |1\rangle)|1\rangle \\ &= |-\rangle \left(\frac{(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle}{\sqrt{2}} \right). \end{aligned}$$

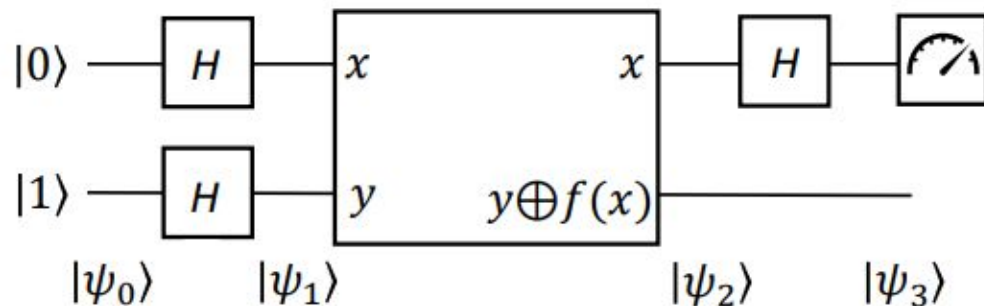
DEUTSCH PROBLEM

$$\begin{aligned} |\pi_2\rangle &= (-1)^{f(0)}|-\rangle \left(\frac{|0\rangle + (-1)^{f(0)\oplus f(1)}|1\rangle}{\sqrt{2}} \right) \\ &= \begin{cases} (-1)^{f(0)}|-\rangle|+\rangle & \text{if } f(0) \oplus f(1) = 0 \\ (-1)^{f(0)}|-\rangle|-\rangle & \text{if } f(0) \oplus f(1) = 1. \end{cases} \end{aligned}$$

DEUTSCH PROBLEM

$$|\pi_3\rangle = \begin{cases} (-1)^{f(0)}|-\rangle|0\rangle & \text{if } f(0) \oplus f(1) = 0 \\ (-1)^{f(0)}|-\rangle|1\rangle & \text{if } f(0) \oplus f(1) = 1, \end{cases}$$

Deutsch's algorithm (1)



Initially we prepare the state:

$$|\psi_0\rangle = |01\rangle$$

Which the initial Hadamard gates put in the superposition state:

$$\begin{aligned} |\psi_1\rangle &= \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right) \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) \\ &= \frac{1}{2} (|00\rangle - |01\rangle + |10\rangle - |11\rangle) \end{aligned}$$

Deutsch's algorithm (2)

$$|\psi_1\rangle = \frac{1}{2} (|00\rangle - |01\rangle + |10\rangle - |11\rangle)$$

Next the unitary is implemented, which sets the second qubit to $y \oplus f(x)$, so we have four options for $|\psi_2\rangle$:

$$\begin{array}{ll} f(x) = 0 & |\psi_2\rangle = \frac{1}{2} (|00\rangle - |01\rangle + |10\rangle - |11\rangle) \\ f(x) = 1 & |\psi_2\rangle = \frac{1}{2} (|01\rangle - |00\rangle + |11\rangle - |10\rangle) \\ f(x) = x & |\psi_2\rangle = \frac{1}{2} (|00\rangle - |01\rangle + |11\rangle - |10\rangle) \\ f(x) = x \oplus 1 & |\psi_2\rangle = \frac{1}{2} (|01\rangle - |00\rangle + |10\rangle - |11\rangle) \end{array}$$

which factorises as:

$$|\psi_2\rangle = \begin{cases} \pm \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right) \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) & \text{if } f(0) = f(1) \\ \pm \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) & \text{if } f(0) \neq f(1) \end{cases}$$

That is the two balanced cases differ only by an unobservable global phase (and likewise for the two constant cases).

Deutsch's algorithm (3)

$$|\psi_2\rangle = \begin{cases} \pm \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right) \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) & \text{if } f(0) = f(1) \\ \pm \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) & \text{if } f(0) \neq f(1) \end{cases}$$

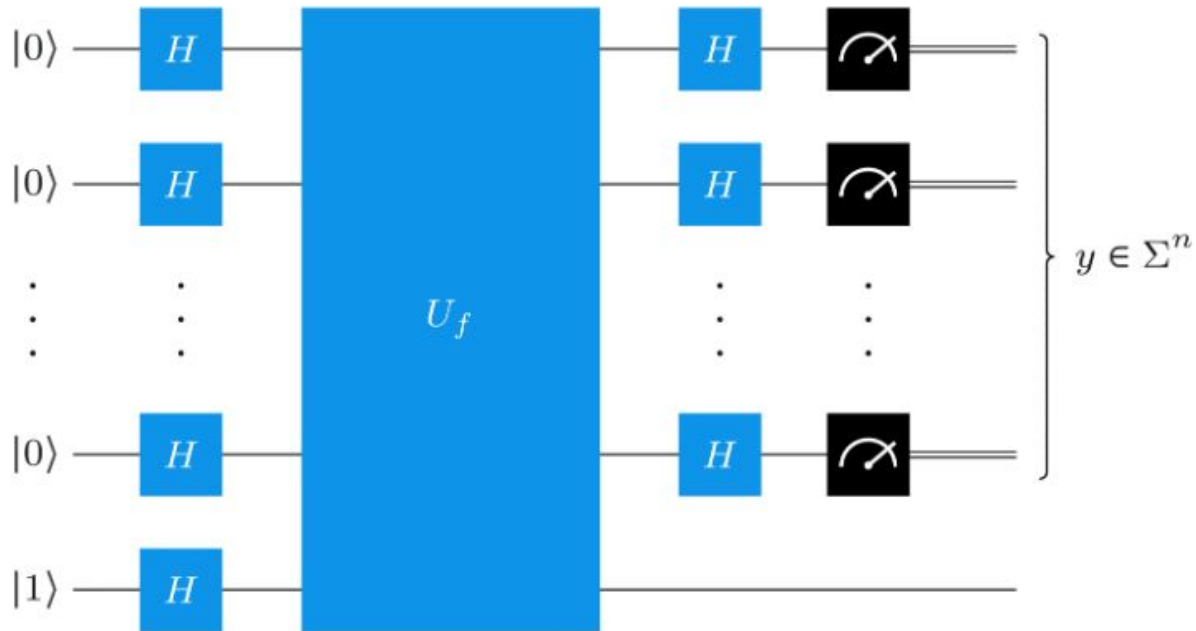
The next step is to use the Hadamard gate to interfere the superposition on the first qubit, which yields:

$$|\psi_3\rangle = \begin{cases} \pm |0\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) & \text{if } f(0) = f(1) \\ \pm |1\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) & \text{if } f(0) \neq f(1) \end{cases}$$

The final step is to measure the first qubit, and **we can see that the outcome will always be 0 if the function is constant, and 1 if balanced.**

We can see that superposition and interference, in some sense, play complementary roles: we prepare a state in superposition, perform some operations, and then use interference to discern some global property of the state.

DEUTSCH JOZSA ALGORITHM



DEUTSCH JOZSA ALGORITHM

$$H = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix},$$

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

$$H|1\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle.$$

$$H|a\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}(-1)^a|1\rangle = \frac{1}{\sqrt{2}} \sum_{b \in \{0,1\}} (-1)^{ab}|b\rangle,$$

DEUTSCH JOZSA ALGORITHM

$$H|a\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}(-1)^a|1\rangle = \frac{1}{\sqrt{2}} \sum_{b \in \{0,1\}} (-1)^{ab}|b\rangle,$$

$$\begin{aligned} H^{\otimes n}|x_{n-1} \cdots x_1 x_0\rangle &= (H|x_{n-1}\rangle) \otimes \cdots \otimes (H|x_0\rangle) \\ &= \left(\frac{1}{\sqrt{2}} \sum_{y_{n-1} \in \Sigma} (-1)^{x_{n-1}y_{n-1}} |y_{n-1}\rangle \right) \otimes \cdots \otimes \left(\frac{1}{\sqrt{2}} \sum_{y_0 \in \Sigma} (-1)^{x_0y_0} |y_0\rangle \right) \\ &= \frac{1}{\sqrt{2^n}} \sum_{y_{n-1} \cdots y_0 \in \Sigma^n} (-1)^{x_{n-1}y_{n-1} + \cdots + x_0y_0} |y_{n-1} \cdots y_0\rangle \end{aligned}$$

$$(H|1\rangle)(H^{\otimes n}|0 \cdots 0\rangle) = |-\rangle \otimes \frac{1}{\sqrt{2^n}} \sum_{x_{n-1} \cdots x_0 \in \Sigma^n} |x_{n-1} \cdots x_0\rangle.$$

DEUTSCH JOZSA ALGORITHM

$$(H|1\rangle)(H^{\otimes n}|0 \cdots 0\rangle) = |-\rangle \otimes \frac{1}{\sqrt{2^n}} \sum_{x_{n-1} \cdots x_0 \in \Sigma^n} |x_{n-1} \cdots x_0\rangle.$$

After Uf

$$|-\rangle \otimes \frac{1}{\sqrt{2^n}} \sum_{x_{n-1} \cdots x_0 \in \Sigma^n} (-1)^{f(x_{n-1} \cdots x_0)} |x_{n-1} \cdots x_0\rangle$$

After second H

$$|-\rangle \otimes \frac{1}{2^n} \sum_{x_{n-1} \cdots x_0 \in \Sigma^n} \sum_{y_{n-1} \cdots y_0 \in \Sigma^n} (-1)^{f(x_{n-1} \cdots x_0) + x_{n-1}y_{n-1} + \cdots + x_0y_0} |y_{n-1} \cdots y_0\rangle.$$

$$\left| \frac{1}{2^n} \sum_{x_{n-1} \cdots x_0 \in \Sigma^n} (-1)^{f(x_{n-1} \cdots x_0)} \right|^2 = \begin{cases} 1 & \text{if } f \text{ is constant} \\ 0 & \text{if } f \text{ is balanced.} \end{cases}$$

DEUTSCH-JOZSA PROBLEM

While the problem it solves is too trivial to be of practical interest, the algorithm contains simple versions of a number of key elements of intrinsically quantum computation, including the use of nonstandard bases and quantum analogs of classical functions applied to superpositions, that will recur in more complex quantum algorithms.

QUERY COMPLEXITY

Classical Query Complexity: $O(2^n)$ since in the worst case, you might have to check every input to determine if the function is balanced or constant.

Quantum Query Complexity: 1 query. The Deutsch-Josza algorithm can solve this problem with a single quantum query, leveraging quantum parallelism and interference to get the answer.

REFERENCES

[https://www.cl.cam.ac.uk/teaching/1920/QuantComp/Quantum Computing Lecture 7.pdf](https://www.cl.cam.ac.uk/teaching/1920/QuantComp/Quantum%20Computing%20Lecture%207.pdf)

<https://www.classiq.io/insights/the-deutsch-jozsa-algorithm-explained>

<https://learning.quantum.ibm.com/course/fundamentals-of-quantum-algorithms/quantum-query-algorithms>

<https://www.youtube.com/watch?v=WYAUh-4K5E0>

SIMON'S PROBLEM

Simon's algorithm is a quantum query algorithm for a problem known as Simon's problem.

Simon's algorithm is significant because it provides an exponential advantage of quantum over classical (including probabilistic) algorithms, and the technique it uses inspired Peter Shor's discovery of an efficient quantum algorithm for factoring integers.

SIMON'S PROBLEM

Simon's problem

Input: a function $f : \Sigma^n \rightarrow \Sigma^m$

Promise: there exists a string $s \in \Sigma^n$ such that $[f(x) = f(y)] \Leftrightarrow [(x = y) \vee (x \oplus s = y)]$ for all $x, y \in \Sigma^n$

Output: the string s

SIMON'S PROBLEM

Simon's algorithm is a quantum query algorithm for a problem known as Simon's problem.

Simon's algorithm is significant because it provides an exponential advantage of quantum over classical (including probabilistic) algorithms, and the technique it uses inspired Peter Shor's discovery of an efficient quantum algorithm for factoring integers.

SIMON'S PROBLEM

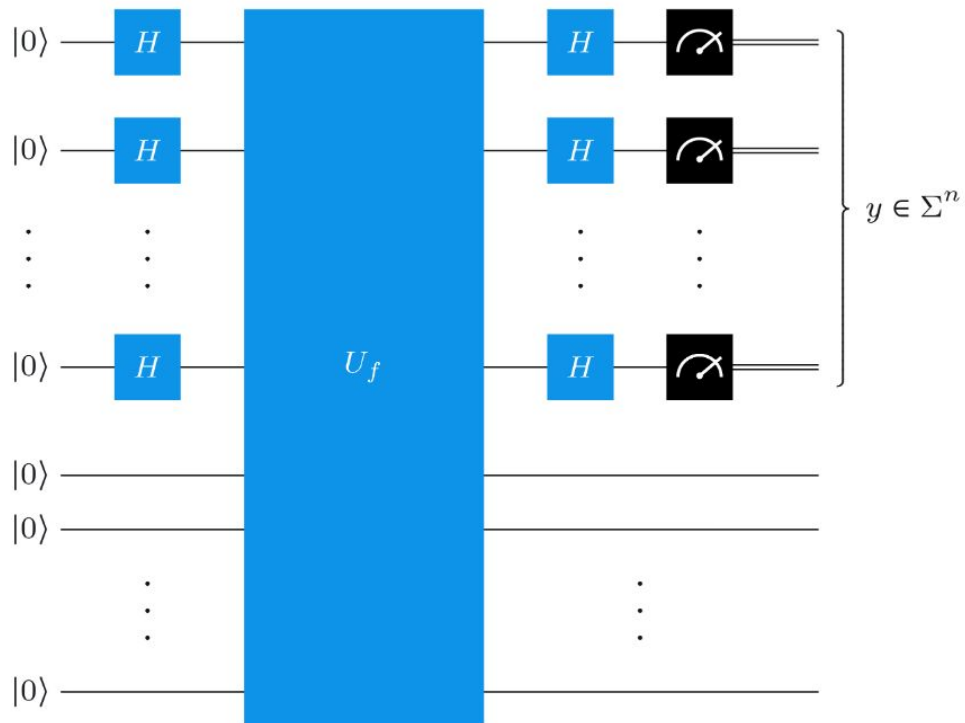
$$f(x) = f(x \oplus a)$$

exclusive or

For example, for $a = 011$, f fulfills this requirement.

$000 \oplus 011 = 011$	$f(000) = 010$	$f(100) = 110$
$f(000) = f(011)$	$f(001) = 101$	$f(101) = 001$
	$f(010) = 101$	$f(110) = 001$
	$f(011) = 010$	$f(111) = 110$

QUANTUM CIRCUIT FOR SIMON'S PROBLEM



SIMON'S PROBLEM

n qubits on the top that are acted upon by Hadamard gates

m qubits on the bottom that go directly into the query gate

It looks very similar to the algorithms we've already discussed in the lesson, but this time there's no phase kickback.

To solve Simon's problem using this circuit will actually require several independent runs of this circuit followed by a classical post-processing step.

SIMON'S PROBLEM - ANALYSIS

After the first layer of Hadamard gates is performed on the top n qubits

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \Sigma^n} |0^m\rangle |x\rangle.$$

SIMON'S PROBLEM - ANALYSIS

When the U_f is performed, the output of the function f is XORed onto the all-zero state of the bottom m qubits, leaving the $n+m$ qubits in the state

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \Sigma^n} |f(x)\rangle |x\rangle.$$

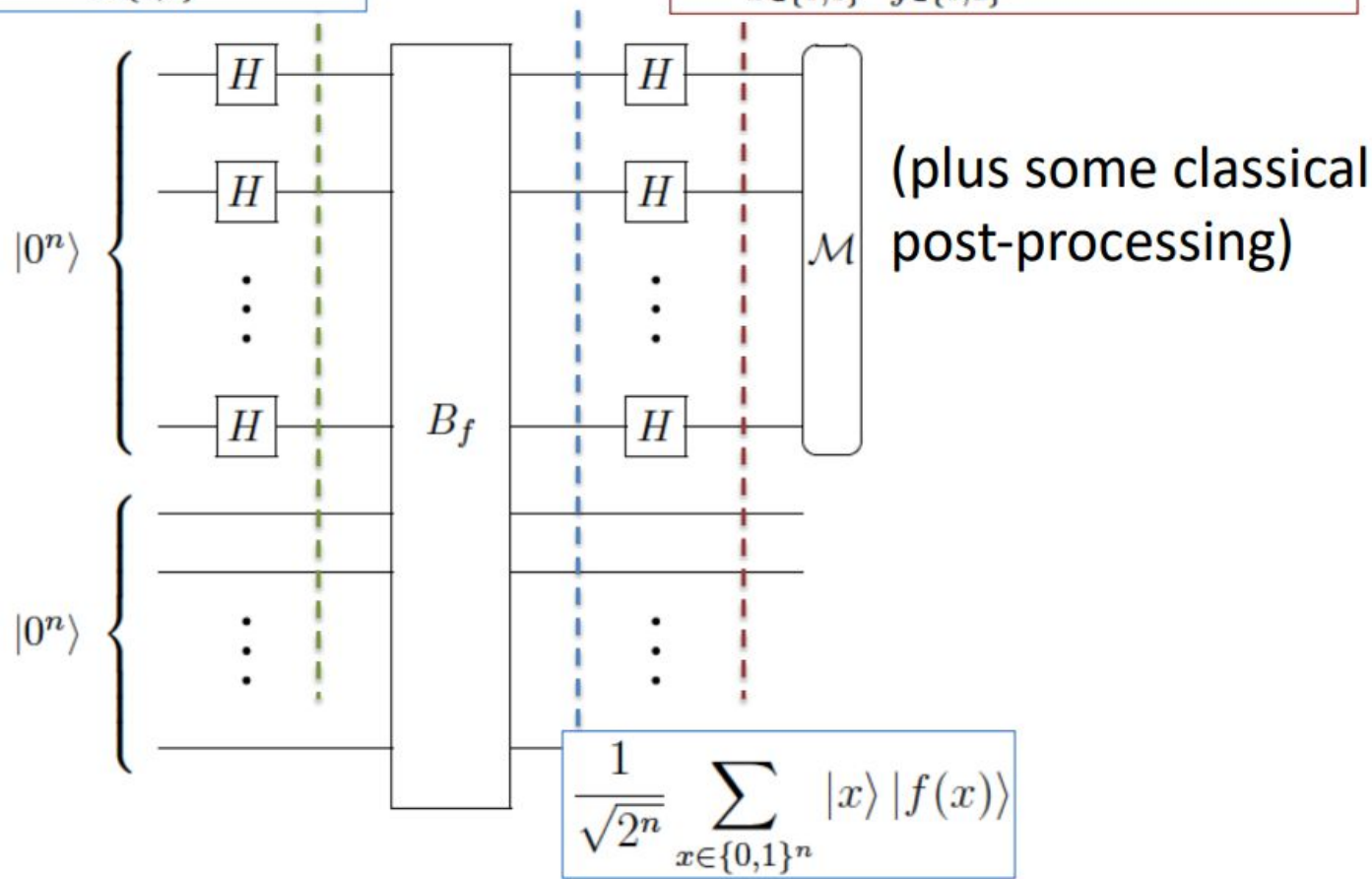
SIMON'S PROBLEM - ANALYSIS

When the second layer of Hadamard gates is performed, we obtain the following state by using the same formula for the action of a layer of Hadamard gates as before.

$$\frac{1}{2^n} \sum_{x \in \Sigma^n} \sum_{y \in \Sigma^n} (-1)^{y \cdot x} |f(x)\rangle |y\rangle$$

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle |0^n\rangle$$

$$\frac{1}{2^n} \sum_{x \in \{0,1\}^n} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle |f(x)\rangle$$



QUERY COMPLEXITY

Classical Query Complexity: $O(2^{(n/2)})$ queries in the best classical approach to solve this hidden subgroup problem.

Quantum Query Complexity: $O(n)$ queries. Simon's algorithm can find the hidden string s with only $O(n)$ quantum queries, demonstrating a substantial speedup.

REFERENCES - SIMON'S ALGORITHM

We have covered the examples from the below website in detail.

<https://jonathan-hui.medium.com/qc-simons-algorithm-be570a40f6de>

<https://www.youtube.com/watch?v=sxIdRKtdRRU>

SHOR'S ALGORITHM

Step: 1 Input N , the integer you want to factorize.

Step: 2 Randomly choose an integer k such that $1 < k < N$

Step: 3 Compute $\gcd(N, k)$

If $\gcd(a, N) \neq 1$, then N has a factor (namely, $\gcd(a, N)$), and you're done. Otherwise, proceed to the next step.

Step: 4 We need to find smallest positive integer r such that if $f(x) = k^x \bmod N$, then $f(a) = f(a+r)$

Step: 4.1 Define a new variable $q=1$

Step: 4.2 Find $(q \cdot k) \bmod N$

If remainder $\neq 1$, then set the value of q to the value of remainder we got, repeat this until remainder is 1.

Otherwise, proceed to the next step.

Step: 4.3 The number of transformations you did in Step 4.2 is your value of r .

SHOR'S ALGORITHM

Step: 5 If r is odd, go back to Step 2 and choose a different value of k .

Step: 6 Define $p = \text{remainder in } (r/2)\text{th transformation}$.

If $p + 1 = N$, then go back to Step 2 and choose a different value of k . Else, proceed to Step 7.

Step: 7 This is the final step. The factors of N are

$$f_1 = \text{GCD}(p+1, N)$$

$$f_2 = \text{GCD}(p-1, N)$$

We have covered numerical examples in class from the attached link in Reference.

SHOR'S ALGORITHM

<https://kaustubhrakhade.medium.com/shors-factoring-algorithm-94a0796a13b1>