

※ 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 110. Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ 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?

Question 111. Suppose that we prepared the state

$$\frac{1}{\sqrt{2^n}}(|00\dots 00\rangle|0\rangle + |00\dots 01\rangle|0\rangle + |00\dots 10\rangle|0\rangle + \dots + |11\dots 10\rangle|0\rangle + |11\dots 11\rangle|0\rangle) \quad (95)$$

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

How do we prepare the state discussed above? Let's start with smaller examples.

Question 112. What is state (95) when $n = 1$? What is the circuit to prepare this state?

Question 113. What is state (95) when $n = 2$? What is the circuit to prepare this state?

Question 114. What is the circuit to prepare (95)?

Proposition 12.1 (*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:

$$H^{\otimes n} |x\rangle = H|x_1\rangle \otimes H|x_2\rangle \otimes \cdots \otimes H|x_n\rangle \quad (96)$$

$$= \frac{(|0\rangle + (-1)^{x_1}|1\rangle)}{\sqrt{2}} \otimes \cdots \otimes \frac{(|0\rangle + (-1)^{x_n}|1\rangle)}{\sqrt{2}} \quad (97)$$

$$= \frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle \quad (98)$$

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

※ Computation 1: Query-based algorithms

13.1 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 $f : \{0, 1\}^n \rightarrow \{0, 1\}$, 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.

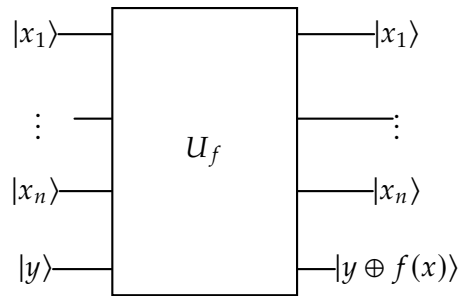
Question 116. What is the query complexity of a classical algorithm to call a function to determine the following properties?

- Is there any input x such that $f(x) = 1$?
- Does $f(x) = 1$ for a majority of the inputs?
- Is f periodic?

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

$$(x, y, 0^k) \rightarrow (x, y \oplus f(x), 0^k). \quad (99)$$

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:



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

$$|x\rangle \rightarrow (-1)^{f(x)} |x\rangle \quad (100)$$

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

Question 117. Show that if we set $|y\rangle = |-\rangle$, we can use the above circuit to implement equation (100).

13.2 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.

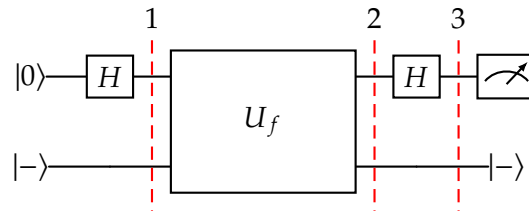
Consider a single bit Boolean function $f : \{0, 1\} \rightarrow \{0, 1\}$. We will denote its output bit for each input as

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

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

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 119. Analyze the state of the circuit at each timestep.

Question 120. 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?

Question 121. 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?

13.3 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:

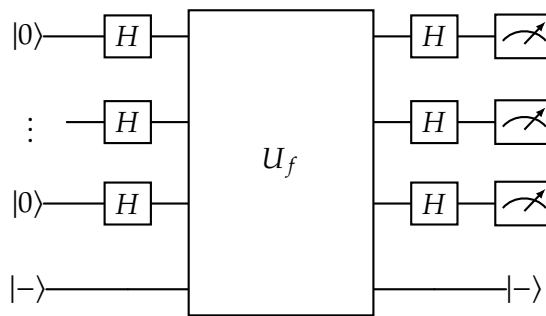
$$f : \{0, 1\}^n \rightarrow \{0, 1\} \tag{101}$$

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

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:

$$H^{\otimes n} |x\rangle = H|x_1\rangle \otimes H|x_2\rangle \otimes \dots \otimes H|x_n\rangle \tag{102}$$

$$= \frac{(|0\rangle + (-1)^{x_1} |1\rangle)}{\sqrt{2}} \otimes \dots \otimes \frac{(|0\rangle + (-1)^{x_n} |1\rangle)}{\sqrt{2}} \tag{103}$$

$$= \frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle \tag{104}$$

Question 123. Using Proposition 11.1, what is the state of the Deutsch-Josza algorithm before the call to U_f ?

Question 124. What is the state of the Deutsch-Josza algorithm after the call to U_f ?

Question 125. Using Proposition 11.1 again, what is the state of the Deutsch-Josza algorithm after the second layer of H gates?

Question 126. What is the amplitude of $|0 \cdots 0\rangle$ if f is constant?

Question 127. What is the amplitude of $|0 \cdots 0\rangle$ if f is balanced?

Question 128. 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.