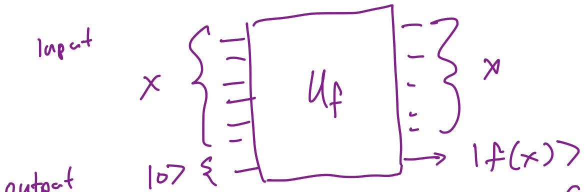
※ Computation 0: Quantum Parallelism

“The best lies are half-truths” - src

Let’s start this module by debunking a common misconception about quantum computing. There are many headlines claiming that quantum computation is able to achieve an exponential speed up over classical computing, because it is able to run an algorithm over an exponential number of inputs at once.

Question 99. Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function. Draw a circuit that applies a unitary U_f that takes as input an n bit string x, and outputs $f(x)$ on the $n+1$-th wire initialized to $|0\rangle$.

If we want to determine what each input string maps to, how many times do we have to run this circuit if we don’t use superposition?



x	$f(x)$
$00\cdots00$	1
$00\cdot\cdot01$	0
$\vdots$	
$11\cdots10$	1
$11\cdots11$	1

2^n queries,

$$U_f(|x\rangle|0\rangle) = |x\rangle|0\oplus f(x)\rangle$$

Question 100. Suppose that we prepared the state

$$|\psi\rangle = \frac{1}{\sqrt{2^n}}(|00..00\rangle|0\rangle + |00..01\rangle|0\rangle + |00..10\rangle|0\rangle + \cdots + |11..11\rangle|0\rangle) \qquad (80)$$

and applied U_f to it. What is the resulting state?

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

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

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

→ ~~Quantum Parallelism! Exponential Speed up!~~

No! Measuring collapses the state, so we only get 1 result!!

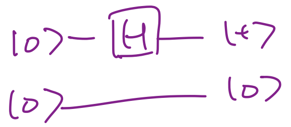
(80) $\underset{\uparrow}{\frac{1}{\sqrt{2^n}}}\left(|00\cdots00\rangle|0\rangle + |00\cdots01\rangle|0\rangle + |00\cdots10\rangle|0\rangle + \cdots + |11\cdots11\rangle|0\rangle\right)$

How do we prepare the state discussed above? Since it is a general state, the best way to approach this is from the small examples.

Question 101. What is state (80) when $n = 1$? What is the circuit to prepare this state?

$$\frac{1}{\sqrt{2}}\left(\overset{\downarrow}{|0\rangle}|0\rangle + \overset{\downarrow}{|1\rangle}|0\rangle\right) = \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right)|0\rangle = |+\rangle|0\rangle$$

$\Rightarrow$

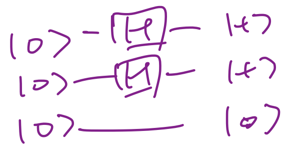


Question 102. What is state (80) when $n = 2$? What is the circuit to prepare this state?

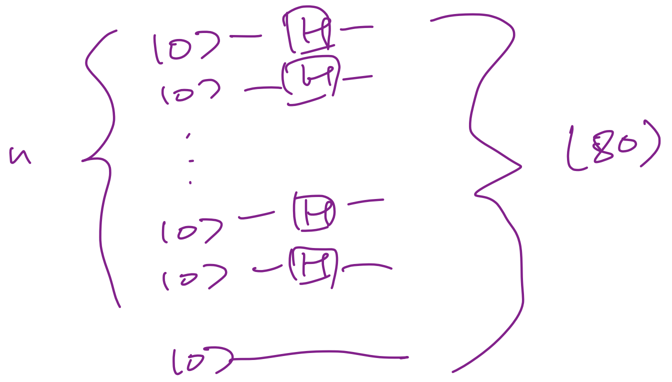
$$\frac{1}{2}\left(|00\rangle + |01\rangle + |10\rangle + |11\rangle\right)|0\rangle = \frac{1}{2}\left(|0\rangle\left(\underline{|0\rangle+|1\rangle}\right) + |1\rangle\left(\underline{|0\rangle+|1\rangle}\right)\right)|0\rangle$$

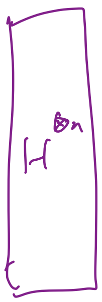
$$= \frac{1}{2}\left(|0\rangle+|1\rangle\right)\left(|0\rangle+|1\rangle\right)|0\rangle$$

$$= \underset{\uparrow}{|+\rangle|+\rangle}|0\rangle$$



Question 103. What is the circuit to prepare (80)?





Proposition 11.1 (*n*-qubit Hadamard)**.** Let $x = x_1x_2\cdots x_n$ be the binary expansion of x. In other words, x_i is the i-th bit of x when x is written in binary. Then, we have the following identity:

$$H^{\otimes n}|x\rangle = H|x_1\rangle \otimes H|x_2\rangle \otimes \cdots \otimes H|x_n\rangle \tag{81}$$

$$= \frac{(|0\rangle + (-1)^{x_1}|1\rangle)}{\sqrt{2}} \otimes \cdots \otimes \frac{(|0\rangle + (-1)^{x_n}|1\rangle)}{\sqrt{2}} \tag{82}$$

$$= \frac{1}{\sqrt{2^n}} \sum_{y\in\{0,1\}^n} (-1)^{x\cdot y}|y\rangle \tag{83}$$

$$H^{\otimes n}|y\rangle$$
$$\|$$
$$H^{\otimes n}(83)$$
$$\|$$
$$H^{\otimes n}(H^{\otimes n}|x\rangle)$$

where $x \cdot y$ is the bit wise dot product of x and y (i.e., $x \cdot y = x_1y_1 + \cdots + x_ny_n$).

Question 104. Verify that the above proposition holds for $x = 100$.

$$|100\rangle \xrightarrow{H^{\otimes 3}} H|1\rangle \otimes H|0\rangle \otimes H|0\rangle$$

$$= |-\rangle \quad |+\rangle \quad |+\rangle$$

$$= \frac{|0\rangle + (-1)^{x_1}|1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + (-1)^{x_2}|1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + (-1)^{x_3}|1\rangle}{\sqrt{2}}$$

$x = 100$

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

$$= \frac{1}{\sqrt{2}^3}\left(+|000\rangle + |001\rangle + |010\rangle + |011\rangle - |100\rangle - |101\rangle - |110\rangle - |111\rangle\right)$$

$$(-1)^{(100)\cdot\begin{pmatrix}0\\0\\0\end{pmatrix}}$$
$$\|$$
$$(-1)^0 = 1$$

$$(-1)^{(100)\begin{pmatrix}1\\1\\0\end{pmatrix}}$$
$$\|$$
$$(-1)^1 = -1$$

※ Computation 1: Query-based algorithms

12.1 Query Complexity

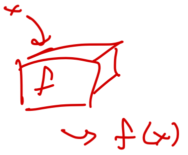
To mathematically prove the advantage that quantum computers have over classical computers, we would love to be able to answer a question like the following:

"Does there exist a problem that can be efficiently solved with a quantum computer that **cannot** be solved efficiently with a classical computer?" In complexity theoretic language, it is asking if there is a problem that is in BQP, but not in P.

We don't really know how to prove this, because we don't know how to show that some problems **cannot** be solved efficiently. To get some handle on this issue, we study a more limited model, and analyze what is called the **query complexity** of a problem.

In query complexity, we assume that we have black box access to a Boolean function $f : \{0,1\}^n \to \{0,1\}$, and we want to know how many times we have to call this function to determine some property of the function. As you will see, some of these settings are quite artificial, but they provide good insight into the techniques that we know about quantum algorithm design, and are a proof of concept that there are settings where quantum computers perform better than classical. They are also one of our best tools for proving lower bounds for problems.

Question 105. What is the query complexity of a classical algorithm to call a function to determine the following properties?



- Is there any input x such that $f(x) = 1$?

$\sim 2^n$

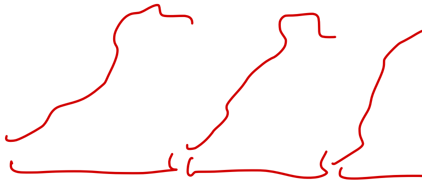
- Does $f(x) = 1$ for a majority of the inputs?

2^{n-1} $\sim o(2^n)$ ○ ● ○ ● ○ ● ○ ●

- Is f periodic?

$f(x) = f(x+c)$

2^n



To analyze the query complexity in a quantum setting, we need to embed this black box access to f into a quantum circuit. At the end of the previous module, we showed that if f can be computed by a classical circuit, then there exists a reversible circuit that computes f. Mathematically, we will express the general action of the reversible circuit as

$$(x, y, 0^k) \to (x, y \oplus f(x), 0^k). \tag{84}$$

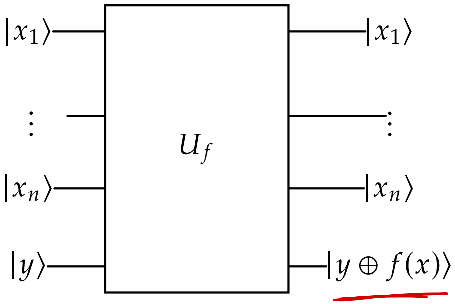
Since the last register starts and ends with 0s for all inputs, we can just ignore it. Now we can embed our query to f as the reversible circuit with the following action:

input output

$U_f|x\rangle|0\rangle = |x\rangle|f(x)\rangle$

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

$U_f|x\rangle|-\rangle = (-1)^{f(x)}|x\rangle|-\rangle$



$|0\rangle$

$|y\rangle=|0\rangle \rightarrow |x\rangle|f(x)\rangle$

$|y\rangle=|-\rangle \rightarrow (-1)^{f(x)}|x\rangle|-\rangle$

Often when implementing quantum algorithms, we want the output to be stored in the phase instead of in an extra qubit:

$$|x\rangle \longrightarrow (-1)^{f(x)}|x\rangle \tag{85}$$

This can be very useful for orchestrating interference patterns as we will see.

Question 106. Show that if we set $|y\rangle = |-\rangle$, we can use the above circuit to implement equation (85).

$U_f|x\rangle|-\rangle = U_f|x\rangle\left(\frac{1}{\sqrt{2}}(|0\rangle-|1\rangle)\right)$

$= \frac{1}{\sqrt{2}}\left(U_f|x\rangle|0\rangle - |x\rangle|1\rangle\right)$

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

$f(x)=0$

$\frac{1}{\sqrt{2}}\left(|x\rangle|0\rangle - |x\rangle|1\rangle\right)$

$= |x\rangle|-\rangle$

$f(x)=1$

$\frac{1}{\sqrt{2}}\left(|x\rangle|1\rangle - |x\rangle|0\rangle\right)$

$= -|x\rangle|-\rangle$

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

$f(x) = x_1 \oplus \bar{x}_2$

$\uparrow$

$0 \oplus \bar{1} = 0 \oplus 0 = 0.$

$1 \oplus \bar{1} = 1 \oplus 0 = 1$

$$\Rightarrow \quad U_f |01\rangle \underline{|0\rangle} \overset{?}{=} |01\rangle |f(01)\rangle = +|01\rangle |0\rangle$$

$$U_f |01\rangle \underline{|-\rangle} \overset{?}{=} (-1)^{\overset{f(01)}{\uparrow}} |01\rangle \underline{|-\rangle} = +|01\rangle |-\rangle$$

$$\rightarrow \quad U_f |11\rangle |0\rangle \overset{?}{=} |11\rangle |f(11)\rangle = |11\rangle |1\rangle$$

$$U_f |11\rangle \underline{|-\rangle} \overset{?}{=} (-1)^{\overset{f(11)}{\uparrow}} |11\rangle \underline{|-\rangle} = -|11\rangle \underline{|-\rangle}$$

12.2 Deutsch's Algorithm

Deutsch's algorithm is the smallest quantum algorithm that one would come up with to experiment with quantum speedup in our circuit model. It is a toy example, but the ideas used will give us the groundwork for thinking about more complex quantum algorithms.

Consider a single bit Boolean function $f : \{0,1\} \to \{0,1\}$. We will denote its output bit for each input as

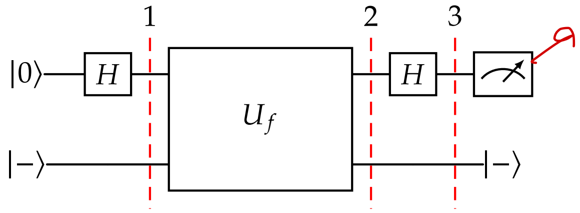
- $f(0) = b_0$
- $f(1) = b_1$

Given this function, we would like to determine the **parity** of $b_0 + b_1$, or more succinctly, we want to compute $b_0 \oplus b_1$.

Question 107. How many queries to f do we need classically to determine the parity of f?

2

I now claim that using a quantum computer, we can determine the parity using just one call to f. Here is the circuit for Deutsch's algorithm.



Question 108. Analyze the state of the circuit at each timestep.

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

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

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

③ $\frac{1}{2}((-1)^{f(0)}(\underline{|0\rangle}+\underline{|1\rangle}) + (-1)^{f(1)}(\underline{|0\rangle}\underset{\uparrow}{-}\underline{|1\rangle}))|-\rangle$

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

→ **Question 109.** What is the state of the system at ~~2~~ 3 if $\underline{f(0) = \overset{=a}{f(1)}}$? What are the possible measurement outcomes for Deutsch's algorithm in this case? ↳ parity is 0.

0

$\frac{1}{2}(((-1)^{a}+(-1)^{a})|0\rangle + (\cancel{(-1)^{a}-(-1)^{a}})|1\rangle)|-\rangle$

$= |0\rangle|-\rangle$

Measurement: $|0\rangle$ w/ 100%.

Question 110. What is the state of the system at 3 if $f(0) \neq f(1)$? What are the possible measurement outcomes for Deutsch's algorithm in this case?

$\frac{1}{2}((\cancel{(-1)^{a}+(-1)^{\bar{a}}})|0\rangle + ((-1)^{a}-(-1)^{\bar{a}})|1\rangle)|-\rangle$

$= |1\rangle|-\rangle$

Measurement: $|1\rangle$ w/ 100%.

12.3 Deutsch-Josza Algorithm

The Deutsch-Josza algorithm is a generalization of what we saw in the previous section. This time, we have access to a Boolean function with n-bit inputs:

$$f : \{0,1\}^n \to \{0,1\} \tag{86}$$

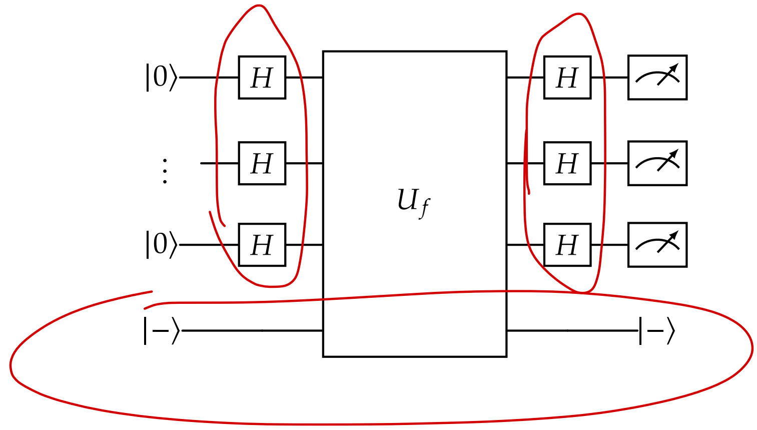
and are **promised** that f satisfies one of the two following properties:

- f is a **constant function**, meaning that $f(x) = c$ for all inputs x
- f is a **balanced function**, meaning that $f(x) = 0$ for half of the inputs, and $f(x) = 1$ for the remaining half.

Question 111. How many queries do we need to make to this function to decide with 100% certainty which property is satisfied using a classical computer?

$$2^{n-1} + 1 = \frac{2^n}{2} + 1 \sim O(2^n)$$

A quantum circuit can answer this question using just one query, with 0 probability of error. Here's the circuit:



For convenience, here is the main equation of Proposition 11.1 repeated:

$$H^{\otimes n}|x\rangle = H|x_1\rangle \otimes H|x_2\rangle \otimes \cdots \otimes H|x_n\rangle \tag{87}$$
$$= \frac{(|0\rangle + (-1)^{x_1}|1\rangle)}{\sqrt{2}} \otimes \cdots \otimes \frac{(|0\rangle + (-1)^{x_n}|1\rangle)}{\sqrt{2}} \tag{88}$$
$$\boxed{= \frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle} \tag{89}$$

Question 112. Using Proposition 11.1, what is the state of the Deutsch-Jozsa algorithm before the call to U_f?

$$\underbrace{\left(H^{\otimes n}|\underbrace{00\cdots00}_{n}\rangle\right)}|-\rangle = \frac{1}{\sqrt{2^n}}\sum_{y=\{0,1\}^n}(-1)^{\overset{0}{\cancel{x}}\cdot y}|y\rangle$$

Applying n H gates to n |0⟩'s, prepares an equal superposition of n bit strings.

$$= \frac{1}{\sqrt{2^n}}\sum_{y=\{0,1\}^n}|y\rangle = \frac{1}{\sqrt{2^n}}\sum_{y=0}^{2^n-1}|y\rangle$$

$$= \frac{1}{\sqrt{2^n}}\left(|00\cdots00\rangle + |00\cdots01\rangle + |0\cdots10\rangle + \cdots + |11\cdots11\rangle\right)$$

Question 113. What is the state of the Deutsch-Josza algorithm after the call to U_f?

$$U_f\left(\underbrace{\frac{1}{\sqrt{2^n}}\sum_{y=0}^{2^n-1}}|y\rangle|-\rangle\right) = \frac{1}{\sqrt{2^n}}\sum_{y=0}^{2^n-1}\left(U_f|y\rangle|-\rangle\right)$$

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

Question 114. Using Proposition 11.1 again, what is the state of the Deutsch-Josza algorithm after the second layer of H gates?

$$H^{\otimes n}\left[\frac{1}{\sqrt{2^n}}\sum_{y}(-1)^{f(y)}|y\rangle|-\rangle\right] = \frac{1}{\sqrt{2^n}}\sum_{y}(-1)^{f(y)}\underbrace{H^{\otimes n}|y\rangle}|-\rangle$$

by prop 11.1

$$= \frac{1}{\sqrt{2^n}}\sum_{y}(-1)^{f(y)}\left[\frac{1}{\sqrt{2^n}}\sum_{z\in\{0,1\}^n}(-1)^{y\cdot z}|z\rangle\right]|-\rangle$$

$$= \frac{1}{2^n}\sum_{y}\Big[\underbrace{(-1)^{f(y)}(-1)^{y\cdot\vec{0}}|0\cdots0\rangle + (-1)^{f(y)}(-1)^{y\cdot\vec{1}}|0\cdots1\rangle + \cdots}_{2^n}\Big]|-\rangle$$

2^n

$$\begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Question 115. What is the amplitude of $|0 \cdots 0\rangle$ if f is constant?

$$\frac{1}{2^n}\sum_y (-1)^{f(y)}\cancel{(-1)^{y\cdot 0}} = \frac{1}{2^n}\sum_y (-1)^{f(y)}$$

$f(y)=0$ for all y: $\Rightarrow \frac{1}{2^n}\sum_y (-1)^0 = \frac{1}{2^n}\overbrace{\sum_{y=\{0,1\}^n} 1}^{1\cdot 2^n} = \underline{1}$

$f(y)=1$ for all y: $\Rightarrow \frac{1}{2^n}\sum_y -1 = \frac{-2^n}{2^n} = \underline{\underline{-1}}.$

$$\begin{pmatrix} 0 \\ \vdots \\ 1 \end{pmatrix}$$

Question 116. What is the amplitude of $|0 \cdots 0\rangle$ if f is balanced?

$$\frac{1}{2^n}\sum_y (-1)^{f(y)}\cancel{(-1)^{y\cdot 0}} = \frac{1}{2^n}\sum_y \underbrace{(-1)^{f(y)}}$$

In the sum, half the terms are -1, half the terms are $+1$

$$\Rightarrow \frac{1}{2^n}\sum_y (-1)^{f(y)} = \underline{\underline{0}}$$

Question 117. How can we use the measurement results to decide which property is held for the function f?

Prob. of seeing $|0\rangle$ is 1 if f is constant

" 0 if f is balanced.

$O(2^n)$ queries $\Rightarrow$ 1 query!

It turns out that if we allow for randomized classical algorithms where we can make errors, a simple sampling algorithm will very quickly be able to decide which property is held with high confidence. Because of this, the quantum speedup is not as glamorous as it seems.

umesh.

12.4 Bernstein-Vazirani

The Bernstein-Vazirani algorithm is given black box access to a function $f : \{0,1\}^n \to \{0,1\}$ that we know is in the form "promise"

$$f_s(x) = x \cdot s \pmod 2 \tag{90}$$

for some mystery string $s \in \{0,1\}^n$. The goal of this algorithm is to figure out what s is.

Question 118. Let's consider an example where $n = 5$ and the secret string is $s = 10110$. What is $f(11101)$?

x

$$f_{10110}(11101) = x \cdot s \pmod 2 = 1 + 0 + 1 + 0 + 0 = \underline{\underline{0}}.$$

s

Question 119. What is a strategy we can use using a classical computer to decide what s is? What is the optimal query complexity classically?

$$f(10000) \to (1\ 0\ 0\ 0\ 0)\begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \end{pmatrix} = s_0$$

$$f(01000) \to (0\ 1\ 0\ 0\ 0)\begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ ? \end{pmatrix} = s_1$$

$$f(00100) \to$$

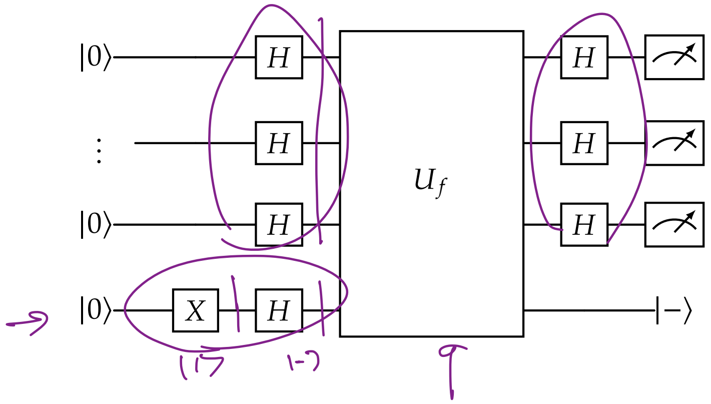
$$f(00010)$$

$$f(00001)$$

$$\Rightarrow \Theta(n)$$

queries.

Here is the circuit for the Bernstein-Vazirani algorithm:



Question 120. What is the state of the algorithm before the query to U_f?

$$H^{\otimes n}|0\rangle|-\rangle = \frac{1}{\sqrt{2^n}} \sum_{y\in\{0,1\}^n} |y\rangle\,|-\rangle$$

$f(y) = s\cdot y$

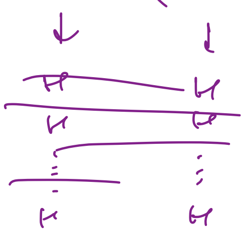
Question 121. What is the state of the algorithm after the query to U_f?

$$U_f \frac{1}{\sqrt{2^n}} \sum_y |y\rangle|-\rangle = \frac{1}{\sqrt{2^n}} \sum_y \left(U_f |y\rangle|-\rangle\right)$$

$$= \frac{1}{\sqrt{2^n}} \sum_y \left((-1)^{f(y)} |y\rangle\,|-\rangle\right) = \underbrace{\frac{1}{\sqrt{2^n}} \sum_x (-1)^{s\cdot x} |x\rangle|-\rangle}_{H^{\otimes n}|s\rangle\,|-\rangle}$$

Question 122. What is the state of the algorithm after the second layer of H gates?

$$H^{\otimes n}\left(H^{\otimes n}|s\rangle\,|-\rangle\right) = |s\rangle|-\rangle$$



$O(n)$ queries → 1 query.

Bernstein and Vazirani chose this problem since there is a way to have all the amplitudes for $y \neq s$ interfere destructively to become 0, while the amplitudes for s all "point in the same direction" and interfere constructively to become 1. We have found a way to achieve a linear query complexity speed up using a quantum algorithm, but can we do even better? Are there setting where we can achieve exponential speed up?

12.5 Simon's Algorithm

In this problem, we will consider a function with an n bit input and an n bit output. The function $f : \{0,1\}^n \to \underline{\{0,1\}^n}$ will encode a secret string s in the following way.

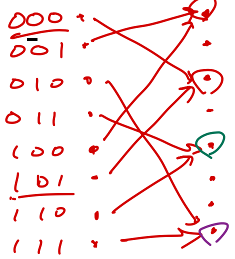
promise. →

$$\underbrace{f(x) = f(y) \iff \overset{\text{bit wise sum mod 2.}}{x \oplus y} = s} \iff x \oplus s = y \tag{91}$$

As we have been doing, given blackbox access to this function, the goal is to find s.

Question 123. Let f be a function that takes an n bit input and satisfies the property above. What is the size of the domain of this function? What is the size of the range of this function?

3 bit example:



$$\begin{array}{r} 000 \\ \oplus\ 101 \\ \hline s \to 101 \end{array} \qquad \begin{array}{r} 001 \\ \oplus\ 100 \\ \hline s \to 101 \end{array}$$

$$\begin{array}{r} 011 \\ \oplus\ 110 \\ \hline s \to 101 \end{array} \qquad \begin{array}{r} 010 \\ \oplus\ 111 \\ \hline s \to 101 \end{array}$$

To determine what s is, we need to find a pair x and y such that $f(x) = f(y)$, and then take the sum module 2 of these strings to recover s.

→ **Question 124.** How many queries will we need classically in the worst case to determine s?

$$2^n/2 + 1$$

What if we use a randomized classical algorithm? In this case, we can show that we will require approximately $\sqrt{2^n} = 2^{n/2}$ queries. Let's try to prove this together. To prove this, we will use a general version of the Birthday Paradox.

Suppose we have a set of items, each with a uniformly random tag from $\{1, 2, \ldots, T\}$. How many samples do we need to collect before we have at least two items with the same tag with probability greater than 1/2?

Question 125. What is the probability that a random pair of items have matching tags?

$$1 \cdot 1/T = 1/T$$

pick item #1

pick item w/ the same tag as #1.

Question 126. Suppose we have chosen m items so far. How many different ways can we pair two items from this set (the tags do not have to match)?

$$\binom{m}{2} = \frac{m\cdot(m-1)}{2} \sim \frac{m^2}{2}$$

Question 127. Determine how many items m we have to choose until the probability that there is a collision is over 1/2.

$$\binom{m}{2} \cdot \frac{1}{T} \geq \frac{1}{2} \Rightarrow m^2 \geq T$$

$$m \geq \Omega(\sqrt{T})$$

# of pairs in current selection

prob. a pair collides.

Now suppose that this randomized algorithm queries the function using t bit strings, x_1, x_2, ..., x_t.

2^n

- If we find a pair such that $f(x_i) = f(x_j)$, then we are done.
- If none of these $x_i's$ are matches, then we know that $s \neq x_i \oplus x_j$ for all i, j pairs. In other words, we have ruled out $\binom{t}{2} \sim \frac{t^2}{2}$ possibilities, and all other choices are equally likely. In the worst case, we need to find t such that the number of items we rule out equals all possible inputs.

We can conclude then, that classically we need at least $\Omega(2^{n/2})$ queries.

Need $\sqrt{2^n}$ samples, even for randomized alg.

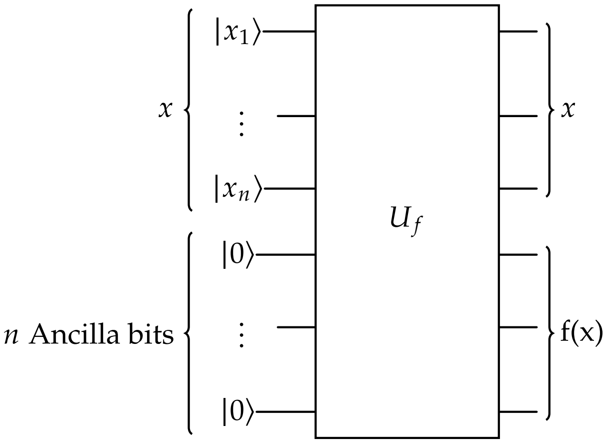
How can we solve this problem using a quantum computer? The unitary encoding the function would act as follows:

$$U_f(\vec{x},\vec{0}) = (\vec{x},\vec{0} \oplus f(x)) \tag{92}$$

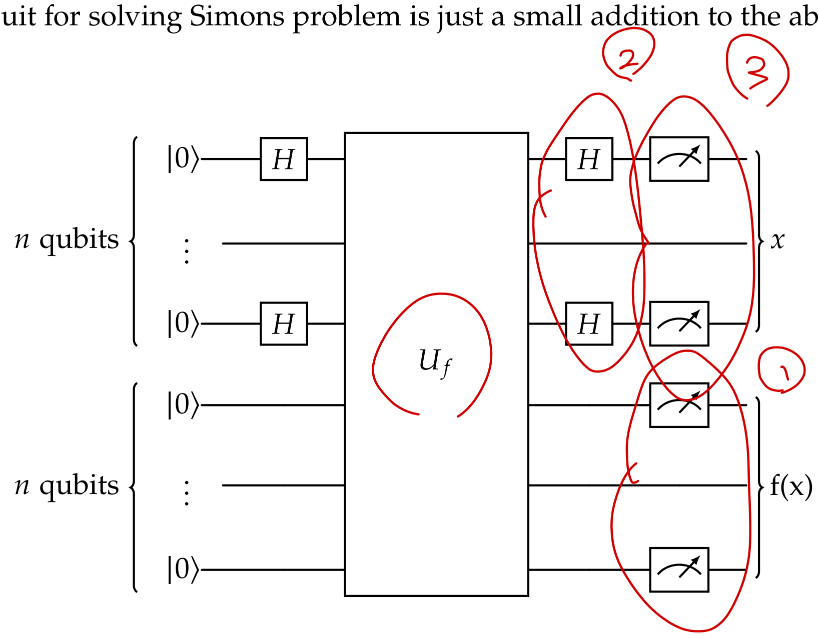
or in ket notation

$$U_f \left|x\right\rangle \left|0\right\rangle = \left|x\right\rangle \left|f(x)\right\rangle . \tag{93}$$

As a circuit, they would look like the following.



Now the circuit for solving Simons problem is just a small addition to the above circuit and is drawn below.



The neat thing about this algorithm is that we don't actually care about what our measurement result is, but just the interference pattern that is created. Let's go through the circuit to see what we mean by this.

Question 128. What is the state of the algorithm before the U_f gate?

$$H^{\otimes n}\left(|\vec{0}\rangle|\vec{0}\rangle\right) = \frac{1}{\sqrt{2^n}} \sum_{y\in\{0,1\}^n} |y\rangle|\vec{0}\rangle$$

$$\Rightarrow H^{\otimes 3}|000\rangle|000\rangle = \frac{1}{\sqrt{2^3}}\Big(|000\rangle+|001\rangle+|010\rangle+|011\rangle + |100\rangle+|101\rangle+|110\rangle+|111\rangle\Big)\otimes|000\rangle$$

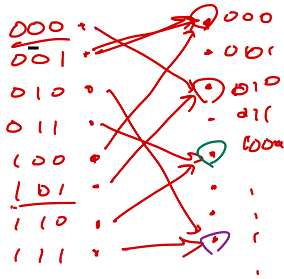
Question 129. What is the state of the algorithm after the U_f gate?

$$U_f\left(\frac{1}{\sqrt{2^n}}\sum_y |y\rangle|\vec{0}\rangle\right) = \frac{1}{\sqrt{2^n}}\sum_y \Big(U_f\, \underset{\overset{\uparrow}{\text{input}}}{\underline{|y\rangle}}|\vec{0}\rangle\Big) = \frac{1}{\sqrt{2^n}}\sum_y |y\rangle|f(y)\rangle$$

$$\Rightarrow \frac{1}{\sqrt{2^3}}\Big(|000\rangle|010\rangle + |001\rangle\underline{|000\rangle} + |010\rangle|111\rangle + \underbrace{|011\rangle|100\rangle}_{} + |100\rangle\underline{|000\rangle} + |101\rangle|010\rangle + \underbrace{|110\rangle|100\rangle}_{} + |111\rangle|111\rangle\Big)$$

Possible outcomes for second reg. measurement:

$|000\rangle, |010\rangle, \underline{|100\rangle}, |111\rangle$



It turns out that we can measure the last n qubits before applying the H gates on the first n qubits. The output distribution will be the same in either case!

Question 130. Suppose that upon measuring the second register at this stage, we get the result $|w\rangle$. What is the state of the first register?

$$\frac{1}{\sqrt{2}}\Big(|011\rangle + |110\rangle\Big)|100\rangle$$

$$\frac{1}{\sqrt{2}}\Big[|x\rangle + |y\rangle\Big]|w\rangle \quad \text{s.t. } f(x)=f(y)=w$$

It would be great if we could have multiple copies of the above state, because then we can directly measure the first register to recover all the relevant states. The problem is that if we rerun this experiment, it is extremely unlikely (how unlikely?) that we measure $|w\rangle$ again!

Instead, what this circuit is doing is measuring in the H basis by applying the H gates.

Question 131. What is the state of the superposition after the final Hadamard gates?

$$H^{\otimes n}\left[\frac{|x\rangle+|y\rangle}{\sqrt{2}}\right] = \frac{1}{\sqrt{2}}\cdot\frac{1}{\sqrt{2^n}}\sum_{z\in\{0,1\}^n}\left[(-1)^{x\cdot z}+(-1)^{y\cdot z}\right]|z\rangle$$

Prop 11.1

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

Prop 11.1

$$H^{\otimes n}|y\rangle = \frac{1}{\sqrt{2^n}}\sum_{z\in\{0,1\}^n}(-1)^{y\cdot z}|z\rangle$$

Question 132. Suppose we measured the first register and observe some random $|z\rangle$. What can we say about the coefficient of such a $|z\rangle$?

For $|z\rangle$ to have nonzero coefficient, we need $(-1)^{x\cdot z}+(-1)^{y\cdot z}\neq 0$

$$\Rightarrow (-1)^{x\cdot z} = (-1)^{y\cdot z}$$
$$\Rightarrow x\cdot z \equiv y\cdot z \pmod{2}$$
$$\Rightarrow (x-y)\cdot z \equiv 0 \pmod{2}$$
$$\Rightarrow \underline{(x\oplus y)}\cdot z \equiv 0 \pmod{2}$$
$$s\cdot z \equiv 0 \pmod{2}$$

This can be analyzed using modular arithmetic:

$$x\cdot z \mod 2 = y\cdot z \mod 2 \tag{94}$$
$$(x-y)\cdot z \mod 2 = 0. \tag{95}$$

When working in binary, $x-y$ is equivalent to $x\oplus y$. Therefore what we get is that for the string z we recovered,

$$(x\oplus y)\cdot z = s\cdot z = 0. \tag{96}$$

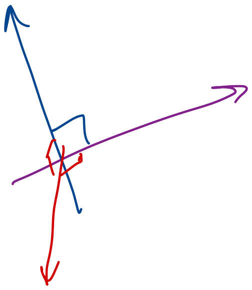
So in 1 run of Simon's algorithm we found a random z that is orthogonal to s!

The measurement yields a random z such that $s \cdot z \equiv 0(\mod 2)$. We can repeat this $O(n)$ times to get a set of linearly independent strings who are all orthogonal to s. Once we have this, we can use Gaussian elimination (mod 2) to find s in $O(n^3)$ time.

$$\begin{bmatrix} --- & z_1 & --- \\ --- & z_2 & --- \\ & \vdots & \\ --- & z_m & --- \end{bmatrix} \cdot \begin{bmatrix} | \\ s \\ | \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \qquad \frac{1}{2^{n-1}} \tag{97}$$

This gives us a polynomial time quantum algorithm to find s, whereas classically the best we could do was still exponential.

Here is another one



this is orthogonal to s!

3 Dimensions

Needed 2 orthogonal vectors to identify the secret vector.

Length of strings are $\underline{n}$.

Need $\Omega(n-1)$ samples from Simon's alg.

Classically $\Omega(2^n)$ → Quantum $\Omega(n)$

12.6 Query Based Algorithm Wrap Up

We've looked at several algorithms in this strange query model, which achieves speedups in a non-standard way. You may be suspicious that we are sweeping too many details under the rug, and for that you would be correct. To actually *implement* Simon's algorithm, you need an actual circuit to compute f, and when given to the actual circuit (as opposed to a black-box oracle), classical algorithms can exploit the details of the circuit to significantly reduce the number of queries.

Unfortunately, because of this reason these algorithms we have seen so far are not actually very practical for finding ways to speed up our computations. However, they provided valuable practice using some tools that will be useful for analyzing other quantum algorithms. Furthermore, I hope it gave you a peek into the workflow of a computer science researcher, and some ways that we try to separate the power of classical and quantum computing. It is not perfect, but it provides some concrete examples and intuition behind why quantum computers may excel at certain tasks over classical computers.