Calculating Hamming distance with the IBM Q Experience

José Manuel Bravo
pestecnologia@gmail.com
Prof. Computer Science and Technology, High School, Málaga, Spain

Abstract

In this brief paper a quantum algorithm to calculate the Hamming distance of two binary strings of equal length (or messages in theory information) is presented. The algorithm calculates the Hamming weight of two binary strings in one query of an oracle. To calculate the hamming distance of these two strings we only have to calculate the Hamming weight of the *xor* operation of both strings. To test the algorithms the quantum computer prototype that IBM has given open access to on the cloud has been used to test the results.

Keywords: quantum algorithm, Hamming weight, Hamming distance

Introduction

The advantage of the quantum computation to solve more efficiently than classical computation some algorithms is one of the most important key for scientists to continue researching and developing news quantum algorithms. Grover´s searching quantum algorithm is able to select one entry between 2^n with output 1 in only O(log(n)) queries to an oracle. Shor´s factoring quantum algorithms is able to factorize one number faster than any classical algorithms known. Deutch-Jozsa algorithm is able to determinate if a function is constant or balanced using only one query to a quantum oracle.

The Hamming weight of two binary strings (messages) of equal length is the number of symbols different of the zero-symbol used in the alphabet. It represents the number of 1´s in the string or the digit sum of the binary representation. The Hamming weight is widely used in different disciplines **[1][2]**, including information theory, coding theory, and cryptography , and his efficient implementation has been widely studied **[3][4][5].**

The Hamming distance in information theory between two strings is the minimum number of substitutions required to change one string into the other. It is used in coding theory to define the error detecting and the error correcting codes. The running time of classical algorithms is proportional to the Hamming distance or to the number of bits in the inputs. Using this quantum algorithm it´s possible to calculate the Hamming distance in only one query to a quantum oracle.

The quantum algorithm calculates the Hamming weight between to binary strings. To calculate the Hamming distance of these two inputs we have to calculate previously the *xor* operation between them and after that to calculate the Hamming weight (*hw*) of the result. Suppose we have to calculate the Hamming distance between the binary string A and B (*hd* (A,B)):

A = 1001 B = 1100

First we calculate A ⊕ B = 0101 and then *hw* (A ⊕ B) = *hd* (A,B) = 2

This quantum algorithms uses quantum parallelism as the fundamental feature to evaluate a function implemented in a quantum oracle for many different values of x at the same time because it exploits the ability of a quantum computer to be in superposition of different states.

Moreover it uses a property of quantum mechanics known as interference. In a quantum computer it is possible for the two alternatives to interfere with one another to yield some global property of a function *f*, by using the Hadamard gate to recombine the different alternatives.

The essence of the design of this algorithm is the choice of a function and a final transformation that allows efficient determination of the Hamming weight of a message (information which cannot be attained quickly on a classical computer).

The quantum algorithm is based on the Deutch-Jozsa algorithm **[6]**. A special function of two binary strings is implemented in an oracle and we only one query to calculate the Hamming weight is needed.

IBM releases in 2016 a 5-qubit universal quantum computer prototype accessible on the cloud. To test the algorithm, two experiments have been simulated on the IBM Q Experience composer.

The Deutsch-Jozsa algorithm

The goal is to find an algorithm which tells if a function *f* is constant or balanced with the least possible number of evaluations of an oracle.

The algorithm is divided into the following steps:

1. Prepare an (n+1)-qubit register in the state $|\Psi_0\rangle = |0\rangle^{\otimes n} \otimes |1\rangle$. First n qubits work as input qubits of the function, while the (n+1)st is an ancilla qubit to store temporary information.
2. Apply the Hadamard transformation to the register. Then we have the state:

$$|\Psi_1\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

3. Perform a quantum transformation *Bf*, associated to a function *f* defined by
$$B_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$$
where *f(x)* is any function
$$f:\{0,1\}^n \rightarrow \{0,1\}$$
Then we obtain the following state.

$$|\Psi_2\rangle = \frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}(-1)^{f(x)}|x\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

4. Apply a Hadamard transformation on the first n qubit.

$$|\Psi_3\rangle = \frac{1}{2^n}\sum_{x,y}^{2^n-1}(-1)^{f(x)+x\cdot y}|y\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$
$$x \cdot y = x_{n-1}y_{n-1} \oplus x_{n-2}y_{n-2} \oplus \ldots \oplus x_0y_0$$

5. The first n qubits are measured. Following the algorithm if the probability that the measurements all give outcome 0, is 1, the function f is constant and balanced is that probability is 0.

How to calculate the Hamming weight

The algorithm to calculate the Hamming weight is based on the Deutsch-Jozsa algorithm so the steps are the same described above.

The function f associated to the transformation *Bf* is the following. Suppose we have to calculate the Hamming weight of a binary string of length *k*. In that case we define a function from $n = log_2(k) + 1$ bits to *1* bit in the following way:

- On the first 2^{n-1} inputs of the function we encode the binary string (message) based on the position of each bit on the string (message).
- On the last 2^{n-1} inputs of the function we fix the binary string with all zero´s.

Suppose we want to calculate the Hamming weight of the string **a = 10**. Then function *f* will be:

$$f:\{0,1\}^{log_2(2)+1} \rightarrow \{0,1\}$$

Input $\|x\rangle$	***F(x)***
00	**0**
01	**1**
10	0
11	0

The transformation Bf:

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

This transformation has associated a unitary permutation so it can be implemented in a quantum computer using a set of universal quantum gates.

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

Then we apply the *Bf* one time and the Hadamard transformations. After the *n* qubits are measured, we can observe:

$$hw(a) = 2^{n-1} \cdot (1 - \sqrt{P|0\rangle}) \qquad (1)$$
$$hw(a) = 2^{n-1} \cdot \sqrt{P|2^{n-1}\rangle} \qquad (2)$$

where $P|2^{n-1}\rangle$ is the probability of measure the state $|2^{n-1}\rangle$ and $P|0\rangle$ is the probability of measure the state $|0\rangle$ for any number n. As you can see if the Hamming weight tell us the number of symbols different of the zero-symbol used in the alphabet, the number $2^{n-1} \cdot (1 - \sqrt{P|0\rangle})$ tell us the number of symbols equal to the zero-symbol in the string.

Proof

If we want to calculate de Hamming weight of the message m of length k *(*$n = log_2(k)$*)*, we define m_L like the first 2^{n-1} bits and m_H the last 2^{n-1} bits of the message:

$$m = m_{2^n-1} \cdots m_0 = m_H \,\&\, m_L$$
$$m_H = m_{2^n-1} \cdots m_{2^{n-1}}$$
$$m_L = m_{2^{n-1}-1} \ldots m_0$$

Following the algorithm m_H will be the binary string with all zero´s. Now consider the summation

$$\sum_{x}^{2^n-1}(-1)^{f(x)+x\cdot y}|y\rangle \qquad (3)$$

for the state $|0\rangle$ and $|2^{n-1}\rangle$ in m_H:

$$\sum_{x \in m_H}(-1)^{0+x\cdot(0\cdots0)}|0\rangle = 2^{n-1} \qquad (4)$$

$$\sum_{x \in m_H}(-1)^{0+x\cdot(10\cdots0)}|2^{n-1}\rangle = -2^{n-1} \qquad (5)$$

Now if we consider the summation (3) for the state $|0\rangle$ and $|2^{n-1}\rangle$ in m_L we will obtain the same result:

$$\sum_{x \in m_L}(-1)^{f(x)+x\cdot(0\cdots0)}|0\rangle = \qquad (6)$$
$$= 2^{n-1} - 2 \cdot hw(m_L)$$

$$\sum_{x\in m_L}(-1)^{f(x)+x\cdot(10\cdots0)}|2^{n-1}\rangle = \qquad (7)$$
$$= 2^{n-1} - 2\cdot hw(m_L)$$

where $hw(m_L)$ is the Hamming weight of the substring m_L. After that we consider the contribution of (4) and (6) to the amplitude associated with the state$|0\rangle$ and we will obtain

$$\frac{1}{2^n}\cdot(2^{n-1} - 2\cdot hw(m_L) + 2^{n-1}) = \qquad (8)$$
$$= 1 - \frac{hw(m_L)}{2^{n-1}}$$

If we consider the contribution of (5) and (7) to the amplitude associated with the state$|2^{n-1}\rangle$, we will obtain

$$\frac{1}{2^n}\cdot(2^{n-1} - 2\cdot hw(m_L) - 2^{n-1}) = \qquad (9)$$
$$= \frac{-hw(m_L)}{2^{n-1}}$$

when the state $|0\rangle$ is measured we will obtain:

$$P|0\rangle = \left(1 - \frac{hw(m_L)}{2^{n-1}}\right)^2 \qquad (8)$$

from where we can easily obtain (1). And if the state $|2^{n-1}\rangle$ is measured we will obtain:

$$P|0\rangle = \left(\frac{-hw(m_L)}{2^{n-1}}\right)^2 \qquad (10)$$

from where we can easily obtain (2).

Algorithm to calculate the Hamming distance

Suppose we have to calculate the Hamming distance of two binary strings A y B of length *k*. The steps are the following:

1. Calculate C = A ⊕ B
2. Build the function *f*

   $$f:\{0,1\}^{n=log_2(k)+1} \to \{0,1\}$$

   that encodes the binary string C in the first *n/2* inputs and the binary string *all zero's* in the last *n/2* inputs.
3. Prepare an (n+1)-qubit register in the state $|\Psi_0\rangle = |0\rangle^{\otimes n}\otimes|1\rangle$. First *n* qubits work as input qubits of the function, while the *(n+1)*st qubit is an ancilla qubit to store temporary information.
4. Apply the steps 2, 3, 4 and 5 of the Deutsch-Jozsa algorithm.
5. The states $|2^{n-1}\rangle$ and $|0\rangle$ are measured:

$$hd\ (\mathrm{A,B}) = hw(\mathrm{C}) = 2^{n-1}\cdot\sqrt{P|2^{n-1}\rangle}$$
$$hd\ (\mathrm{A,B}) = hw(\mathrm{C}) = 2^{n-1}\cdot\left(1-\sqrt{P|0\rangle}\right)$$

Below you can see the probabilities obtained for the Hamming weight ***(hw)*** for different *n* values.

hw	***P*|0⟩**	***P*|2^(n−1)⟩**
0	1,00	0,00
1	0,25	0,25
2	0,00	1,00

Table 1 *n=1*

hw	***P*|0⟩**	***P*|2^(n−1)⟩**
0	1,00000	0,00000
1	0,56250	0,06250
2	0,25000	0,25000
3	0,06250	0,56250
4	0,00000	1,00000

Table 1 *n=2*

hw	***P*|0⟩**	***P*|2^(n−1)⟩**	***hw***	***P*|0⟩**	***P*|2^(n−1)⟩**
0	1,00000	0,00000	5	0,14062	0,39062
1	0,76562	0,01562	6	0,06250	0,56250
2	0,56250	0,06250	7	0,01562	0,76562
3	0,39062	0,14062	8	0,00000	1,00000
4	0,25000	0,25000			

Table 2 *n=3*

hw	***P*|0⟩**	***P*|2^(n−1)⟩**	***hw***	***P*|0⟩**	***P*|2^(n−1)⟩**
0	1,00000	0,00000	9	0,19141	0,31641
1	0,87891	0,00391	10	0,14063	0,39063
2	0,76563	0,01563	11	0,09766	0,47266
3	0,66016	0,03516	12	0,06250	0,56250
4	0,56250	0,06250	13	0,03516	0,66016
5	0,47266	0,09766	14	0,01563	0,76563
6	0,39063	0,14063	15	0,00391	0,87891
7	0,31641	0,19141	16	0,00000	1,00000
8	0,25000	0,25000			

Table 3 *n=4*

Simulations on IBM Quantum Experience

Experiment 1(figure 1): calculate de Hamming distance of the following messages:

$$a = 10100101 \quad b = 01100101$$

In this case *n=4*. The first step is to calculate

$$c = a \oplus b = 11000000$$

and to define the function *f*

Input *(q[3],q[2],q[1],q[0])*	*f*	*Input* *(q[3],q[2],q[1],q[0])*	*f*
0000	0	1000	0
0001	0	1001	0
0010	0	1010	0
0011	0	1011	0
0100	0	1100	0
0101	0	1101	0
0110	1	1110	0
0111	1	1111	0

After define the transformation *Bf* and the oracle, the quantum circuit is showing below.
The results we obtain after 8192 shots are:

State	*Probability*
$\|0\rangle$	0,566
$\|2\rangle$	0,062
$\|4\rangle$	0,066
$\|6\rangle$	0,061
$\|8\rangle$	0,063
$\|10\rangle$	0,058
$\|12\rangle$	0,061
$\|14\rangle$	0,064

$$hd(a,b) = hw(c) = \lfloor\sqrt{0{,}063} \cdot 2^3\rfloor = 2$$

$$hd(a,b) = hw(c) = 2^3 - \lfloor\sqrt{0{,}566} \cdot 2^3\rfloor = 2$$

Experiment 2(figure 2): calculate de Hamming distance of the following messages:

$$a = 1001 \qquad b = 0001$$

In this case *n=3*. The first step is to calculate

$$c = a \oplus b = 1000$$

and to define the function *f*

Input *(q[3],q[2],q[1],q[0])*	*f*	*Input* *(q[3],q[2],q[1],q[0])*	*f*
000	0	100	0
001	0	101	0
010	0	110	0
011	1	111	0

After define the transformation *Bf* and the oracle, the quantum circuit is showing below.
The results we obtain after 8192 shots are:

State	*Probability*
$\|0\rangle$	0,565
$\|1\rangle$	0,066
$\|2\rangle$	0,062
$\|3\rangle$	0,052
$\|4\rangle$	0,066
$\|5\rangle$	0,065
$\|6\rangle$	0,059
$\|7\rangle$	0,065

$$hd(a,b) = hw(c) = \lfloor\sqrt{0{,}066} \cdot 2^2\rfloor = 1$$

$$hd(a,b) = hw(c) = 2^2 - \lfloor\sqrt{0{,}565} \cdot 2^2\rfloor = 1$$

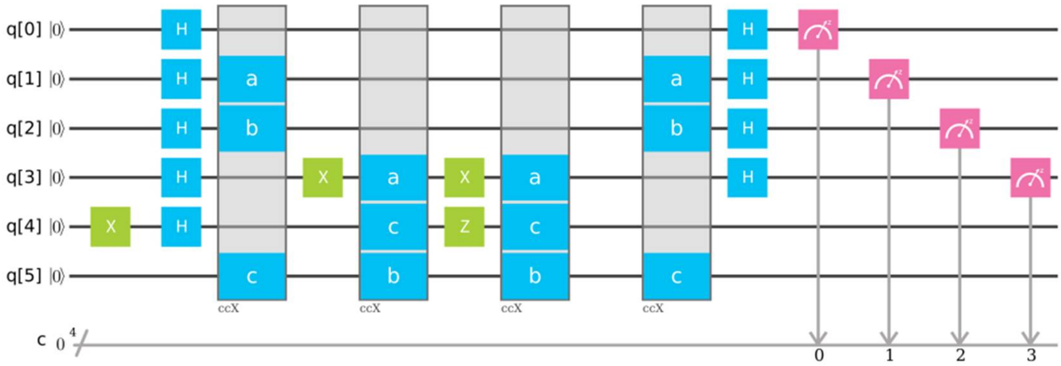

Figure 1 Quantum circuit to calculate de Hamming distance of $a = 10100101$ *and* $b = 01100101$

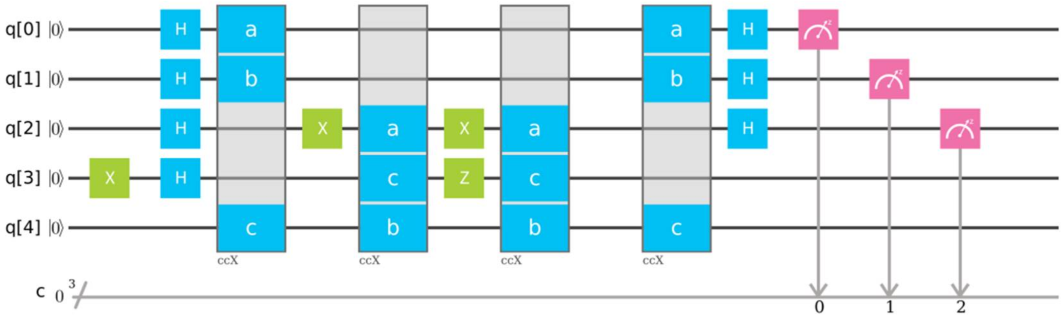

Figure 2 Quantum circuit to calculate de Hamming distance of $a = 1001$ *and* $b = 0001$

Conclusions

This algorithm show one more time the advantage of the quantum computation to solve more efficiently than classical computation some algorithms. As you can see it is able to calculate the Hamming weight of a binary string in just one only query to an oracle. Moreover with a simple classical pre-processing, the *xor* operation between two binary strings, the algorithm calculates the Hamming distance of both. Finally you can see two quantum circuits that implements two examples and the results obtained using IBM Q Experience

Data availability

All relevant data are available within the paper.

Competing interests

The author confirms that there are no known conflicts of interest associated with this publication and there has been no financial support for this work that could have influenced its outcome. The author declares no competing interests.

References

[1] SPARC International, Inc. (1992). "A.41: Population Count. Programming Note". The SPARC architecture manual: version 8 (PDF) (Version 8 ed.). Englewood Cliffs, New Jersey, USA: Prentice Hall. p. 231. ISBN 0-13-825001-4. Archived from the original (PDF) on 2012-01-18.

[2] Blaxell, David (1978), "Record linkage by bit pattern matching", in Hogben, David; Fife, Dennis W., Computer Science and Statistics--Tenth Annual Symposium on the Interface, NBS Special Publication, 503, U.S. Department of Commerce / National Bureau of Standards, pp. 146–156

[3] Muła, Wojciech; Kurz, Nathan; Lemire, Daniel (January 2018), "Faster Population Counts Using AVX2 Instructions", Computer Journal, 61 (1), doi:10.1093/comjnl/bxx046

[4] Knuth, Donald Ervin (2009). "Bitwise tricks & techniques; Binary Decision Diagrams". The Art of Computer Programming. Volume 4, Fascicle 1. Addison–Wesley Professional. ISBN 0-321-58050-8. (NB. Draft of Fascicle 1b available for download.)

[5] Hewlett-Packard HP-16C Computer Scientist Owner's Handbook (PDF). Hewlett-Packard Company. April 1982. 00016-90001. Archived (PDF) from the original on 2017-03-28. Retrieved 2017-03-28

[6] D. Deutsch and R. Jozsa. Rapid solution of problems by quantum computation. Proceedings of the Royal Society of London A, 439:553–558, Jan 1992.

[7] Quantum information science. Michael Nielsen and Isaac Chuang. Cambridge University Press. ISBN 978-1-107-00217-3

[8] Quantum Computing. From Linear Algebra to Physical Realizations. Mikio Nakahara and Tetsuo Ohmi. CRC Press. ISBN 978-0-7503-0983-7