

# **An Introduction to Quantum Algorithms**

**Emma Strubell**  
COS498 – Chawathe  
Spring 2011

# Contents

<b>1</b>	<b>What are quantum algorithms?</b>	<b>3</b>
1.1	Background . . . . .	3
1.2	Caveats . . . . .	4
<b>2</b>	<b>Mathematical representation</b>	<b>5</b>
2.1	Fundamental differences . . . . .	5
2.2	Hilbert spaces and Dirac notation . . . . .	6
2.3	The qubit . . . . .	9
2.4	Quantum registers . . . . .	11
2.5	Quantum logic gates . . . . .	12
2.6	Computational complexity . . . . .	19
<b>3</b>	<b>Grover's Algorithm</b>	<b>20</b>
3.1	Quantum search . . . . .	20
3.2	Grover's algorithm: How it works . . . . .	22
3.3	Grover's algorithm: Worked example . . . . .	24
<b>4</b>	<b>Simon's Algorithm</b>	<b>28</b>
4.1	Black-box period finding . . . . .	28
4.2	Simon's algorithm: How it works . . . . .	29
4.3	Simon's Algorithm: Worked example . . . . .	32
<b>5</b>	<b>Conclusion</b>	<b>33</b>
	<b>References</b>	<b>34</b>

# 1 What are quantum algorithms?

## 1.1 Background

The idea of a quantum computer was first proposed in 1981 by Nobel laureate Richard Feynman, who pointed out that accurately and efficiently simulating quantum mechanical systems would be impossible on a classical computer, but that a new kind of machine, a computer itself “built of quantum mechanical elements which obey quantum mechanical laws” [1], might one day perform efficient simulations of quantum systems. Classical computers are inherently unable to simulate such a system using sub-exponential time and space complexity due to the exponential growth of the amount of data required to completely represent a quantum system. Quantum computers, on the other hand, exploit the unique, non-classical properties of the quantum systems from which they are built, allowing them to process exponentially large quantities of information in only polynomial time. Of course, this kind of computational power could have applications to a multitude of problems outside quantum mechanics, and in the same way that classical computation quickly branched away from its narrow beginnings facilitating simulations of Newtonian mechanics, the study of quantum algorithms has diverged greatly from simply simulating quantum physical systems to impact a wide variety of fields, including information theory, cryptography, language theory, and mathematics.

Probably the most widely known development in quantum computation was Peter Shor’s 1997 publication of a quantum algorithm for performing prime factorization of integers in essentially polynomial time [2]. Shor’s algorithm was a monumental discovery not only because it provides exponential speedup over the fastest classical algorithms, but because a number of algorithms for public-key cryptography, including the commonly used RSA algorithm, depend on the fact that there is no known efficient classical algorithm to factor integers into prime numbers [3]. Shor demonstrated that the realization of a full-scale quantum computer would have the potential to provide a truly significant increase in computing speed, at the same time pointing out the possible implications of such an increase in computational power to the field of cybersecurity. For both reasons, Shor’s discovery sparked a great deal of interest in the design of quantum algorithms and computers that endures today.

In addition to Shor’s algorithm, there is a wealth of other interesting and important algorithms that have been developed for quantum computers. Two of those algorithms will be described in detail in this tutorial in order to better elucidate the study of quantum computing theory and quantum algorithm design. These two algorithms are good models for our current understanding of quantum computation as many other quantum algorithms use similar techniques to achieve their results, whether they be algorithms to solve linear systems of equations [4], or quickly compute discrete logarithms. Shor, for example, credits one of these algorithms as the inspiration for his own [2].

The first algorithm that will be explored in this tutorial is Lov Grover's quantum database search [5]. Grover's algorithm searches for a specified entry in an unordered database, employing an important technique in quantum algorithm design known as amplitude amplification to achieve a polynomial speedup over the best classical algorithms. In fact, Grover's algorithm is optimal for any quantum algorithm for performing such a search [6]. Of course, searching for a unique element in an unordered set can be generalized to apply to a wide variety of problems in computer science, such as the problem of boolean satisfiability (SAT). Determining whether there exists a set of truth values that satisfy a given set of clauses is equivalent to searching for an element, the set of truth values, that satisfies a certain condition, the set of clauses, in an unordered search space, the set of all  $2^n$  possible truth assignments of  $n$  boolean variables. Although this algorithm does not provide an extreme increase in efficiency over its classical counterparts, the speedup is still certainly significant with large input.

The second algorithm that this tutorial will present is Daniel Simon's algorithm for determining the exclusive-or (XOR) mask over which a given black-box function is invariant [7]. Simon's was the first quantum algorithm found to have exponential speedup over any equivalent classical algorithm, and the runtime of his algorithm is optimal [8]. Although the problem may at first seem very abstract, Simon's problem corresponds to an important problem with a number of applications in computer science known as the *hidden Abelian subgroup problem*, which includes factoring integers into primes and calculating discrete logarithms [9].

## 1.2 Caveats

Now, if quantum computers are so much faster than classical machines, how is it that their use is not yet widespread? Unfortunately, the engineering required to construct a quantum computer is very difficult, although it has come a long way. For example, the largest number that has been factored by a quantum computer using Shor's algorithm is 15, and the circuit was hard-wired to factor only the number 15, not any other input, as the algorithm is designed to do. To put this in perspective, in order for Shor's algorithm to break 1024-bit RSA keys, the length currently recommended for corporate use [10], the algorithm would have to be applied to a 1024-bit integer, which would require a quantum computer orders of magnitude larger than that needed to factor 15 into 5 and 3.

Engineers continue to experiment with many different physical implementations of quantum computers, the details of which are beyond the scope of this tutorial. Currently, the most popular implementation, known as an ion trap quantum computer, works by using lasers to change the spin states of supercooled ions trapped in an electromagnetic field. This implementation has led to some of the largest quantum machines, boasting registers of up to 8 quantum bits (qubits) [11]. The difficulty faced by all of the architec-

tures considered to date is foremost that of creating a machine that is minimally effected by its surroundings, while at the same time responsive to control by the engineer, or user. The quality of a quantum computer implementation is measured in terms of its *decoherence*, error caused by the inherent coupling of artificial quantum systems to the unpredictable natural world that we use to construct them. For example, in the ion trap implementation, possible sources of undesirable interference are any electric and magnetic fields in the vicinity, including those necessary to operate the machine itself, which can cause the ions to change spin states at random and ultimately collapse into a classical system [12]. Current research focuses on minimizing such sources of decoherence, but eliminating decoherence completely is impossible. So, while it is currently possible to decrease decoherence enough to allow for relatively accurate computation on the scale of 8 qubits, as the number of qubits increases, so does the decoherence, to the extent that properly functioning quantum computers with more than 8 qubits have yet to be realized.

Still, many universities and corporate research centers around the world are working to build the first scalable quantum computer, as quantum computation, with its promises of speed proven unattainable by classical machines, remains a good candidate for the next computational paradigm. Even if general purpose quantum computers might seem a long way off, it is likely that small scale quantum computers might soon be used by classical computers to perform certain calculations more quickly, such as the calculations required to simulate complex biological systems, or to identify the content of a raster image for improved web search [13]. For these reasons, it is important to become familiar with the basics of quantum computation now in order to be prepared for the quantum computational tools that will likely be available in the not so distant future.

## 2 Mathematical representation

### 2.1 Fundamental differences

Quantum computers employ the laws of quantum mechanics to provide a vastly different mechanism for computation than that available from classical machines. Fortunately for computer scientists interested in the field of quantum computing, a deep knowledge of quantum physics is not a prerequisite for understanding quantum algorithms, in the same way that one need not know how to build a processor in order to design classical algorithms. However, it is still important to be familiar with the basic concepts that differentiate quantum mechanical systems from classical ones in order to gain a better intuitive understanding of the mathematics of quantum computation, as well as of the algorithms themselves.

The first distinguishing trait of a quantum system is known as *superposition*, or more formally the *superposition principle of quantum mechanics*. Rather than existing in one distinct state at a time, a quantum system is actually in all of its possible states at the

same time. With respect to a quantum computer, this means that a quantum register exists in a superposition of all its possible configurations of 0's and 1's at the same time, unlike a classical system whose register contains only one value at any given time. It is not until the system is observed that it collapses into an observable, definite classical state.

It is still possible to compute using such a seemingly unruly system because probabilities can be assigned to each of the possible states of the system. Thus a quantum system is *probabilistic*: there is a computable probability corresponding to the likelihood that that any given state will be observed if the system is measured. Quantum computation is performed by increasing the probability of observing the correct state to a sufficiently high value so that the correct answer may be found with a reasonable amount of certainty.

Quantum systems may also exhibit *entanglement*. A state is considered entangled if it cannot be decomposed into its more fundamental parts. In other words, two distinct elements of a system are entangled if one part cannot be described without taking the other part into consideration. In a quantum computer, it is possible for the probability of observing a given configuration of two qubits to depend on the probability of observing another possible configuration of those qubits, and it is impossible to describe the probability of observing one configuration without considering the other. An especially interesting quality of quantum entanglement is that elements of a quantum system may be entangled even when they are separated by considerable space. The exact physics of quantum entanglement remain elusive even to professionals in the field, but that has not stopped them from applying entanglement to quantum information theory. Quantum teleportation, an important concept in the field of quantum cryptography, relies on entangled quantum states to send quantum information adequately accurately and over relatively long distances<sup>1</sup>.

## 2.2 Hilbert spaces and Dirac notation

Before delving into the formal mathematics used to describe quantum algorithms, it is important to first become familiar with the notation that is commonly used to describe more general quantum mechanical systems, as well as the states of quantum registers. This notation is known as Bra-ket or Dirac notation, named after the Nobel laureate Paul Dirac. Dirac notation is just another way of describing vectors. In dirac notation:

$$\mathbf{v} = \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_n \end{bmatrix} = |\mathbf{v}\rangle$$

---

<sup>1</sup>As of May 2010, researchers were able to teleport quantum information over a distance of about 16 kilometers with 89% accuracy [14].

The column vector  $|\mathbf{v}\rangle$  is referred to as “ket- $v$ .” The *dual vector* of  $|\mathbf{v}\rangle$  has the following Dirac notation:

$$\langle \mathbf{v} | = \overline{\mathbf{v}^T} = [\overline{v_0} \quad \overline{v_1} \quad \dots \quad \overline{v_n}]$$

Where  $\bar{v}$  is the complex conjugate<sup>2</sup> of  $v$ . This dual vector is called “bra- $v$ .”

Dirac notation is a convenient way to describe vectors in the *Hilbert space*  $\mathbb{C}^n$ , which is the vector space that is most useful for reasoning about quantum systems. For the purposes of this tutorial<sup>3</sup>, a Hilbert space is a vector space with an *inner product*, and a norm<sup>4</sup> defined by that inner product. The inner product of a vector space is an operation that assigns a scalar value to each pair of vectors  $\mathbf{u}$  and  $\mathbf{v}$  in the vector space, and the inner product of two vectors in a Hilbert space in  $\mathbb{C}^n$  is denoted using the Dirac notation  $\langle \mathbf{u} | \mathbf{v} \rangle$ . Thus the inner product  $\langle \mathbf{u} | \mathbf{v} \rangle$  of two vectors in a complex Hilbert space is given by the dot product of the vectors  $\mathbf{v}$  and  $\overline{\mathbf{u}^T}$ , the conjugate transpose of  $\mathbf{u}$ :

$$\langle \mathbf{u} | \mathbf{v} \rangle = \overline{\mathbf{u}^T} \mathbf{v} = [\overline{u_0} \quad \overline{u_1} \quad \dots \quad \overline{u_n}] \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_n \end{bmatrix} = \overline{u_0} \cdot v_0 + \overline{u_1} \cdot v_1 + \dots + \overline{u_n} \cdot v_n$$

By definition, the inner product satisfies the following conditions:

1.  $\langle \mathbf{v} | \mathbf{v} \rangle \geq 0$ , with  $\langle \mathbf{v} | \mathbf{v} \rangle = 0$  if and only if  $|\mathbf{v}\rangle = \mathbf{0}$ .
2.  $\langle \mathbf{u} | \mathbf{v} \rangle = \overline{\langle \mathbf{v} | \mathbf{u} \rangle}$  for all  $|\mathbf{u}\rangle, |\mathbf{v}\rangle$  in the vector space.
3.  $\langle \mathbf{u} | \alpha_0 \mathbf{v} + \alpha_1 \mathbf{w} \rangle = \alpha_0 \langle \mathbf{u} | \mathbf{v} \rangle + \alpha_1 \langle \mathbf{u} | \mathbf{w} \rangle$ .

More generally, the inner product of  $|\mathbf{u}\rangle$  and  $\sum_i \alpha_i |\mathbf{v}_i\rangle$  is equal to  $\sum_i \alpha_i \langle \mathbf{u} | \mathbf{v}_i \rangle$  for all scalars  $\alpha_i$  and vectors  $|\mathbf{u}\rangle, |\mathbf{v}\rangle$  in the vector space (this is known as *linearity in the second argument*).

The norm of a vector  $|\mathbf{v}\rangle$  in a Hilbert space is defined using the square root of the inner product of  $|\mathbf{v}\rangle$  with itself:

$$\| |\mathbf{v}\rangle \| = \sqrt{\langle \mathbf{v} | \mathbf{v} \rangle}$$

---

<sup>2</sup>The notation  $v^*$  is also often used to denote the complex conjugate, though more so in applied sciences such as engineering than in mathematics or computer science.

<sup>3</sup>For finite-dimensional vector spaces, such as the vector space  $\mathbb{C}^n$  with which this tutorial is concerned, Hilbert spaces are no different than inner product spaces. Formally, a Hilbert space must be an inner product space that is also a *complete metric space*: the inner product of a Hilbert space  $H$  must induce a norm such that for all sets of vectors in  $H$ , if  $\sum_{i=0}^{\infty} \|\mathbf{v}_i\|$  converges absolutely, then it converges to a value in  $H$  (every Cauchy sequence converges in  $H$ ). More intuitively,  $H$  must be defined at every point, so that calculus is a valid set of operations in  $H$ .

<sup>4</sup>The norm is the length of a vector in a vector space, or geometrically, the distance from the origin to the point that the vector represents.

This norm is also known as the  $\ell^2$ -norm, 2-norm or Euclidean norm over the complex numbers. Geometrically, this norm gives the distance from the origin to the point  $|\mathbf{v}\rangle$  that follows from the Pythagorean theorem.

Along with the inner product, the *outer product* of two vectors can also be defined. The outer product, denoted  $|\mathbf{v}\rangle\langle\mathbf{u}|$  or  $|\mathbf{v}\rangle\otimes\langle\mathbf{u}|$ , is the *tensor* or *Kronecker product*  $|\mathbf{v}\rangle$  with the conjugate transpose of  $|\mathbf{u}\rangle$ . The result is not a scalar, as with the inner product, but a matrix:

$$|\mathbf{v}\rangle\langle\mathbf{u}| = \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_n \end{bmatrix} \begin{bmatrix} \overline{u_0} & \overline{u_1} & \dots & \overline{u_m} \end{bmatrix} = \begin{bmatrix} v_0\overline{u_0} & v_0\overline{u_1} & \dots & v_0\overline{u_m} \\ v_1\overline{u_0} & v_1\overline{u_1} & \dots & v_1\overline{u_m} \\ \vdots & \vdots & \ddots & \vdots \\ v_n\overline{u_0} & v_n\overline{u_1} & \dots & v_n\overline{u_m} \end{bmatrix}$$

The outer product is often used to describe a linear transformation between vector spaces. Rather than using inelegant transformation matrices like the one depicted above, a linear transformation from a Hilbert space  $U$  to another Hilbert space  $V$  on a vector  $|\mathbf{w}\rangle$  in  $U$  may be succinctly and conveniently described:

$$(|\mathbf{v}\rangle\langle\mathbf{u}|)|\mathbf{w}\rangle = |\mathbf{v}\rangle\langle\mathbf{u}|\mathbf{w}\rangle = \langle\mathbf{u}|\mathbf{w}\rangle|\mathbf{v}\rangle$$

Since  $\langle\mathbf{u}|\mathbf{w}\rangle$  is a commutative, scalar value.

The tensor product, a way of combining vector spaces into larger vector spaces, is a very important operation in quantum computation. The tensor product between two vectors  $|\mathbf{u}\rangle\otimes|\mathbf{v}\rangle$  is so common that it is usually simplified to  $|\mathbf{u}\rangle|\mathbf{v}\rangle$  or  $|\mathbf{uv}\rangle$ , and a vector tensored with itself  $n$  times is denoted  $|\mathbf{v}\rangle^{\otimes n}$ . Two column vectors  $|\mathbf{u}\rangle$  and  $|\mathbf{v}\rangle$  of lengths  $m$  and  $n$  yield a column vector of length  $m \cdot n$  when tensored:

$$|\mathbf{u}\rangle|\mathbf{v}\rangle = |\mathbf{uv}\rangle = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_m \end{bmatrix} \otimes \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} u_0 \cdot v_0 \\ u_0 \cdot v_1 \\ \vdots \\ u_0 \cdot v_n \\ u_1 \cdot v_0 \\ \vdots \\ u_{m-1} \cdot v_n \\ u_m \cdot v_0 \\ \vdots \\ u_m \cdot v_n \end{bmatrix}$$

Tensor products are so ubiquitous because they are the mathematical abstraction that describes the interaction between two quantum systems: the vector space describing one quantum system tensored with the vector space describing another is the vector space

made up of linear combinations of all of the vectors in the two vector spaces. The applications of the tensor product to quantum computation will be considered in more depth in Section 2.4, which discusses quantum registers, large quantum systems made up of their combined smaller qubit counterparts.

## 2.3 The qubit

A classical bit may be represented as a base-2 number that takes either the value 1 or the value 0. Qubits are represented in a similar way in that they are also base-2 numbers, and they take on the value 1 or 0 when measured and thus collapsed to a classical state. Very much unlike classical bits, in its uncollapsed, quantum state, a qubit is in a superposition of the measurable values 1 and 0. The most convenient way to mathematically represent the state of a qubit at any given time is as a two-dimensional *state space* in  $\mathbb{C}^2$  with orthonormal<sup>5</sup> basis vectors  $|0\rangle$  and  $|1\rangle$ . The superposition  $|\psi\rangle$  of a qubit is represented as a linear combination of those basis vectors:

$$|\psi\rangle = a_0 |0\rangle + a_1 |1\rangle$$

Where  $a_0$  is the complex scalar *amplitude* of measuring  $|0\rangle$ , and  $a_1$  the amplitude of measuring the value  $|1\rangle$ . Amplitudes may be thought of as “quantum probabilities” in that they represent the chance that a given quantum state will be observed when the superposition is collapsed. The most fundamental difference between probabilities of states in classical probabilistic algorithms and amplitudes of states in quantum algorithms is that amplitudes are represented by complex numbers, while traditional probabilities are described by real numbers. Complex numbers are required to fully describe the superposition of states and interference or entanglement inherent in quantum systems.<sup>6</sup> It follows that, as the probabilities of a classical system must sum to 1 in order for the probabilities to form a complete probability distribution, the squares of the absolute values of the amplitudes of states in a quantum system must similarly add up to 1.

Of course, just as the hardware underlying the bits of a classical computer may vary in voltage, quantum systems are not usually so perfectly behaved. It can be assumed that those voltages correspond to discrete boolean values, and a similar assumption is made about quantum state vectors that simplifies reasoning about quantum computational systems, called the *normalization condition*:  $|\psi\rangle$  is a unit vector. This means that  $\| |\psi\rangle \| = \langle \psi | \psi \rangle = 1$ , and since  $|0\rangle$  and  $|1\rangle$  are orthonormal<sup>7</sup>, it follows that  $|a_0|^2 + |a_1|^2 = 1$ .

---

<sup>5</sup>The basis vectors are mutually orthogonal unit vectors.

<sup>6</sup>See [15] for a great discussion by Scott Aaronson of why complex numbers and the 2-norm are used to describe quantum mechanical systems.

<sup>7</sup>If  $|0\rangle$  and  $|1\rangle$  are orthonormal, then by orthogonality  $\langle 0|1\rangle = \langle 1|0\rangle = 0$ , and by normality  $\langle 0|0\rangle = \langle 1|1\rangle = 1$

Reducing the inner product to its component vectors:

$$\begin{aligned}
1 &= \langle \psi | \psi \rangle \\
&= (\overline{a_0} \langle 0| + \overline{a_1} \langle 1|) \cdot (a_0 |0\rangle + a_1 |1\rangle) \\
&= (\overline{a_0} [\overline{\psi_{00}} \quad \overline{\psi_{01}}] + \overline{a_1} [\overline{\psi_{10}} \quad \overline{\psi_{11}}]) \cdot \left( a_0 \begin{bmatrix} \psi_{00} \\ \psi_{01} \end{bmatrix} + a_1 \begin{bmatrix} \psi_{10} \\ \psi_{11} \end{bmatrix} \right) \\
&= [\overline{a_0 \psi_{00}} + \overline{a_1 \psi_{10}} \quad \overline{a_0 \psi_{01}} + \overline{a_1 \psi_{11}}] \cdot \begin{bmatrix} a_0 \psi_{00} + a_1 \psi_{10} \\ a_0 \psi_{01} + a_1 \psi_{11} \end{bmatrix} \\
&= \overline{a_0 \psi_{00}} a_0 \psi_{00} + \overline{a_1 \psi_{10}} a_0 \psi_{00} + \overline{a_0 \psi_{00}} a_1 \psi_{10} + \overline{a_1 \psi_{10}} a_1 \psi_{10} \\
&\quad + \overline{a_0 \psi_{01}} a_0 \psi_{01} + \overline{a_1 \psi_{11}} a_0 \psi_{01} + \overline{a_0 \psi_{01}} a_1 \psi_{11} + \overline{a_1 \psi_{11}} a_1 \psi_{11} \\
&= |a_0|^2 (|\psi_{00}|^2 + |\psi_{01}|^2) + |a_1|^2 (|\psi_{10}|^2 + |\psi_{11}|^2) \\
&\quad + \overline{a_1} a_0 (\overline{\psi_{10}} \psi_{00} + \overline{\psi_{11}} \psi_{01}) + \overline{a_0} a_1 (\overline{\psi_{00}} \psi_{10} + \overline{\psi_{01}} \psi_{11}) \\
&= |a_0|^2 + |a_1|^2
\end{aligned}$$

This derivation is exemplary of the convenience of Dirac notation; the same equivalence is made clear in much fewer steps and through much less tedious math in Dirac notation:

$$\begin{aligned}
1 &= \langle \psi | \psi \rangle \\
&= (\overline{a_0} \langle 0| + \overline{a_1} \langle 1|) \cdot (a_0 |0\rangle + a_1 |1\rangle) \\
&= |a_0|^2 \langle 0|0\rangle + |a_1|^2 \langle 1|1\rangle + \overline{a_1} a_0 \langle 1|0\rangle + \overline{a_0} a_1 \langle 0|1\rangle \\
&= |a_0|^2 + |a_1|^2
\end{aligned}$$

Thus the probability of observing a single possible state from the superposition is obtained by squaring the absolute value of its amplitude; the probability of the qubit being in the state  $|0\rangle$  is  $|a_0|^2$ , and the probability that the qubit will be measured as  $|1\rangle$  is  $|a_1|^2$ , or  $1 - |a_0|^2$ .

Remember that the contents of Dirac bras and kets are labels that describe the underlying vectors.  $|0\rangle$  and  $|1\rangle$  may be transformed into any two vectors that form an orthonormal basis in  $\mathbb{C}^2$ . The most common basis used in quantum computing is called the *computational basis*:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

But any other orthonormal basis could be used. For example, the basis vectors:

$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}, |-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Provide a slightly different but equivalent way of expressing of a qubit:

$$\begin{aligned} |\psi\rangle &= a_0 |0\rangle + a_1 |1\rangle \\ &= a_0 \frac{|+\rangle + |-\rangle}{\sqrt{2}} + a_1 \frac{|+\rangle - |-\rangle}{\sqrt{2}} \\ &= \frac{a_0 + a_1}{\sqrt{2}} |+\rangle + \frac{a_0 - a_1}{\sqrt{2}} |-\rangle \end{aligned}$$

Here, instead of measuring the states  $|0\rangle$  and  $|1\rangle$  each with respective probabilities  $|a_0|^2$  and  $|a_1|^2$ , the states  $|+\rangle$  and  $|-\rangle$  would be measured with probabilities  $|a_0 + a_1|^2/2$  and  $|a_0 - a_1|^2/2$ . Because the computational basis tends to be the most straightforward basis for computing and understanding quantum algorithms, the rest of this tutorial will assume the computational basis is being used unless otherwise stated.

## 2.4 Quantum registers

It is hard to do any interesting computation with only a single qubit. Like classical computers, quantum computers use quantum registers made up of multiple qubits. When collapsed, quantum registers are bit strings whose length determines the amount of information they can store. In superposition, each qubit in the register is in a superposition of  $|1\rangle$  and  $|0\rangle$ , and consequently a register of  $n$  qubits is in a superposition of all  $2^n$  possible bit strings that could be represented using  $n$  bits. The state space of a size- $n$  quantum register is a linear combination of  $n$  basis vectors, each of length  $2^n$ :

$$|\psi_n\rangle = \sum_{i=0}^{2^n-1} a_i |i\rangle$$

Here  $i$  is the base-10 integer representation of a length- $n$  number in base-2. A three-qubit register would thus have the following expansion:

$$|\psi_2\rangle = a_0 |000\rangle + a_1 |001\rangle + a_2 |010\rangle + a_3 |011\rangle + a_4 |100\rangle + a_5 |101\rangle + a_6 |110\rangle + a_7 |111\rangle$$

Or in vector form, using the computational basis:

$$|\psi_2\rangle = a_0 \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + a_1 \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + a_2 \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + a_3 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + a_4 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + a_5 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} + a_6 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} + a_7 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

As mentioned earlier, each possible bit configuration in the quantum superposition is denoted by the tensor product of its counterpart qubits. Consider  $|101\rangle$ , the bit string that represents the integer value 5:

$$\begin{aligned} |101\rangle &= |1\rangle \otimes |0\rangle \otimes |1\rangle \\ &= \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\ &= [0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0]^T \end{aligned}$$

As with single qubits, the squared absolute value of the amplitude associated with a given bit string is the probability of observing that bit string upon collapsing the register to a classical state, and the the squares of the absolute values of the amplitudes of all  $2^n$  possible bit configurations of an  $n$ -bit register sum to unity:

$$\sum_{i=0}^{2^n-1} |a_i|^2 = 1$$

Quantum registers are a relatively straightforward extension of quantum bits.

## 2.5 Quantum logic gates

Understanding how quantum registers work mathematically and how they differ from classical registers, it is now possible to think about how the state of a quantum register can evolve over time, changing to reach some ultimate goal. In classical computing, one way of thinking about algorithm design and computation is via universal Turing machines. Quantum universal Turing machines were first described by David Deutsch in 1985 [16], but designing algorithms for quantum Turing machines is even more difficult and tedious than doing so for classical Turing machines; both a quantum Turing machine's tape and its read-write head exist in superpositions of an exponential number states! As in classical computer science, instead of using the Turing machine as a computational model, operations on a quantum computer are most often described using quantum circuits made up of qubits and quantum logic gates, a concept also introduced by Deutsch a few years after his specification of the quantum analog to a Turing machine [17]. In classical computer science but arguably even more so in quantum computing, although circuits are computationally equivalent to Turing machines, they are usually much simpler to depict, manipulate and understand.

In classical computing, binary values, as stored in a register, pass through logic gates that, given a certain binary input, produce a certain binary output. Mathematically, classical logic gates are described using boolean algebra. Quantum logic gates act in a similar way, in that quantum logic gates applied to quantum registers map the quantum superposition to another, together allowing the evolution of the system to some desired

final state, a correct answer.

Quantum logic gates are mathematically represented as transformation matrices, or linear operators, applied to a quantum register by tensoring the transformation matrix with the matrix representation of the register. All linear operators that correspond to quantum logic gates must be *unitary*. That is, if a complex matrix  $U$  is unitary, then it must be true that  $U^{-1} = U^\dagger$ , where  $U^\dagger$  is the conjugate transpose:  $U^\dagger = \overline{U}^T$ . It follows that  $UU^\dagger = U^\dagger U = I$ . Unitary operators preserve the inner product of two vectors, geometrically preserving the lengths of the vectors and the angle between them:

$$\langle \mathbf{u} | U^\dagger U | \mathbf{v} \rangle = \langle \mathbf{u} | I | \mathbf{v} \rangle = \langle \mathbf{u} | \mathbf{v} \rangle$$

The composition of two unitary operators is also unitary. Given unitary transformation matrices  $U$  and  $V$ :

$$(UV)^\dagger = V^\dagger U^\dagger = V^{-1} U^{-1} = (UV)^{-1}$$

Unitary transformations performed on a single qubit may be visualized as rotations and reflections about the  $x$ ,  $y$ , and  $z$  axes of the *Bloch sphere*. All the possible linear combinations  $a_0 |0\rangle + a_1 |1\rangle$  in  $\mathbb{C}^2$  correspond to all the points  $(\theta, \psi)$  on the surface of the unit sphere, where  $a_0 = \cos(\theta/2)$  and  $a_1 = e^{i\phi} \sin(\theta/2)$ :

$$\begin{aligned} |\psi\rangle &= a_0 |0\rangle + a_1 |1\rangle \\ &= \cos \frac{\theta}{2} |0\rangle + e^{i\phi} \sin \frac{\theta}{2} |1\rangle \\ &= \cos \frac{\theta}{2} |0\rangle + (\cos \phi + i \sin \phi) \sin \frac{\theta}{2} |1\rangle \end{aligned}$$

While circuit diagrams facilitate understanding the flow of operations in quantum algorithms and the bigger picture of how they work to produce the solution to a problem, the Bloch sphere provides an intuitive reminder of the underlying implementation that drives the strange properties that distinguish quantum computing from classical on a gate-to-gate basis. In an actual quantum computer, the “wires” of a circuit diagram could correspond to a single electron moving through time, and each “gate” a change in the pattern of movement of that electron. Since quantum gates are so tightly coupled to their underlying quantum mechanical systems, it is important to know how those gates are evolving the quantum system to produce a result.

Although any unitary transformation is a valid elementary operation on a quantum computer, there are a small number of frequently used quantum logic gates analogous to classical logic gates such as NOT or XOR in the role that they play in the field of quantum computing. In fact, any unitary transformation can be performed using a few of

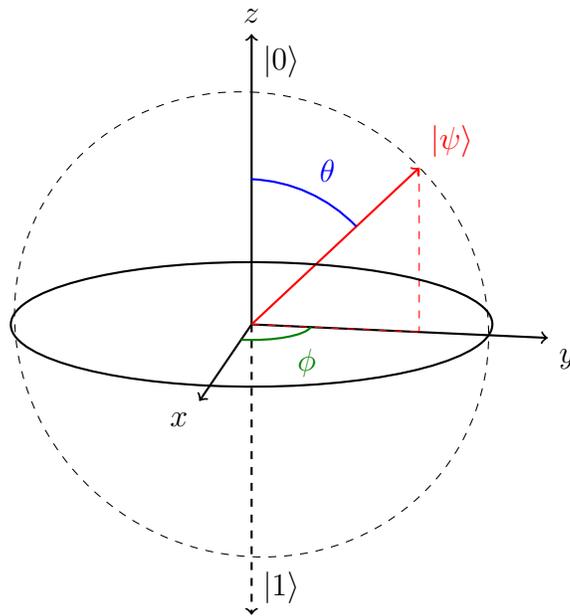


Figure 1: The Bloch sphere

the following gates [18]. One important example is the single-qubit *Hadamard operator*:

$$\text{---} \boxed{H} \text{---} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \langle 0| + \frac{|0\rangle - |1\rangle}{\sqrt{2}} \langle 1|$$

Often referred to as a “fair coin flip,” the Hadamard operator applied to a qubit with the value  $|0\rangle$  or  $|1\rangle$  will induce an equal superposition of the states  $|0\rangle$  and  $|1\rangle$ , an equal probability of the qubit being in the state  $|1\rangle$  or  $|0\rangle$  when observed. Many quantum algorithms begin by applying the Hadamard operator to each qubit in a register, which gives each of the  $2^n$  possible bitwise configurations of the  $n$  qubits an equal probability of  $2^{-n}$  of being observed when the system is measured. Geometrically, the Hadamard operator performs a rotation of  $\pi/2$  about the  $y$  axis followed by a rotation about the  $x$  axis by  $\pi$  radians on the Bloch sphere:

The difference between amplitudes and simple probabilities discussed in Section 2.3 becomes apparent when operations are applied to quantum systems which may have the same probability distributions, but different amplitudes from which those probabilities are derived. For example, consider the results of applying the Hadamard transform to  $|0\rangle$

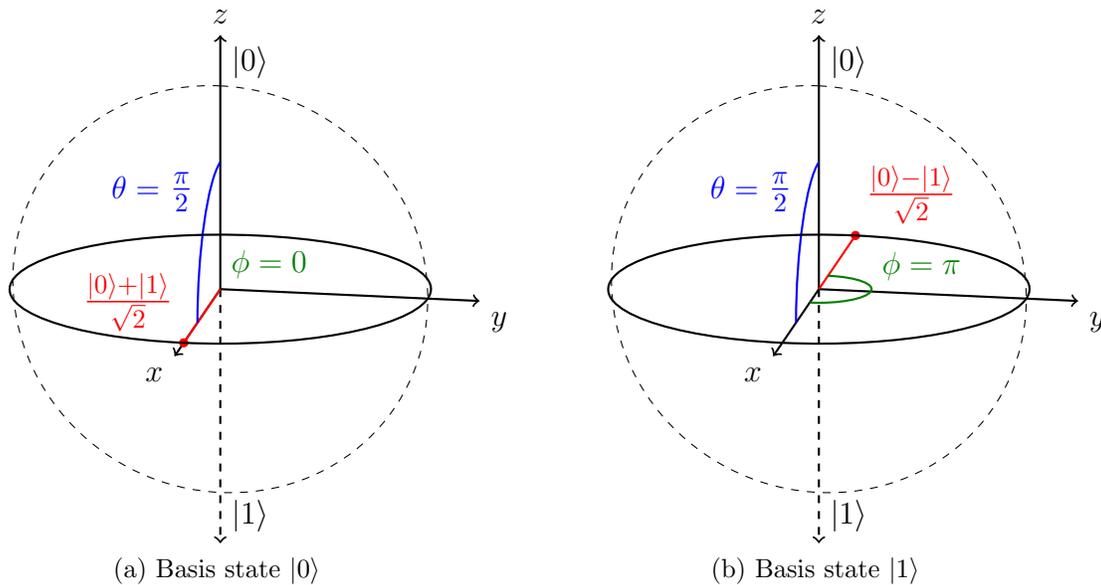


Figure 2: Bloch sphere representation of the Hadamard operator applied to  $|0\rangle$  and  $|1\rangle$

versus  $|1\rangle$  :

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

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

The probability distributions of the two results are exactly the same, the two states differing only by the phase of  $|1\rangle$ . Still, applying the Hadamard transform again on each state will result in very different distributions:

$$H \left[ \frac{|0\rangle + |1\rangle}{\sqrt{2}} \right] = \frac{|0\rangle + |1\rangle}{2} \langle 0|0\rangle + \frac{|0\rangle - |1\rangle}{2} \langle 1|0\rangle + \frac{|0\rangle + |1\rangle}{2} \langle 0|1\rangle + \frac{|0\rangle - |1\rangle}{2} \langle 1|1\rangle$$

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

$$= |0\rangle$$

$$H \left[ \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right] = \frac{|0\rangle + |1\rangle}{2} \langle 0|0\rangle + \frac{|0\rangle - |1\rangle}{2} \langle 1|0\rangle - \frac{|0\rangle + |1\rangle}{2} \langle 0|1\rangle - \frac{|0\rangle - |1\rangle}{2} \langle 1|1\rangle$$

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

$$= |1\rangle$$

When the same series of operations is performed on seemingly only slightly differing systems, the two systems can, and do, produce critically different results. Amplitudes are thus an important distinguishing trait of quantum computers.

The three Pauli gates, named after yet another Nobel laureate Wolfgang Pauli, are also important single-qubit gates for quantum computation. The Pauli-X, -Y, and -Z gates correspond to rotations by  $\pi$  radians about the  $x$ ,  $y$ , and  $z$  axes respectively on the Bloch sphere. The Pauli-X gate swaps the amplitudes of  $|0\rangle$  and  $|1\rangle$ :

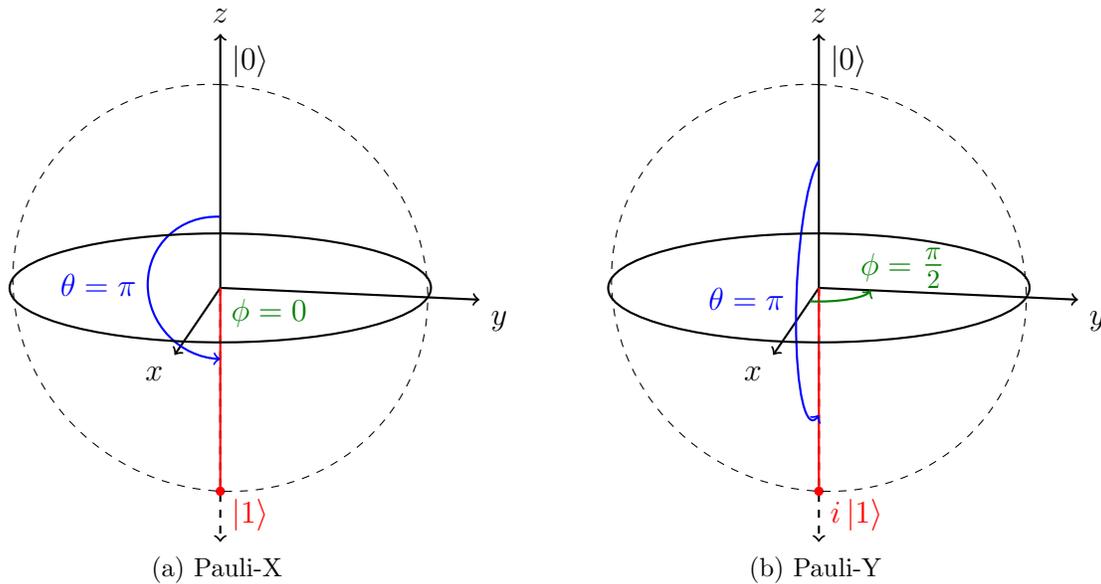


Figure 3: The results of Pauli-X and Pauli-Y gates applied to  $|0\rangle$  differ only by phase

$$\text{---} \boxed{X} \text{---} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = |1\rangle \langle 0| + |0\rangle \langle 1|$$

The Pauli-Y gate swaps the amplitudes of  $|0\rangle$  and  $|1\rangle$ , multiplies each amplitude by  $i$ , and negates the amplitude of  $|1\rangle$ :

$$\text{---} \boxed{Y} \text{---} = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} = i |1\rangle \langle 0| - i |0\rangle \langle 1|$$

And the Pauli-Z gate negates the amplitude of  $|1\rangle$ , leaving the amplitude of  $|0\rangle$  the same:

$$\text{---} \boxed{Z} \text{---} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = |1\rangle \langle 0| - |0\rangle \langle 1|$$



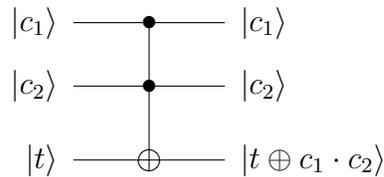
With the matrix representation:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & x_{00} & x_{10} \\ 0 & 0 & x_{01} & x_{11} \end{bmatrix}$$

And Dirac equivalent:

$$|00\rangle\langle 00| + |01\rangle\langle 01| + x_{00}|10\rangle\langle 10| + x_{01}|10\rangle\langle 11| + x_{10}|11\rangle\langle 10| + x_{11}|11\rangle\langle 11|$$

In fact, controlled operations are possible with any number  $n$  control qubits and any unitary operator on  $k$  qubits. The Toffoli gate is probably the most well-known of these gates. Also known as the controlled-controlled-NOT gate, the Toffoli gate acts on three qubits: as the alternative name would suggest, two control qubits and one target. If both control qubits are set, then the amplitudes of the target qubit are flipped:



The Toffoli gate, originally devised as a universal, reversible classical logic gate by Tommaso Toffoli [19], is especially interesting because depending on the input, the gate can perform logical AND, XOR, NOT and FANOUT operations, making it universal for classical computing. Since quantum computing is reversible<sup>8</sup>, the reversible Toffoli gate is a valid transformation not only on classical circuits, but also of quantum circuits, implying that quantum computation is at least as powerful as classical computation. The Dirac representation of the 3-qubit Toffoli gate is a bit unwieldy, but the matrix representation is fairly straightforward:

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

Understanding qubits, quantum registers, and the quantum gates that act on them, it

<sup>8</sup>All evolution in a quantum system can be described by unitary matrices. All unitary transformations are invertible, and thus all quantum computation is reversible.

is possible to begin understanding quantum algorithms. One last mathematical formality remains: how to describe the computational complexity of quantum systems. After all, it's hard to assert that a quantum algorithm can solve a problem faster than a classical algorithm with no way of quantifying and comparing the computational complexity.

## 2.6 Computational complexity

In order to begin to understand the possible power of quantum computing, it helps to look at the computational power of quantum computers in relation to their classical counterparts. Remember that problems in  $P$  are decision problems that can be solved in polynomial time by a deterministic Turing machine. The equivalent for space efficiency, the set of decision problems that can be solved by a Turing machine<sup>9</sup> using polynomial space, is referred to as  $PSPACE$ .  $NP$  problems, on the other hand, are those that require a nondeterministic Turing machine in order to be solved efficiently, and so exponential time on a deterministic machine. The class of  $NP$ -complete problems, abbreviated  $NPC$ , consists of the hardest problems in  $NP$ . Every problem in  $NP$  can be reduced to a problem in  $NPC$ . If one  $NPC$  problem was found to be in  $P$ , then all of the problems in  $NP$  would also be in  $P$ , proving the long-open problem of whether  $P = NP$ . Most theoretical computer scientists believe that  $P \neq NP$ , but nobody has been able to successfully prove the conjecture either way.

There is another important complexity class called  $BPP$ : Bounded-error Probabilistic Polynomial time.  $BPP$  describes decision problems that can be solved in polynomial time by a *probabilistic* Turing machine, with a chance that the solution could be wrong. Probabilistic Turing machines are those with “the ability to flip coins in order to make random decisions” [20], machines with direct access to some source of truly random input. The downside to using randomness in an algorithm is that it often means that there will be a chance that the algorithm will produce an incorrect result. In  $BPP$ , the error of the solution is bounded in that the probability that the answer is correct must be at least two-thirds. More formally, if the answer to the decision problem is “yes,” then at least two-thirds of the possible computational paths must accept, and if the solution is “no,” then at most one-third can accept<sup>10</sup>. Although there are currently problems solvable in  $BPP$  that are not in  $P$ , the number of such problems has been decreasing since the introduction of  $BPP$  in the 1970's. While it is not yet been proven whether  $P \subset BPP$ , it is conjectured that  $P = BPP$  [21, 22].

Quantum computation introduces a number of new complexity classes to the polynomial hierarchy. Probably the most studied complexity class is Bounded-error Quantum Polynomial time, or  $BQP$ .  $BQP$  is the quantum extension of  $BPP$ : the class of decision

---

<sup>9</sup>Savitch's theorem states that a deterministic Turing machine requires only quadratically more space to solve the same problems as a nondeterministic Turing machine;  $PSPACE = NPSPACE$ .

<sup>10</sup>In terms of accepting paths,  $P$  and  $NP$  require that if the answer is “yes,” there must be at least one accepting path, and no accepting path otherwise.

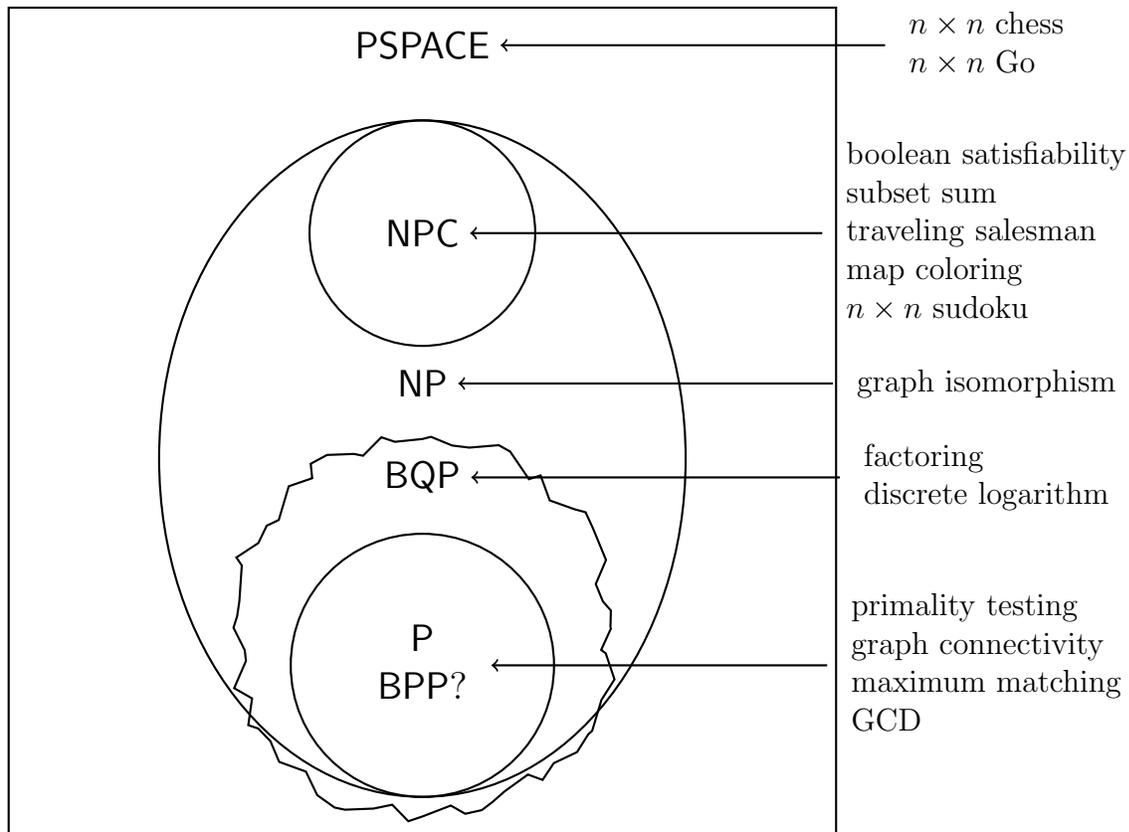


Figure 4: A conjectured polynomial hierarchy [23]

problems solvable in polynomial time by an innately probabilistic quantum Turing machine, with the same error constraint as defined for BPP. Unlike BPP, it is suspected that  $P \subset BQP$  [24], which would mean that quantum computers are capable of solving some problems in polynomial time that cannot be solved efficiently by a classical Turing machine! The conjectured relationships between all of these complexity classes, as well as some of the biggest problems known to be in each class, are depicted in Figure 4.

### 3 Grover's Algorithm

#### 3.1 Quantum search

As discussed earlier, Grover's algorithm performs a search over an unordered set of  $N = 2^n$  items to find the unique element that satisfies some condition. While the best classical

algorithm for a search over unordered data requires  $O(N)$  time<sup>11</sup>, Grover’s algorithm performs the search on a quantum computer in only  $O(\sqrt{N})$  operations, a quadratic speedup. As Grover himself notes, if the algorithm were to run in a finite power of  $O(\lg N)$  steps, then it would provide an algorithm in BQP for problems in NPC [5]. However, Grover’s algorithm does *not* provide such a runtime, and is an asymptotically optimal solution, so no definitive statement can be made about the relationship between the complexity classes BQP and NP based on the performance of Grover’s algorithm.

Grover’s search algorithm is a good introduction to quantum algorithms because it demonstrates how the qualities of quantum systems can be used to improve upon the lower runtime bounds of classical algorithms. In order to achieve such a speedup, Grover relies on the quantum superposition of states. Like many quantum algorithms, Grover’s begins by putting the machine into an equal superposition of all possible  $2^n$  states of the  $n$ -qubit register. Remember that means there is an equal amplitude of  $1/\sqrt{2^n}$  associated with every possible configuration of qubits in the system, and an equal probability of  $1/2^n$  that the system will be in any of the  $2^n$  states. All of these possible states correspond to all the possible entries in Grover’s database, and so starting with equal amplitudes assigned to each element in the search space, every element can be considered at once in a quantum superposition, and amplitudes can be manipulated from there to produce the correct entry in the database with a probability of “at least”  $1/2$  [5].

Along with the superposition of states, Grover’s algorithm, and more generally the family of quantum algorithms that use what is known as *amplitude amplification*, exploit the qualities of quantum amplitudes that differentiate those amplitudes from simple probabilities. The key to these algorithms is the selective shifting of the phase of one state of a quantum system, one that satisfies some condition, at each iteration. Performing a phase shift of  $\pi$  is equivalent to multiplying the amplitude of that state by  $-1$ : the amplitude for that state changes, but the probability of being in that state remains the same (since the probability disregards the sign of the amplitude). However, subsequent transformations performed on the system take advantage of that difference in amplitude to single out that state of differing phase and to ultimately increase the probability of the system being in that state. Such sequences of operations would not be possible if the amplitudes did not hold that extra information regarding the phase of the state in addition to the probability. These amplitude amplification algorithms are unique to quantum computing because of this quality of amplitudes that has no analog in classical probabilities.

---

<sup>11</sup>Refresher: assuming uniform probability distribution that any of the  $n$  items is the correct item in a given search, an average of  $\frac{n+1}{2} = O(n)$  probes will be required in order to find the correct item on a classical computer. With a uniform distribution, the probability that any given item is the correct one is  $\frac{1}{n}$ . In order to calculate the average number of probes necessary to find the correct item in the array, we find the weighted average of the number of probes required to find the correct item in each of its possible positions, from the first position probed to the  $n^{\text{th}}$ :  $\sum_{i=1}^n \frac{1}{n} i = \frac{1}{n} \sum_{i=1}^n i = \frac{1}{n} \left[ \frac{n(n+1)}{2} \right] = \frac{n+1}{2}$ .

### 3.2 Grover's algorithm: How it works

Grover's algorithm begins with a quantum register of  $n$  qubits, where  $n$  is the number of qubits necessary to represent the search space of size  $2^n = N$ , all initialized to  $|0\rangle$ :

$$|0\rangle^{\otimes n} = |0\rangle \quad (1)$$

The first step is to put the system into an equal superposition of states, achieved by applying the Hadamard transform  $H^{\otimes n}$ , which requires  $\Theta(\lg N) = \Theta(\lg 2^n) = \Theta(n)$  operations,  $n$  applications of the elementary Hadamard gate:

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

The next series of transformations is often referred to as the *Grover iteration*, and performs the amplitude amplification mentioned earlier, the bulk of the algorithm. The Grover iteration will be repeated  $\frac{\pi}{4}\sqrt{2^n}$  times. According to Grover [5], in order to achieve optimal probability that the state that we ultimately observe is the correct one, we want the overall rotation of the phase to be  $\frac{\pi}{4}$  radians, which will occur on average after  $\frac{\pi}{4}\sqrt{2^n}$  iterations. The first step in the Grover iteration is a call to a *quantum oracle*,  $\mathcal{O}$ , that will modify the system depending on whether it is in the configuration we are searching for. An oracle is basically a black-box function, and this quantum oracle is a quantum black-box, meaning it can observe and modify the system without collapsing it to a classical state, that will recognize if the system is in the correct state. If the system is indeed in the correct state, then the oracle will rotate the phase by  $\pi$  radians, otherwise it will do nothing, effectively marking the correct state for further modification by subsequent operations. Remember that such a phase shift leaves the probability of the system being correct state the same, although the amplitude is negated. Quantum oracle implementations will often use an extra scratch qubit, but in this implementation the extra qubit is unnecessary, so the oracle's effect on  $|x\rangle$  may be written simply:

$$|x\rangle \xrightarrow{\mathcal{O}} (-1)^{f(x)} |x\rangle \quad (3)$$

Where  $f(x) = 1$  if  $x$  is the correct state, and  $f(x) = 0$  otherwise. The exact implementation of  $f(x)$  is dependent on the particular search problem.

Grover refers to the next part of the iteration as the *diffusion transform*, which performs *inversion about the average*, transforming the amplitude of each state so that it is as far above the average as it was below the average prior to the transformation, and vice versa. This diffusion transform consists of another application of the Hadamard transform  $H^{\otimes n}$ , followed by a conditional phase shift that shifts every state except  $|0\rangle$  by  $-1$ , followed by yet another Hadamard transform. The conditional phase shift can be

represented by the unitary operator  $2|0\rangle\langle 0| - I$ :

$$[2|0\rangle\langle 0| - I]|0\rangle = 2|0\rangle\langle 0|0\rangle - I|0\rangle = |0\rangle \tag{4a}$$

$$[2|0\rangle\langle 0| - I]|x\rangle = 2|0\rangle\langle 0|x\rangle - I|x\rangle = -|x\rangle \tag{4b}$$

Giving the entire diffusion transform, using the notation  $|\psi\rangle$  from equation 2:

$$H^{\otimes n} [2|0\rangle\langle 0| - I] H^{\otimes n} = 2H^{\otimes n}|0\rangle\langle 0|H^{\otimes n} - I = 2|\psi\rangle\langle\psi| - I \tag{5}$$

And the entire Grover iteration:

$$[2|\psi\rangle\langle\psi| - I]\mathcal{O} \tag{6}$$

In considering the runtime of the Grover iteration, the exact runtime of the oracle depends on the specific problem and implementation, so a call to  $\mathcal{O}$  is viewed as one elementary operation. The total runtime, then, of a single Grover iteration is  $\Theta(2n)$ , from the two Hadamard transforms, plus the cost of applying  $O(n)$  gates to perform the conditional phase shift [12], is  $O(n)$ . It follows that the runtime of Grover's entire algorithm, performing  $O(\sqrt{N}) = O(\sqrt{2^n}) = O(2^{\frac{n}{2}})$  iterations each with a runtime of  $O(n)$ , is  $O(2^{\frac{n}{2}})$ .

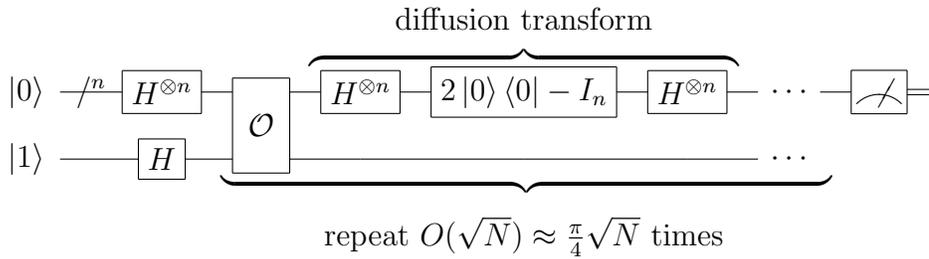


Figure 5: Circuit diagram for Grover's algorithm, with a scratch qubit for the oracle [25].

Once the Grover iteration has been performed an adequate number of times, a classical measurement is performed to determine the result, which will be correct with probability  $O(1)$  completing the execution of the algorithm.

Grover's algorithm is summarized nicely in [12] as follows:

**Input:**

- A quantum oracle  $\mathcal{O}$  which performs the operation  $\mathcal{O}|x\rangle = (-1)^{f(x)}|x\rangle$ , where  $f(x) = 0$  for all  $0 \leq x < 2^n$  except  $x_0$ , for which  $f(x_0) = 1$ .
- $n$  qubits initialized to the state  $|0\rangle$

**Output:**  $x_0$

**Runtime:**  $O(\sqrt{2^n})$  operations, with  $O(1)$  probability of success.

**Procedure:**

1.  $|0\rangle^{\otimes n}$  initial state
2.  $H^{\otimes n} |0\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle = |\psi\rangle$  apply the Hadamard transform to all qubits
3.  $[(2|\psi\rangle\langle\psi| - I)\mathcal{O}]^R |\psi\rangle \approx |x_0\rangle$  apply the Grover iteration  $R \approx \frac{\pi}{4}\sqrt{2^n}$  times
4.  $x_0$  measure the register

### 3.3 Grover's algorithm: Worked example

Consider a system consisting of  $N = 8 = 2^3$  states, and the state we are searching for,  $x_0$ , is represented by the bit string 011:

To describe this system,  $n = 3$  qubits are required, represented as:

$$|x\rangle = \alpha_0 |000\rangle + \alpha_1 |001\rangle + \alpha_2 |010\rangle + \alpha_3 |011\rangle + \alpha_4 |100\rangle + \alpha_5 |101\rangle + \alpha_6 |110\rangle + \alpha_7 |111\rangle \quad (7)$$

where  $\alpha_i$  is the amplitude of the state  $|i\rangle$ . Grover's algorithm begins with a system initialized to 0:

$$1 |000\rangle \quad (8)$$

and then apply the Hadamard transformation to obtain equal amplitudes associated with each state of  $1/\sqrt{N} = 1/\sqrt{8} = 1/2\sqrt{2}$ , and thus also equal probability of being in any of the 8 possible states:

$$H^3 |000\rangle = \frac{1}{2\sqrt{2}} |000\rangle + \frac{1}{2\sqrt{2}} |001\rangle + \dots + \frac{1}{2\sqrt{2}} |111\rangle = \frac{1}{2\sqrt{2}} \sum_{x=0}^7 |x\rangle = |\psi\rangle \quad (9)$$

A geometric interpretation of the amplitudes of states is a useful way of better visualizing how this algorithm works. Since the amplitudes remain real throughout the execution of Grover's algorithm, they may be visualized as lines perpendicular to an axis whose lengths are proportional to the amplitude they represent. The equal superposition of states resulting from the first Hadamard transform appears as follows:

$$\begin{array}{c}
 \text{-----} \alpha_\psi = \frac{1}{2\sqrt{2}} \\
 | \quad | \quad | \quad | \quad | \quad | \quad | \\
 \text{-----} \\
 |000\rangle |001\rangle |010\rangle |011\rangle |100\rangle |101\rangle |110\rangle |111\rangle
 \end{array}$$

It is optimal to now perform 2 Grover iterations in order to obtain the solution:  
 $\frac{\pi}{4}\sqrt{8} = \frac{2\pi}{4}\sqrt{2} = \frac{\pi}{2}\sqrt{2} \approx 2.22$ , which rounds to 2 iterations.

In each iteration, the first step is to call the quantum oracle  $\mathcal{O}$ , then perform inversion about the average, or the diffusion transform. The oracle query will negate the amplitude of the state  $|x_0\rangle$ , in this case  $|011\rangle$ , giving the configuration:

$$|x\rangle = \frac{1}{2\sqrt{2}} |000\rangle + \frac{1}{2\sqrt{2}} |001\rangle + \frac{1}{2\sqrt{2}} |010\rangle - \frac{1}{2\sqrt{2}} |011\rangle + \dots + \frac{1}{2\sqrt{2}} |111\rangle \quad (10)$$

And the geometric representation:

$$\begin{array}{c}
 \text{-----} \alpha_\psi = \frac{1}{2\sqrt{2}} \\
 | \quad | \quad | \quad | \quad | \quad | \quad | \\
 \text{-----} \\
 |000\rangle |001\rangle |010\rangle |011\rangle |100\rangle |101\rangle |110\rangle |111\rangle \\
 \text{-----} \alpha_{|011\rangle} = \frac{-1}{2\sqrt{2}}
 \end{array}$$

Now, perform the diffusion transform  $2|\psi\rangle\langle\psi| - I$ , which will increase the amplitudes by their difference from the average, decreasing if the difference is negative:

$$[2|\psi\rangle\langle\psi| - I] |x\rangle \quad (11a)$$

$$= [2|\psi\rangle\langle\psi| - I] \left[ |\psi\rangle - \frac{2}{2\sqrt{2}} |011\rangle \right] \quad (11b)$$

$$= 2|\psi\rangle\langle\psi|\psi\rangle - |\psi\rangle - \frac{2}{\sqrt{2}} |\psi\rangle\langle\psi|011\rangle + \frac{1}{\sqrt{2}} |011\rangle \quad (11c)$$

$$\quad (11d)$$

Note that  $\langle\psi|\psi\rangle = 8 \frac{1}{2\sqrt{2}} \left[ \frac{1}{2\sqrt{2}} \right] = 1$ . Additionally, since  $|011\rangle$  is one of the basis vectors, we can use the identity  $\langle\psi|011\rangle = \langle 011|\psi\rangle = \frac{1}{2\sqrt{2}}$ :

$$= 2|\psi\rangle - |\psi\rangle - \frac{2}{\sqrt{2}} \left( \frac{1}{2\sqrt{2}} \right) |\psi\rangle + \frac{1}{\sqrt{2}} |011\rangle \quad (11e)$$

$$= |\psi\rangle - \frac{1}{2} |\psi\rangle + \frac{1}{\sqrt{2}} |011\rangle \quad (11f)$$

$$= \frac{1}{2} |\psi\rangle + \frac{1}{\sqrt{2}} |011\rangle \quad (11g)$$

Substituting from Equation 9 gives:

$$= \frac{1}{2} \left[ \frac{1}{2\sqrt{2}} \sum_{x=0}^7 |x\rangle \right] + \frac{1}{\sqrt{2}} |011\rangle \tag{11h}$$

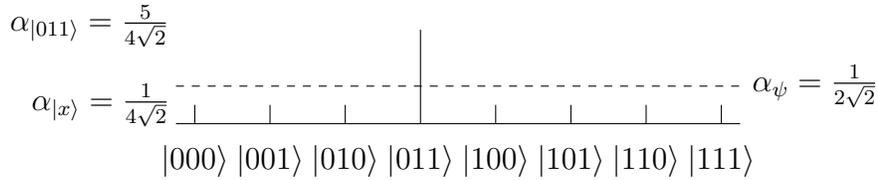
$$= \frac{1}{4\sqrt{2}} \sum_{\substack{x=0 \\ x \neq 3}}^7 |x\rangle + \frac{1}{4\sqrt{2}} |011\rangle + \frac{1}{\sqrt{2}} |011\rangle \tag{11i}$$

$$= \frac{1}{4\sqrt{2}} \sum_{\substack{x=0 \\ x \neq 3}}^7 |x\rangle + \frac{5}{4\sqrt{2}} |011\rangle \tag{11j}$$

In the notation used earlier:

$$|x\rangle = \frac{1}{4\sqrt{2}} |000\rangle + \frac{1}{4\sqrt{2}} |001\rangle + \frac{1}{4\sqrt{2}} |010\rangle + \frac{5}{4\sqrt{2}} |011\rangle + \dots + \frac{1}{4\sqrt{2}} |111\rangle \tag{11k}$$

Which appears geometrically as:



This completes the first iteration. We apply the same two transformations in the second iteration, giving:

$$|x\rangle = \frac{1}{4\sqrt{2}} |000\rangle + \frac{1}{4\sqrt{2}} |001\rangle + \frac{1}{4\sqrt{2}} |010\rangle - \frac{5}{4\sqrt{2}} |011\rangle + \dots + \frac{1}{4\sqrt{2}} |111\rangle \tag{12a}$$

$$= \frac{1}{4\sqrt{2}} \sum_{\substack{x=0 \\ x \neq 3}}^7 |x\rangle - \frac{5}{4\sqrt{2}} |011\rangle \tag{12b}$$

$$= \frac{1}{4\sqrt{2}} \sum_{x=0}^7 |x\rangle - \frac{6}{4\sqrt{2}} |011\rangle \tag{12c}$$

$$= \frac{1}{2} |\psi\rangle - \frac{3}{2\sqrt{2}} |011\rangle \tag{12d}$$

after the oracle query, and after applying the diffusion transform:

$$[2|\psi\rangle\langle\psi| - I] \left[ \frac{1}{2}|\psi\rangle - \frac{3}{2\sqrt{2}}|011\rangle \right] \tag{13a}$$

$$= 2 \left( \frac{1}{2} \right) |\psi\rangle\langle\psi|\psi\rangle - \frac{1}{2}|\psi\rangle - 2 \left( \frac{3}{2\sqrt{2}} \right) |\psi\rangle\langle\psi|011\rangle + \frac{3}{2\sqrt{2}}|011\rangle \tag{13b}$$

$$= |\psi\rangle - \frac{1}{2}|\psi\rangle - \frac{3}{\sqrt{2}} \left( \frac{1}{2\sqrt{2}} \right) |\psi\rangle + \frac{3}{2\sqrt{2}}|011\rangle \tag{13c}$$

$$= -\frac{1}{4}|\psi\rangle + \frac{3}{2\sqrt{2}}|011\rangle \tag{13d}$$

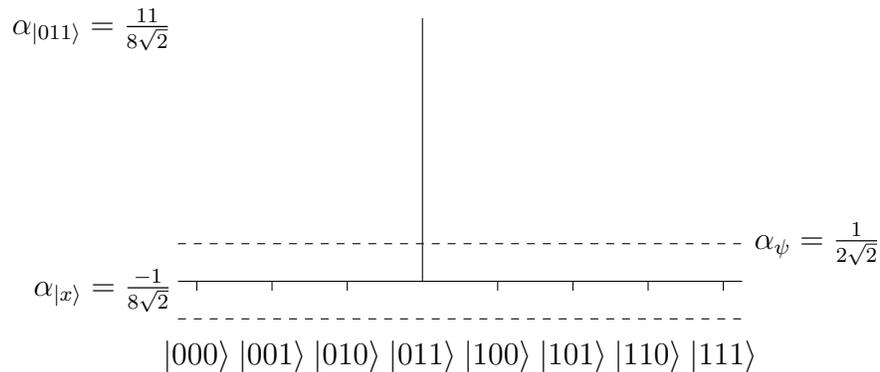
$$= -\frac{1}{4} \left[ \frac{1}{2\sqrt{2}} \sum_{\substack{x=0 \\ x \neq 3}}^7 |x\rangle + \frac{1}{2\sqrt{2}}|011\rangle \right] + \frac{3}{2\sqrt{2}}|011\rangle \tag{13e}$$

$$= -\frac{1}{8\sqrt{2}} \sum_{\substack{x=0 \\ x \neq 3}}^7 |x\rangle + \frac{11}{8\sqrt{2}}|011\rangle \tag{13f}$$

Or in the expanded notation:

$$|x\rangle = -\frac{1}{8\sqrt{2}}|000\rangle - \frac{1}{8\sqrt{2}}|001\rangle - \frac{1}{8\sqrt{2}}|010\rangle + \frac{11}{8\sqrt{2}}|011\rangle - \dots - \frac{1}{8\sqrt{2}}|111\rangle \tag{13g}$$

Geometrically, the success of the algorithm is clear:



Now when the system is observed, the probability that the state representative of the correct solution,  $|011\rangle$ , will be measured is  $|\frac{11}{8\sqrt{2}}|^2 = 121/128 \approx 94.5\%$ . The probability of finding an incorrect state is  $|\frac{-\sqrt{7}}{8\sqrt{2}}|^2 = 7/128 \approx 5.5\%$ ; Grover's algorithm is more than 17 times more likely to give the correct answer than an incorrect one with an input size of  $N = 8$ , and the error only decreases as the input size increases. Although Grover's algorithm is probabilistic, the error truly becomes negligible as  $N$  grows large.

## 4 Simon's Algorithm

### 4.1 Black-box period finding

Simon's problem is, given a function

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

that is known to be invariant under some  $n$ -bit XOR mask  $a$ , determine  $a$ . In other words, determine  $a$  given:

$$f(x) = f(y) \iff x \oplus y \in \{0^n, a\}$$

This problem was one of the first problems for which a quantum algorithm was found to provide exponential speedup over any classical algorithm: the best classical algorithms, including probabilistic ones, require an exponential  $\Omega(2^{n/2})$  queries to the black-box function in order to determine  $a$ . Simon's quantum algorithm solves this problem in polynomial time, performing an optimal  $O(n)$  queries [8].

This algorithm is a good example of the type that might soon be used in conjunction with classical computation. Even though quantum computers are capable of computing all the same problems as a classical computer, it will not necessarily make sense to use quantum computers to do so while quantum technology is still young and pricey. Simon's algorithm requires some processing at the end, solving a system of linear equations, in order to determine the final solution. The system could be solved by a quantum computer, and even exponentially more quickly than by a classical machine [4]. But, there are polynomial-time classical algorithms for solving linear systems of equations that might be preferred if quantum computing time is expensive, as will probably be true for any quantum computing resources that become available in the near future. While a quantum computer could be very beneficial for solving problems that would take exponential time on a classical computer, problems that can be solved efficiently on a classical computer might be better left to the classical machine, whose resources are cheap and deterministic.

Both Simon's algorithm and Shor's well-known prime factorization algorithm solve a similar problem: given a function  $f$ , find the period  $a$  of that function. While Simon's problem uses exclusive-OR to define the period, Shor's uses binary addition as the constraint on  $f$ . These problems are more restricted cases of what is known as the hidden subgroup problem, which correspond to a number of important problems in computer science. In addition to the specific uses of the Abelian hidden subgroup problem that Shor and Simon's algorithms address, any formulation of the Abelian hidden subgroup problem can be solved by a quantum computer requiring a number of operations logarithmic in the size of the group [12].

The more general hidden subgroup problem is harder to solve. Analogous to the graph

isomorphism problem, which is determining whether two finite graphs are isomorphic, and some shortest vector problems in lattices, finding the shortest nonzero vector in a vector space, both of which have a number of applications to cryptography and other areas in computer science. Currently no polynomial-time algorithms have been devised to solve this problem [9]. If a new quantum algorithm were found to solve the hidden subgroup problem, or even just another special case of this problem, it would be a breakthrough in quantum computing similar to Shor's discovery.

### 4.2 Simon's algorithm: How it works

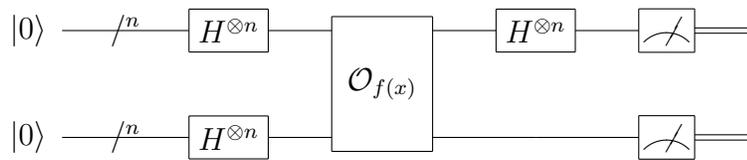


Figure 6: Circuit diagram for one iteration of Simon's algorithm [26]

Given a function acting on  $n$ -bit strings, Simon's algorithm begins by initializing two  $n$ -bit registers to 0:

$$|0\rangle^{\otimes n} |0\rangle^{\otimes n} \tag{14}$$

Then applying the Hadamard transform to the first register to attain an equal superposition of states:

$$H^{\otimes n} |0\rangle |0\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle |0\rangle \tag{15}$$

Next, the given oracle function  $f(x)$  is queried on both the registers. The oracle is implemented as a unitary operation that performs the transformation  $\mathcal{O}_{f(x)} |x\rangle |y\rangle = |x\rangle |f(x) \oplus y\rangle$ . When the oracle is called on the registers in the configuration described above, the result will be no change to the first register, and  $f(x)$  stored in the second register, since  $f(x) \oplus 0 = f(x)$ :

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle |f(x)\rangle \tag{16}$$

Now the second register is measured. There are two possible cases to consider in determining the impact of that measurement on the first register: either the XOR mask  $a = 0^n$  or  $a = \{0,1\}^n$ . If  $a = 0^n$ , then  $f$  is injective: each value of  $x$  corresponds to a

unique value  $f(x)$ . This means that the first register remains in an equal superposition; Regardless of the measured value of  $f(x)$ ,  $x$  could be any bit string in  $\{0, 1\}^n$  with equal probability. On the other hand, if  $a = \{0, 1\}^n$ , measuring the second register determines a concrete value of  $f(x)$ , call it  $f(z)$ , which limits the possible values of the first register. By the definition of the function  $f(x)$ , there are exactly two possible values of  $x$  such that  $f(x) = f(z)$ :  $z$  and  $z \oplus a$ . The state of the first register after measuring the second is thus reduced to an equal superposition of those two values:

$$\frac{1}{\sqrt{2}} |z\rangle + \frac{1}{\sqrt{2}} |z \oplus a\rangle \quad (17)$$

Since there will be no more operations on the second register, further calculations will focus only on the first register.

The next step is to isolate the information about  $a$  that is now stored in the first register. This can be done by applying the Hadamard transform again. Remember that the Hadamard transform may be defined using the bitwise dot product  $x \cdot y$  as:

$$H^{\otimes n} |x\rangle = \frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle \quad (18)$$

Using this notation, the result of applying a second Hadamard operation is:

$$H^{\otimes n} \left[ \frac{1}{\sqrt{2}} |z\rangle + \frac{1}{\sqrt{2}} |z \oplus a\rangle \right] \quad (19a)$$

$$= \frac{1}{\sqrt{2}} H^{\otimes n} |z\rangle + \frac{1}{\sqrt{2}} H^{\otimes n} |z \oplus a\rangle \quad (19b)$$

$$= \frac{1}{\sqrt{2}} \left[ \frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{z \cdot y} |y\rangle \right] + \frac{1}{\sqrt{2}} \left[ \frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{(z \oplus a) \cdot y} |y\rangle \right] \quad (19c)$$

$$= \frac{1}{\sqrt{2^{n+1}}} \sum_{y \in \{0,1\}^n} [(-1)^{z \cdot y} + (-1)^{(z \oplus a) \cdot y}] |y\rangle \quad (19d)$$

$$= \frac{1}{\sqrt{2^{n+1}}} \sum_{y \in \{0,1\}^n} [(-1)^{z \cdot y} + (-1)^{(z \cdot y) \oplus (a \cdot y)}] |y\rangle \quad (19e)$$

$$= \frac{1}{\sqrt{2^{n+1}}} \sum_{y \in \{0,1\}^n} (-1)^{z \cdot y} [1 + (-1)^{a \cdot y}] |y\rangle \quad (19f)$$

Now the value of the first register is measured. In the degenerate case where  $a = 0^n$  ( $f$  is injective), a string will be produced from  $\{0, 1\}^n$  with uniform distribution.

In the case where  $x \oplus y \neq 0^n$ , notice that either  $a \cdot y = 0$  or  $a \cdot y = 1$ . If  $a \cdot y = 1$ , then

Equation 19f becomes:

$$\frac{1}{\sqrt{2^{n+1}}} \sum_{y \in \{0,1\}^n} (-1)^{z \cdot y} [1 + (-1)^1] |y\rangle = \frac{1}{\sqrt{2^{n+1}}} \sum_{y \in \{0,1\}^n} (-1)^{z \cdot y} [0] |y\rangle \quad (20a)$$

$$= 0 |y\rangle \quad (20b)$$

The amplitude, and thus probability, that a value of  $y$  such that  $a \cdot y = 1$  is equal to 0, and so such a  $y$  will never be measured. Knowing that it will always be true that  $a \cdot y = 0$ , Equation 19f can be simplified:

$$\frac{1}{\sqrt{2^{n+1}}} \sum_{y \in \{0,1\}^n} (-1)^{z \cdot y} [1 + (-1)^0] |y\rangle = \frac{2}{\sqrt{2^{n+1}}} \sum_{y \in \{0,1\}^n} (-1)^{z \cdot y} |y\rangle \quad (21a)$$

$$= \frac{1}{\sqrt{2^{n-1}}} \sum_{y \in \{0,1\}^n} (-1)^{z \cdot y} |y\rangle \quad (21b)$$

When  $a \neq 0^n$ , the result of measuring the first register after performing Simon's algorithm will always produce a string  $y \in \{0,1\}^n : a \cdot y = 0$ . From Equation 21a, the amplitude associated with each value  $y$  is equal to  $\pm\sqrt{2^{1-n}}$ , giving the probability:

$$\left| \frac{1}{\sqrt{2^{n-1}}} \right|^2 = \left| \frac{-1}{\sqrt{2^{n-1}}} \right|^2 = \frac{1}{2^{n-1}} \quad (22)$$

of observing any of the strings  $y$  such that  $a \cdot y = 0$ , a uniform distribution over the  $2^{n-1}$  strings that satisfy  $a \cdot y = 0$ .

If Simon's algorithm is executed  $n - 1$  times,  $n - 1$  strings  $y_1, y_2, \dots, y_{n-1} \in \{0,1\}^n$  can be observed, which form a system of  $n - 1$  linear equations in  $n$  unknowns of the form:

$$\begin{aligned} y_1 \cdot a &= y_{11}a_1 + y_{12}a_2 + \dots + y_{1n}a_n = 0 \\ y_2 \cdot a &= y_{21}a_1 + y_{22}a_2 + \dots + y_{2n}a_n = 0 \\ &\vdots \\ y_{n-1} \cdot a &= y_{(n-1)1}a_1 + y_{(n-1)2}a_2 + \dots + y_{(n-1)n}a_n = 0 \end{aligned}$$

To find  $a$  from here is just a matter of solving for the  $n$  unknowns, each a bit in  $a$ , in order to determine  $a$  as a whole. Of course, this requires a system of  $n - 1$  linearly independent equations.

The probability of observing the first string  $y_0$  is  $2^{1-n}$ . After another iteration of Simon's algorithm, the probability of observing another distinct bit string would be  $1 - 2^{1-n}$ . The probability of observing  $n - 1$  distinct values of  $y$  in a row, and so a lower

bound on the probability of obtaining  $n - 1$  linearly independent equations, is:

$$\prod_{n=1}^{\infty} \left[ 1 - \frac{1}{2^n} \right] \approx .2887881 > \frac{1}{4} \tag{23}$$

So a linearly independent system of  $n - 1$  equations, and from there the value of  $a$ , can be obtained by repeating Simon's algorithm no more than  $4n$  times. Simon's algorithm requires only  $O(n)$  queries to  $f$  in order to determine  $a$ , while classical algorithms require exponential time.

### 4.3 Simon's Algorithm: Worked example

Now a worked example with  $n = 3$ ,  $a = 110$ , and  $f(x)$  defined by the following table:

$x$	$f(x)$
000	101
001	010
010	000
011	110
100	000
101	110
110	101
111	010

First, two 3-bit registers are initialized to  $|0\rangle^{\otimes 3}$ :

$$|000\rangle |000\rangle \tag{24}$$

Then the Hadamard transform is applied to the first register to obtain an equal superposition of states:

$$H^{\otimes 3} |000\rangle |000\rangle = \frac{1}{2\sqrt{2}} \sum_{x \in \{0,1\}^3} |x\rangle |000\rangle \tag{25}$$

Next, the oracle  $f(x)$  is queried, giving:

$$\frac{1}{2\sqrt{2}} \sum_{x \in \{0,1\}^3} |x\rangle |f(x)\rangle \tag{26}$$

And then the second register is measured, collapsing the first register to:

$$\frac{1}{\sqrt{2}} |z\rangle + \frac{1}{\sqrt{2}} |z \oplus 110\rangle \tag{27}$$

Then a second Hadamard transform is applied to the first register:

$$H^{\otimes 3} \left[ \frac{1}{\sqrt{2}} |z\rangle + \frac{1}{\sqrt{2}} |z \oplus 110\rangle \right] \quad (28)$$

$$= \frac{1}{4} \sum_{y \in \{0,1\}^n} (-1)^{z \cdot y} [1 + (-1)^{110 \cdot y}] |y\rangle \quad (29)$$

From here,  $3 - 1 = 2$  observations can be made such that  $a \cdot y = 0$ . Observations  $y_1 = 001$  and  $y_2 = 000$  lead to the system of equations:

$$a_0 a_1 a_2 \cdot 001 = 0(a_0) + 0(a_1) + 1(a_2) = 0$$

$$a_0 a_1 a_2 \cdot 000 = 0(a_0) + 0(a_1) + 0(a_2) = 0$$

With two possible solutions  $a = 000$  or  $a = 110$ . Thus Simon's algorithm provides the unique nonzero solution  $a = 110$ , which is in fact the XOR mask associated with the function  $f$  defined above.

## 5 Conclusion

Many of the more interesting quantum algorithms, such as quantum simulated annealing or quantum Bayesian networks, require a much more thorough understanding of the underlying math. Still, with the new quantum paradigm looming in the distance, it no longer makes sense for quantum computation to be ignored in the undergraduate computer science curriculum. Hopefully, the study of quantum algorithms will soon be commonplace. Until then, this tutorial at least demonstrates that simple quantum algorithms are not beyond the understanding of the average undergraduate computer science student, providing a gentle introduction to the basics of quantum computation to the undergraduate population.

## References

- [1] R. P. Feynman, “Simulating Physics with Computers,” *International Journal of Theoretical Physics*, vol. 21, no. 6/7, pp. 467–488, 1982.
- [2] P. W. Shor, “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer,” *SIAM Journal on Computing*, no. 5, p. 1484.
- [3] R. L. Rivest, A. Shamir *et al.*, “A method for obtaining digital signatures and public-key cryptosystems,” *Communications of the ACM*, no. 2, pp. 120–126, Feb.
- [4] A. Harrow, A. Hassidim *et al.*, “Quantum Algorithm for Linear Systems of Equations,” *Physical Review Letters*, no. 15, Oct.
- [5] L. K. Grover, “A fast quantum mechanical algorithm for database search,” in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*. New York, New York, USA: ACM Press, pp. 212–219.
- [6] C. Zalka, “Grover’s quantum searching algorithm is optimal,” *Physical Review A*, no. 4, pp. 2746–2751, Oct.
- [7] D. R. Simon, “On the Power of Quantum Computation,” *SIAM Journal on Computing*, no. 5, p. 1474.
- [8] P. Koiran, V. Nesan *et al.*, “A quantum lower bound for the query complexity of Simon’s problem,” *Lecture Notes in Computer Science*, pp. 1287–1298, Jun.
- [9] R. Josza, “Quantum factoring, discrete logarithms, and the hidden subgroup problem,” *IEEE Computing in Science & Engineering*, no. 2, pp. 34–43.
- [10] R. Laboratories. (2010) 3.1.5 How large a key should be used in the RSA cryptosystem? (Accessed 3 January, 2011). [Online]. Available: <http://www.rsa.com/rsalabs/node.asp?id=2218>
- [11] J. Markoff. (2010) Quantum Computing Reaches for True Power. New York Times. (Accessed 3 January, 2011). [Online]. Available: <http://www.nytimes.com/2010/11/09/science/09compute.html>
- [12] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. New York, New York, USA: Cambridge University Press, 2000.
- [13] H. Neven, M. Drew-brook *et al.*, “NIPS 2009 Demonstration : Binary Classification using Hardware Implementation of Quantum Annealing,” pp. 1–17, 2009.
- [14] X.-M. Jin, J.-G. Ren *et al.*, “Experimental free-space quantum teleportation,” *Nature Photonics*, no. 6, pp. 376–381, May.

- 
- [15] S. Aaronson. (2006) Quantum Computing Since Democritus, Lecture 9: Quantum. (Accessed 1 February, 2011). [Online]. Available: <http://www.scottaaronson.com/democritus/lec9.html>
- [16] D. Deutsch, “Quantum theory, the Church-Turing principle and the universal quantum computer,” *Proceedings of the Royal Society of London*, vol. 400, pp. 97–117, 1985.
- [17] —, “Quantum Computational Networks,” *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, no. 1868, pp. 73–90, Sep.
- [18] A. Barenco, C. Bennett *et al.*, “Elementary gates for quantum computation,” *Physical Review A*, no. 5, pp. 3457–3467, Nov.
- [19] T. Toffoli, “Reversible Computation,” Cambridge, MA, p. 36, 1980.
- [20] J. T. Gill, “Computational complexity of probabilistic Turing machines,” in *Proceedings of the sixth annual ACM symposium on Theory of computing - STOC '74*. New York, New York, USA: ACM Press, pp. 91–95.
- [21] C. H. Bennett and J. Gill, “Relative to a Random Oracle  $A$ ,  $P^A \neq NP^A \neq co-NP^A$  with Probability 1,” *SIAM Journal on Computing*, no. 1, p. 96.
- [22] R. Impagliazzo and A. Wigderson, “ $P = BPP$  if  $E$  requires exponential circuits,” in *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing - STOC '97*. New York, New York, USA: ACM Press, pp. 220–229.
- [23] S. Aaronson, “The limits of quantum computers.” *Scientific American*, no. 3, pp. 50–7, Mar.
- [24] —, “BQP and the polynomial hierarchy,” in *Proceedings of the 42nd ACM symposium on Theory of computing - STOC '10*. New York, New York, USA: ACM Press, p. 141.
- [25] J. C. Benoist. Quantum circuit representation of Grover’s algorithm. Wikimedia, Inc. (Accessed 5 January, 2011). [Online]. Available: [http://en.wikipedia.org/wiki/File:Grovers\\_algorithm.svg](http://en.wikipedia.org/wiki/File:Grovers_algorithm.svg)
- [26] —. (2011) Quantum subroutine in Simon’s algorithm. Wikimedia, Inc. (Accessed 5 January, 2011). [Online]. Available: [http://en.wikipedia.org/wiki/File:Simons\\_algorithm.svg](http://en.wikipedia.org/wiki/File:Simons_algorithm.svg)