❈ Computation 1: Query-based algorithms

1.1 Query Complexity

> **Learning Outcomes**
> Upon following these notes and the corresponding lecture, students will be able to
>
> - define what query complexity is and how we calculate it in the classical and quantum setting.
>
> - describe the two ways to reversibly access a black box function.

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 work around this issue, we study a more limited model, and analyze what is called **query complexity**.

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 often used to prove lower bounds on algorithms.

Question 1. 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 $f(x) = 1$?

  Worst case, 2^n queries.
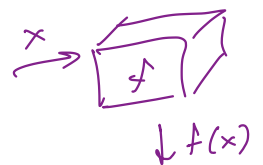
- Does $f(x) = 1$ for *most* of the inputs?

  $$\frac{2^{n-1}}{2} + 1 = \frac{2^n}{2} + 1 \qquad 2^n \text{ queries.} \qquad 1, 0, 1, 0, \ldots$$

- Is f periodic?

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

  $$2^n$$

2

XOR, additions mod 2

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

$$(x, y, \cancel{0}) \rightarrow (x, y \oplus f(x), \cancel{0}). \tag{1}$$

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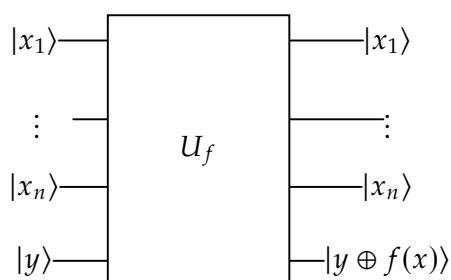
$$U_f |x\rangle |-\rangle$$
$$= (-1)^{f(x)} |x\rangle |-\rangle.$$

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

$$|x\rangle \rightarrow (-1)^{f(x)} |x\rangle \tag{2}$$

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

Question 2. Show that for a particular value of $|y\rangle$, we can use the above circuit to implement equation (2).

$$|y\rangle = |-\rangle$$

$$|x\rangle |-\rangle = |x\rangle \tfrac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

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

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

$f(x) = 0.$

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

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

$f(x) = 1.$

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

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

$$= |x\rangle \tfrac{1}{\sqrt{2}}\underbrace{(|1\rangle - |0\rangle)}_{-|-\rangle} = -|x\rangle |-\rangle.$$
$$= (-1)^{f(x)} |x\rangle |-\rangle.$$

1.2 Deutsch's Algorithm
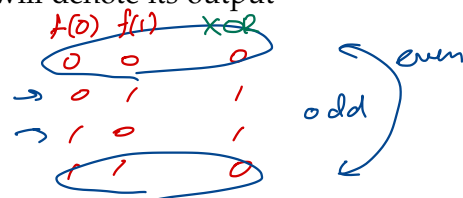
> **Learning Outcomes**
>
> Upon following these notes and the corresponding lecture, students will be able to
>
> - describe the property of the function Deutsch's Algorithm is trying to determine.
>
> - analyze the circuit of Deutsch's Algorithm.

Deutsch's algorithm was the first quantum algorithm proposed that demonstrated a speed up in query complexity over classical, though the speed up isn't too exciting. Nevertheless, 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

- $f(0) = b_0$

- $f(1) = b_1$

(handwritten annotation: table with columns f(0), f(1), XOR)

| $f(0)$	$f(1)$	XOR	
0	0	0	even
0	1	1	odd
1	0	1	odd
1	1	0	even

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

(handwritten answer:) 2.

Query $f(0), f(1).$ Compute $f(0) \oplus f(1)$

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.
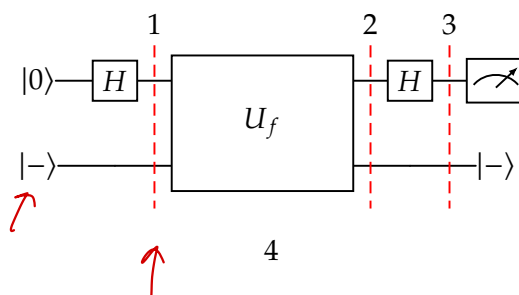
Question 4. What is the state of the system at 1?

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

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

Question 5. What is the state of the system at 2?

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

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

Question 6. What is the state of the system at 2 if $f(0) = f(1)$? What are the possible measurement outcomes for Deutsch's algorithm in this case?

$f(0) = f(1) = 0.$

$$\frac{1}{\sqrt{2}}\left[(-1)^0|0\rangle|-\rangle + (-1)^0|1\rangle|-\rangle\right] = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|-\rangle = |+\rangle|-\rangle.$$

$$\xrightarrow{\;\;H\;\;} |0\rangle|-\rangle$$

$f(0) = f(1) = 1.$

$$\frac{1}{\sqrt{2}}\left[(-1)^1|0\rangle|-\rangle + (-1)^1|1\rangle|-\rangle\right] = -\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|-\rangle = -|+\rangle|-\rangle$$

$$\xrightarrow{\;\;H\;\;} -|0\rangle|-\rangle$$

Question 7. 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?

$f(0) = 0,\ f(1) = 1;$

$$\frac{1}{\sqrt{2}}\left[(-1)^0|0\rangle|-\rangle + (-1)^1|1\rangle|-\rangle\right] = \frac{1}{\sqrt{2}}\left[|0\rangle|-\rangle - |1\rangle|-\rangle\right]$$

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

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

$$\xrightarrow{\;\;H\;\;} |1\rangle|-\rangle.$$

1.3 Deutsch-Josza Algorithm

> **Learning Outcomes**
> Upon following these notes and the corresponding lecture, students will be able to
>
> - describe the property of the function the Deutsch-Josza Algorithm is trying to determine.
>
> - apply the n-qubit Hadamard identity.
>
> - analyze the circuit of Deutsch's 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:

$$f : \{0,1\}^n \to \{0,1\} \tag{3}$$

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

$$2^{n-1} + 1 \quad = \frac{2^n}{2} + 1$$

A quantum circuit can answer this question using just one query, with 0 probability of error. Here's the circuit:
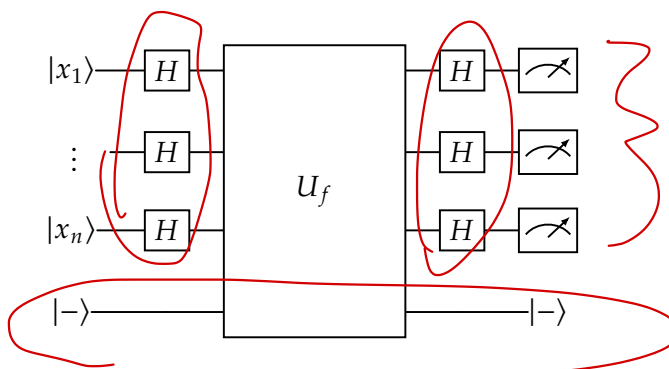
To analyze this algorithm, we make use of an identity which will appear a lot through-out the rest of this class.

Proposition 1.1 (*n*-qubit Hadamard). Let $x = x_1 x_2 \cdots x_n$ 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:

$$H^{\otimes n} |x\rangle = H |x_1\rangle \otimes H |x_2\rangle \otimes \cdots \otimes H |x_n\rangle \tag{4}$$

$$= \frac{(|0\rangle + (-1)^{x_1} |1\rangle)}{\sqrt{2}} \otimes \cdots \otimes \frac{(|0\rangle + (-1)^{x_n} |1\rangle)}{\sqrt{2}} \tag{5}$$

$$= \frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle \tag{6}$$

where $x \cdot y$ is the bit wise dot product of x and y (i.e., $x \cdot y = x_1 y_1 + \cdots + x_n y_n$).

Question 9. Using the above identity, what is the state of the Deutsch-Jozsa algorithm before the call to U_f?

Question 10. What is the state of the Deutsch-Josza algorithm after the call to U_f?

Question 11. Using the above identity again, what is the state of the Deutsch-Josza algorithm after the second layer of H gates?

Question 12. What is the amplitude of $|0 \cdots 0\rangle$ if f is constant?

Question 13. What is the amplitude of $|0 \cdots 0\rangle$ if f is balanced?

Question 14. How can we use the measurement results to decide which property is held for the function f?

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.

1.4 Bernstein-Vazirani

> **Learning Outcomes**
>
> Upon following these notes and the corresponding lecture, students will be able to
>
> • describe the property of the function the Bernstein-Vazirani Algorithm is trying to determine.
>
> • analyze the circuit of the Bernstein-Vazirani's Algorithm.

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

$$f_s(x) = x \cdot s \ (\text{mod } 2) \tag{7}$$

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

Question 15. Let's consider an example where $n = 5$ and the secret string is $s = 10110$. What is $f(11101)$?

Question 16. What is a strategy we can use using a classical computer to decide what s is? What is the optimal query complexity classically?

Here is the circuit for the Bernstein-Vazirani algorithm:
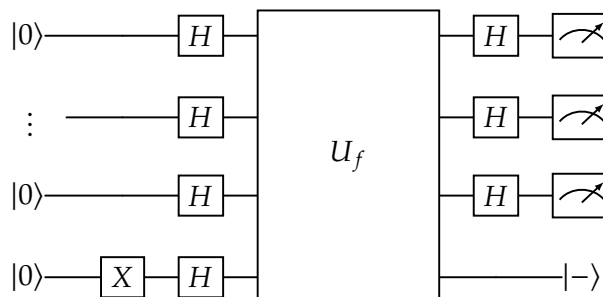
Question 17. What is the state of the algorithm before the query to U_f?

Question 18. What is the state of the algorithm after the query to U_f?

Question 19. What is the state of the algorithm after the second layer of H gates?

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?

1.5 Simon's Algorithm

> **Learning Outcomes**
>
> Upon following these notes and the corresponding lecture, students will be able to
>
> • describe the property of the function the Simon's Algorithm is trying to determine.
>
> • prove the generalized Birthday paradox.
>
> • analyze how hard it is to decide the property classically.
>
> • analyze the circuit of the Bernstein-Vazirani's Algorithm.

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

$$f(x) = f(y) \Longleftrightarrow x \oplus y = s \Longleftrightarrow x \oplus s = y \tag{8}$$

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

Question 20. 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?

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 21. How many queries will we need classically in the worst case to determine s?

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

Question 23. 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)?

Question 24. Determine how many items m we have to choose until the probability that there is a collision is over $1/2$.

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

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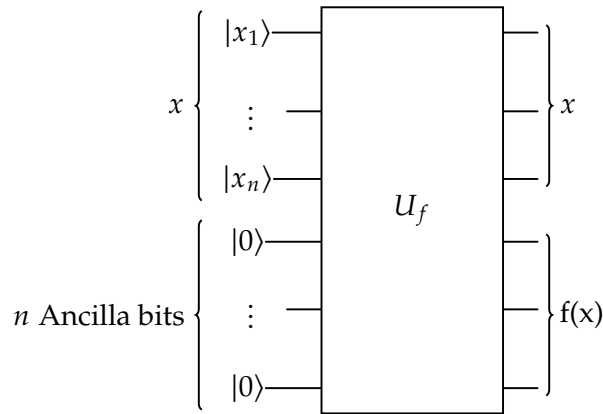
How can we solve this problem using a quantum computer? The unitary encoding the function would act as follows:

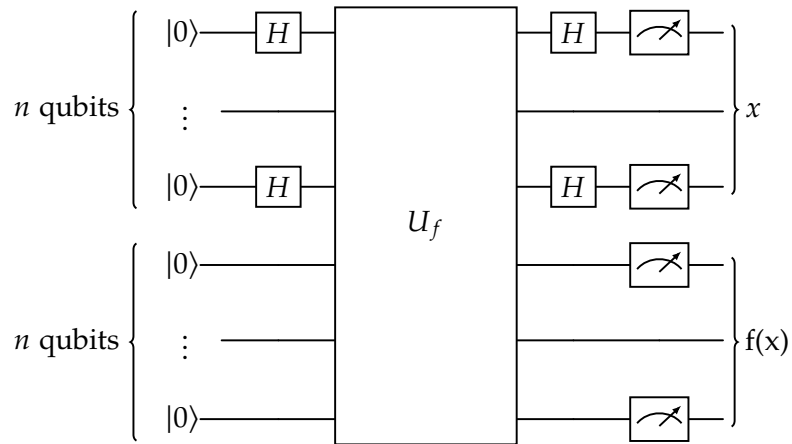$$U_f(\vec{x}, \vec{0}) = (\vec{x}, \vec{0} \oplus f(x)) \tag{9}$$

or in ket notation

$$U_f \ket{x} \ket{0} = \ket{x} \ket{f(x)}. \tag{10}$$

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 25. What is the state of the algorithm before the U_f gate?

Question 26. What is the state of the algorithm after the U_f gate?

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

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

Question 29. Suppose we measured the first register and observe some random $|z\rangle$. What can we say about the coefficient of such a $|z\rangle$?

This can be analyzed using modular arithmetic:

$$x \cdot z \quad \text{mod } 2 = y \cdot z \quad \text{mod } 2 \tag{11}$$

$$(x - y) \cdot z \quad \text{mod } 2 = 0. \tag{12}$$

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

$$(x \oplus y) \cdot z = s \cdot z = 0. \tag{13}$$

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 $s \cdot z \equiv 0(\text{ mod } 2)$. 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.

$$\begin{bmatrix} --- & z_1 & --- \\ --- & z_2 & --- \\ & \vdots & \\ --- & z_m & --- \end{bmatrix} \cdot \begin{bmatrix} | \\ s \\ | \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \tag{14}$$

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

1.6 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.