Worksheet: Deutsch's Algorithm – Computational approach

Deutsch's algorithm, developed by David Deutsch in 1985, marks a significant milestone in the world of computing. It is the first instance where a quantum algorithm showed a potential advantage over classical algorithms. This discovery opened the door to the idea that computers based on the principles of quantum mechanics could solve certain types of problems more efficiently than traditional computers. Deutsch's work laid the foundation for quantum computing, inspiring further research and development in this exciting field.

In this lesson, you will learn how Deutsch's algorithm works. To fully grasp its significance, you will first need to understand how the problem the algorithm solves can be approached using classical computing methods.

The problem

Imagine we have a black box function f which takes an input of a single classical bit of data (0 or 1) and returns an output of another single classical bit of data (0 or 1). There are four possible functions this could be:

$f_1(0) = 0$ $f_1(1) = 0$	$f_2(0) = 1$ $f_2(1) = 1$	$f_3(0) = 0$ $f_3(1) = 1$	$f_4(0) = 1$ $f_4(1) = 0$

The problem is we do not know which function is inside the black box. Our task is to work out which it is just by changing the inputs to the black box and viewing the outputs.

Exercise 1

The "Deutsch Functions" spreadsheet shows how these four functions operate and has a mystery function generator. Click on the "Generate mystery function" button to pick one of the four functions at random. If you change the input in Cell B12 the output of the mystery function will be returned in Cell E12. How can you deduce what the mystery function is by altering the inputs and how many guesses does it take? Justify your answer.

Constant functions versus balanced functions

Suppose now we group the four possible functions into two classes as follows:

Constant functions		Balanced functions	
$f_1(0) = 0$ $f_1(1) = 0$	$f_2(0) = 1$ $f_2(1) = 1$	$f_3(0) = 0$ $f_3(1) = 1$	$f_4(0) = 1$ $f_4(1) = 0$

Exercise 2

Using the spreadsheet again, how many guesses does it take to establish whether the mystery function is either a constant function or a balanced function? How does this compare to finding the actual function itself? Justify your answer.

Summary of classical results

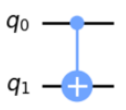
We have seen that for both of these problems it takes two queries to find the desired result using classical search techniques. For the first case (finding the specific function), a quantum computer would also require two queries. However, for the second case (deciding whether the function is constant or balanced), quantum computing can provide a radical improvement.

Building the function on a quantum computer

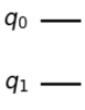
The first step to investigating the problem on a quantum computer is to construct the functions using quantum gates. It is possible to construct all of the four functions using combinations of NOT gates, CNOT gates and no gates at all.

We will need two qubits: q_0 which will take the input of the function and q_1 which will return the output (after being initial set to state |0>).

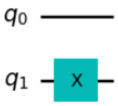
Use the IBM quantum composer to build each of the following four circuits:



a)



b)



c)

d)

Example

For circuit (a), if we set the input qubit q_0 to state |0> and the output qubit q_1 initially to state |0>, we see that the final state of q_1 is also state |0> (as the CNOT gate is not triggered):

This means that an input of 0 for the function is mapped to an output of 0.

If we now change the input qubit q_0 to state |1>, we see that the final state of q_1 is now state |1> (as the CNOT gate is now triggered, flipping the state of q_1):

This means that an input of 1 for the function is mapped to an output of 1.

These mappings correspond to **function f_3** (the first of the balanced functions)

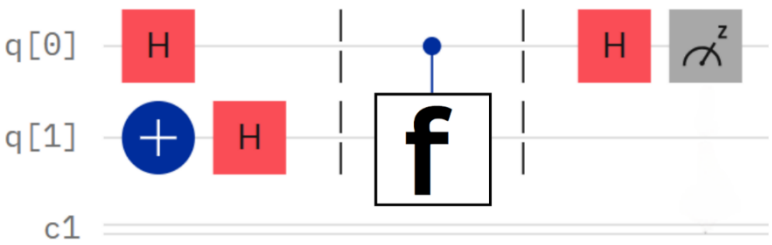
Exercise 3

The spreadsheet “Quantum Circuits” contains each of the four circuits shown. The input state for q_0 is shown in the bold blue cells, the final state for the output qubit q_1 is shown in the bold red cells. By changing the initial inputs to q_0 and measuring the final outputs of q_1, work out which function (f_1, f_2, f_3 and f_4) each of the circuits (b), (c) and (d) corresponds to. Write your answers below:

Finding the mystery function using a quantum computer

The technique we will use here employs superposition through the application of Hadamard gates. This is one of the most important tricks of quantum computing and we will see it used in every quantum search algorithm in this course!

Create the following circuit in the IBM quantum composer:



Note that the NOT gate is used to initialise the state of q_1 to |1> which the absence of a NOT gate on q_0 means this qubit is initialized to state |0>.

Exercise 4

This exercise can be done either using a quantum composer or using the spreadsheet "Quantum Circuits with Hadamards". Initialise q_0 to state |0> and q_1 to |1>, then insert each of the circuits for f_1, f_2, f_3 and f_4 in the section of the circuit marked with the mystery function box **f**. Look at the values that q_0 takes after measurement. How can this information be used to deduce whether a function is balanced or constant? How many queries does it now take to show this? Write your answers below:

Summary of quantum results

By using superposition, we were able to find whether the function was constant or balanced in **one query rather than two**, i.e. a quantum computer can solve this problem twice as fast as a classical computer. There are two important points the Deutsch problem also highlights:

(1) Quantum computers cannot outperform classical computers in every situation. As we have seen here, the quantum algorithm only helped us with the problem of constant versus balanced functions, not which specific function was in the black box. Whether a problem can be solved by a quantum computer always requires careful thought.
(2) The solution to this problem was found by looking at the input qubit rather than the output qubit. Information was gained in the input qubit at the expense of information lost in the output qubit. Quantum computing often involves such trade-offs, and we often have to look for solutions in unusual places!

While Deutsch's problem is a slightly artificial example, it showed the potential for quantum computers and paved the way for more meaningful quantum algorithms that we will explore in the next lessons.