# Quantum Computing Algorithms: Theory and Practice

Aisling Heanue

Supervisor: Adrian Ottewill

University College Dublin
24-04-2022

The research underlying this thesis has been carried out by Aisling Heanue under the supervision of Adrian Ottewill. This thesis is submitted to University College Dublin in partial fulfillment of the requirements for the degree of BSc in Applied and Computational Mathematics.

## Abstract

We look at Shor's algorithm and Grover's search algorithm and their derivations. We reproduce the results of Shor's algorithm using the classical discrete Fourier transform for the periodic function $f(x) = a^x \bmod N$. We then construct the circuits for Shor's algorithm and Grover's search algorithm, and verify that their results are as expected by simulating them with Aer. We create a circuit for Shor's algorithm capable of factorising any odd composite number up to $2^n$ with $5n+2$ qubits, and demonstrate how it could be used to factorise 33 with 32 qubits. We run demonstration circuits for Shor's algorithm and Grover's search algorithm on a quantum computer and compare their results with the simulated results.

## Acknowledgements

I would like to thank Adrian Ottewill for supervising me for this project. His guidance and advice really made the project enjoyable for me, and pushed me to achieve as much as I could.

I am also thankful to Anton Dekusar for his help in gaining access to a higher-qubit quantum computer. A lot of this project wouldn't have been possible without his help.

## Contents

## List of Figures

# 1 Introduction

## 1.1 Bra-Ket Notation and Quantum Mechanics

Quantum computers use the circuit model of computation to process instructions. Each instruction can be broken down into smaller logical pieces. In classical computers we have the NOT, AND, OR, NAND, NOR, XOR and XNOR gates which take one or two bits as inputs and output a single bit. Quantum computers makes use of the same kind of circuits, but with a new set of gates which obey the postulates of quantum computing[1].

Quantum bits, or 'qubits', can take the values $|0\rangle$ and $|1\rangle$ (referred to as "ket 0" and "ket 1", using Dirac's Bra-ket notation) which correspond to the values 0 and 1 in classical computers. They can also exist in linear combinations of these states,

$$|\psi\rangle = c_1 |0\rangle + c_2 |1\rangle \tag{1}$$

where $c_1$ and $c_2$ are complex values with $|c_1|^2 + |c_2|^2 = 1$. These form a unit vector in a 2-dimensional Hilbert space whose basis states are $|0\rangle$ and $|1\rangle$. Each state vector in this space has a corresponding dual vector $\langle\psi|$ (or "bra $\psi$"), which is defined as

$$\langle\psi| = c_1^* \langle 0| + c_2^* \langle 1| \,. \tag{2}$$

The inner product of a vector in the dual space with a vector in the Hilbert space is written as $\langle\alpha|\beta\rangle$. Since $|0\rangle$ and $|1\rangle$ are orthogonal unit vectors, we have

$$\begin{aligned} \langle 0|1\rangle &= \langle 1|0\rangle = 0, \\ \langle 0|0\rangle &= \langle 1|1\rangle = 1. \end{aligned} \tag{3}$$

This allows us to define the inner product of two states as

$$\begin{aligned} \langle\alpha|\beta\rangle &= (a_1^* \langle 0| + a_2^* \langle 1|)(b_1 |0\rangle + b_2 |1\rangle) \\ &= a_1^* b_1 \langle 0|0\rangle + a_1^* b_2 \langle 0|1\rangle + a_2^* b_1 \langle 1|0\rangle + a_2^* b_2 \langle 1|1\rangle \\ &= a_1^* b_1 + a_2^* b_2. \end{aligned} \tag{4}$$

Qubit states can be expressed in matrix form, where

$$\begin{aligned} |0\rangle \Leftrightarrow \begin{pmatrix} 1 \\ 0 \end{pmatrix},\ |1\rangle \Leftrightarrow \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ \langle 0| \Leftrightarrow \begin{pmatrix} 1 & 0 \end{pmatrix},\ \langle 1| \Leftrightarrow \begin{pmatrix} 0 & 1 \end{pmatrix}. \end{aligned} \tag{5}$$

So $|\psi\rangle$ and $\langle\psi|$ may be rewritten as

$$\begin{aligned} |\psi\rangle = c_1 |0\rangle + c_2 |1\rangle \Leftrightarrow c_1 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + c_2 \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} \\ \langle\psi| = c_1^* \langle 0| + c_2 \langle 1| \Leftrightarrow c_1^* \begin{pmatrix} 1 & 0 \end{pmatrix} + c_2^* \begin{pmatrix} 0 & 1 \end{pmatrix} = \begin{pmatrix} c_1^* & c_2^* \end{pmatrix}. \end{aligned} \tag{6}$$

Outer products are written in bra-ket notation as $|\alpha\rangle\langle\beta|$, for $\alpha$ and $\beta$ in the Hilbert Space. Operators are linear combinations of outer products, and can act on either bras or kets. Another way of expressing operators are as $N \times N$ matrices, where $N$ is the dimension of the Hilbert space. They follow the same rules as matrix multiplication, so for an operator $\mathbf{X}$, $\mathbf{X}|\phi\rangle$ is a valid statement, whereas $|\phi\rangle \mathbf{X}$ is not.

The dual of $\mathbf{X}|\phi\rangle$ is written as $\langle\phi| \mathbf{X}^\dagger$, where $\mathbf{X}^\dagger$ is called the adjoint operator of $\mathbf{X}$. It is obtained by taking the conjugate transpose of $\mathbf{X}$. If $\mathbf{X} = \mathbf{X}^\dagger$, the operator is Hermitian. An operator is unitary if $\mathbf{X}^\dagger\mathbf{X} = \mathbf{X}\mathbf{X}^\dagger = \mathbb{1}$, with $\mathbb{1}$ being the identity operator.

We consider the multiple qubits at once by taking their direct (tensor) product. A register of 5 qubits in the state $|10110\rangle$ can be expressed as $|22\rangle_5$ since 10110 is the binary expansion of 22. $|10110\rangle$ and its dual are written as

$$\begin{aligned} |22\rangle_5 = |10110\rangle = |1\rangle \otimes |0\rangle \otimes |1\rangle \otimes |1\rangle \otimes |0\rangle \\ \langle 22|_5 = \langle 01101| = \langle 0| \otimes \langle 1| \otimes \langle 1| \otimes \langle 0| \otimes \langle 1|. \end{aligned} \tag{7}$$

Since the 5 qubit states that compose this vector are each vectors in a 2-dimensional Hilbert space, their direct product form a vector in a $2^5 = 32$-dimensional Hilbert space.

Two important postulates of quantum mechanics are Born's rule and the collapse hypothesis. Born's rule states that the act of measuring a state with a Hermitian operator $\mathbf{A}$ results in one of its eigenvalues, and the probability of obtaining a nondegenerate eigenvalue $a$ is given by $|\langle a|\Psi\rangle|^2$, where $|a\rangle$ is the eigenvector that corresponds to $a$. The collapse hypothesis states that once a state has been measured by $\mathbf{A}$ with result $a$, the system is described by state vector $|a\rangle$ if $a$ is an nondegenerate eigenvalue, or a linear combination of the corresponding eigenvectors if $a$ is a degenerate eigenvalue.

## 1.2 The Hadamard Gate

The first quantum gate we look at is the Hadamard gate, which acts on one qubit and outputs one qubit. In matrix form, it is written as

$$\mathbf{H} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{8}$$

When it acts on the basis states $|0\rangle$ and $|1\rangle$, it gives the following states

$$\begin{aligned} \mathbf{H}|0\rangle &= \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix} = |u\rangle \\ \mathbf{H}|1\rangle &= \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ -1 \end{pmatrix} = |v\rangle \end{aligned} \tag{9}$$

where we define $|u\rangle$ and $|v\rangle$ as the states $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ respectively.

Generalising this to multi-qubit systems, we can define $\mathbf{H}^{\otimes n}$ as $n$ Hadamard gates acting on $n$ qubits in the circuit. For 2 qubits,

$$\mathbf{H} \otimes \mathbf{H}(|00\rangle) = \frac{1}{2}(|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle) = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle). \tag{10}$$

Extending this to $n$ qubits, we find

$$\mathbf{H}^{\otimes n}(|0\rangle \otimes ... \otimes |0\rangle) = \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle_n \tag{11}$$

The Hadamard gates convert the state $|0\rangle_n$ into a linear combination of the $2^n$ basis states of this $n$-bit register. We denote this state as $|\phi\rangle$. Adding one qubit to the system allows us to consider twice the number of states $|x\rangle_n$ simultaneously, so the amount of simultaneous operations possible scales much faster on a quantum computer than on a classical one. This concept of processing $2^n$ values at once with $n$ qubits is called quantum parallelism, and allows us to perform certain calculations in less steps than previously thought possible.

## 1.3 Shor's Algorithm

### 1.3.1 Discrete Fourier Transform

The discrete Fourier transform takes a sequence of $N$ complex numbers $f_j$ (with $j$ ranging from 0 to $N-1$) and maps it to another set of $N$ complex numbers $h_j$ according to the following rule

$$h_j = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} f_k e^{2\pi ijk/N}. \tag{12}$$

This can be expressed as an $N \times N$ matrix such that $\mathbf{h} = \mathbf{U}_{DFT}\mathbf{f}$, where $\mathbf{f}$ is an $N \times 1$ column vector with components $\mathbf{f}_j = \frac{1}{\sqrt{N}} f_{j-1}$. The components of the discrete Fourier transform matrix are

$$(\mathbf{U}_{DFT})_{jk} = e^{2\pi i(j-1)(k-1)/N} \tag{13}$$

To obtain a Fourier transform of $N$ numbers, $N^2$ calculations are carried out. A more efficient way to calculate this, the Fast Fourier Transform, was developed in 1965 which reduced the required calculations to $Nlog(N)$ as $N$ approaches infinity[2]. There is no solution involving quantum computers that would reduce the number of calculations for the full Fourier transform, but quantum entanglement and interference allow us to use the Fourier transform to deduce another property of discrete functions more efficiently.

A list of $N$ values $f_k$ is said to be periodic with period $r$ if for all non-negative $n$ and $k$, $f_k = f_{k+nr}$ when $k + nr \leq N - 1$. We can assume that $r$ divides $N$ evenly for the purpose of this calculation. We generally look for values of $r$ where $r \ll N$, so any remainder of $\frac{N}{r}$ will be insignificant. We find the Fourier transform of $f$ to be the following sequence.

$$\begin{aligned}\sqrt{N}h_j &= \sum_{k=0}^{N-1} f_k e^{2\pi ijk/N} \\ &= \sum_{k=0}^{r-1} f_k e^{2\pi ijk/N} + f_{k+r} e^{2\pi i\frac{j(k+r)}{N}} + ... + f_{k+r(m-1)} e^{2\pi ij(k+r(m-1))/N} \\ &= \sum_{k=0}^{r-1} f_k e^{2\pi ijk/N} \frac{1-e^{2\pi ij}}{1-e^{2\pi ijr/N}}.\end{aligned} \tag{14}$$

The numerator of the last expression will vanish for every $h_j$ because $2\pi ij$ is an integer multiple of $2\pi i$, so $h_j = 0$ as long as the denominator does not vanish along with it. Since the denominator vanishes when $e^{2\pi i\frac{jr}{N}} = 1$, $\frac{jr}{N}$ must be an integer for the Fourier transform to be nonzero.

When we calculate the discrete Fourier transform of a periodic function, the nonzero values of $h_j$ will correspond to the values where $\frac{jr}{N}$ is an integer. This allows us to determine that the period of the function is a multiple of $\frac{N}{j}$.

### 1.3.2 Quantum Fourier Transform

The quantum Fourier transform is a unitary operator which takes an input state $|j\rangle_n$ and transforms it according to the following relation.

$$\mathbf{QFT}\,|j\rangle_n = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi ijk/N} |k\rangle_n\,. \tag{15}$$

From now on, $N$ is taken to be $2^n$, the dimension of the Hilbert space of a system of $n$ qubits. In matrix form, this gives

$$\mathbf{QFT}_{kj} = e^{2\pi ikj/2^n}. \tag{16}$$

Like its classical counterpart, the quantum Fourier transform can be used to find the period of a series of values. A classical computer does this by calculating the full discrete Fourier transform of the function, which can take a large amount of processing power as the number of $f_j$ terms increase. A quantum computer with $n$ qubits can calculate Fourier transforms of a sequence of $N = 2^n$ values, so if we have a quantum computer with 100 qubits, it would be possible to find the Fourier transform for a sequence of about $10^{30}$ values. The disadvantage of this shortcut is that we are only able to determine certain properties of the result, instead of obtaining the full Fourier transform.

When we measure the output of the **QFT** gate, we will obtain a state corresponding to one of the non-null values of the Fourier transform, with probability determined by Born's rule. The collapse hypothesis states that any further measurement of the system will result in the same state. We can get more information about the Fourier transform by computing it several times and measuring the value each time. If a function has period $r$, its Fourier transform will nonzero at the values where $\frac{jr}{N} \in \mathbb{Z}$.

### 1.3.3 Period Finding using Quantum Fourier Transform

To investigate a function $f$, we first need to find $f(x)$ for each value of $x$ using quantum gates. We define a new quantum gate $\mathbf{U}_f$ which takes $k+l$ inputs, where $k$ and $l$ are the binary lengths of the control register and target register of $f$ respectively. It is defined as

$$\mathbf{U}_f \left|x\right\rangle_k \otimes \left|y\right\rangle_l = \left|x\right\rangle_k \otimes \left|f(x) \oplus y\right\rangle_l . \tag{17}$$

Here, $x$ is the input of the function and $y$ is the value of $f(x)$ we compare against. $\mathbf{U}_f$ will return zero only if $f(x) = y$.

We can now use the gates we have defined so far to obtain the period of $f$. We first look at $f : \{0,1\}^n \to \{0,1\}$ for simplicity. Its output register will be a single qubit, so $l = 1$. This can be extended to functions which return larger values by adding more qubits to the target register. We now define $\mathbf{U}_f$ as before and have it act on the input state $\mathbf{H}^{\otimes n} \left|0\right\rangle_n \otimes \left|0\right\rangle$.

$$\begin{aligned}\mathbf{U}_f(\mathbf{H}^{\otimes n} \left|0\right\rangle_n \otimes \left|0\right\rangle) &= \mathbf{U}_f(\left|\phi\right\rangle_n \otimes \left|0\right\rangle) \\ &= \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} \left|x\right\rangle_n \otimes \left|f(x) \oplus 0\right\rangle \\ &= \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} \left|x\right\rangle_n \otimes \left|f(x)\right\rangle\end{aligned} \tag{18}$$

which gives us the superposition of all input values and their corresponding outputs according to $f(x)$. Measuring this state would cause it to collapse into one of its possible states $|x\rangle_n \otimes |f(x)\rangle$. If we operate on this state with the **QFT** gate before measuring it, we obtain the following state, which we call $|\psi\rangle$. An illustration of this circuit is shown in Figure 1.

$$\begin{aligned}(\mathbf{QFT} \otimes \mathbb{1})\mathbf{U}_f(\mathbf{H}^{\otimes n} \otimes \mathbb{1})\,|0\rangle_{n+1} &= \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} \mathbf{QFT}\,|k\rangle_n \otimes |f(k)\rangle \\ &= \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} e^{2\pi ijk/2^n}\,|j\rangle_n \otimes |f(k)\rangle \qquad (19) \\ &= \frac{1}{2^n} \sum_{k=0}^{2^n-1} \sum_{j=0}^{2^n-1} e^{2\pi ijk/2^n}\,|j\rangle_n \otimes |f(k)\rangle\end{aligned}$$

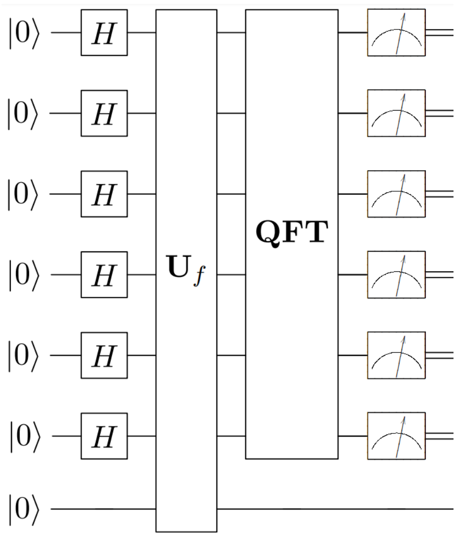


Figure 1: Period Finding Circuit for $n = 6$

The last qubit takes the value $f(k)$, which is either 1 or 0 corresponding to each value of $k$. We measure the first $n$ qubits to find a state $|m\rangle$ where $p(m)$, the chance of measuring this value, is greater than zero. According to Born's rule, the probability $p(m)$ is the sum of the probabilities of all states which would yield this measurement, or in this case, $|m\rangle_n \otimes |0\rangle$ and $|m\rangle_n \otimes |1\rangle$, which we write as $|m0\rangle$ and $|m1\rangle$ respectively. With this, we find

$$\begin{aligned}p(m) &= p(m0) + p(m1) \\ &= |\,\langle 0m|\psi\rangle\,|^2 + |\,\langle 1m|\psi\rangle\,|^2.\end{aligned} \qquad (20)$$

We look at the second term of this, which is the probability that the final state of the system is equal to $|m\rangle_n \otimes |1\rangle$.

$$\begin{aligned} |\langle 1m|\psi\rangle|^2 &= \left| \frac{1}{2^n} \sum_{k=0}^{2^n-1} \sum_{j=0}^{2^n-1} e^{2\pi ijk/2^n} \langle m|j\rangle \langle 1|f(k)\rangle \right|^2 \\ &= \frac{1}{2^{2n}} \left| \sum_{k=0}^{2^n-1} \sum_{j=0}^{2^n-1} e^{2\pi ijk/2^n} \delta_{mj} \langle 1|f(k)\rangle \right|^2 \\ &= \frac{1}{2^{2n}} \left| \sum_{k=0}^{2^n-1} f_k e^{2\pi imk/2^n} \right|^2, f_k = \langle 1|f(k)\rangle \\ &= \frac{1}{2^n} |h_m|^2 \\ &= \frac{1}{2^{2n}} \left| \sum_{k=0}^{r-1} f_k e^{2\pi imk/2^n} \frac{1-e^{2\pi im}}{1-e^{2\pi imr/2^n}} \right|^2 . \end{aligned} \tag{21}$$

This expression is zero if $mr/2^n \notin \mathbb{Z}$, or equivalently, $|\langle 1m|\psi\rangle|^2 \neq 0$ implies $mr/2^n$ is an integer. A similar calculation is true for $|\langle 0m|\psi\rangle|^2$, where we take $f_k$ be the values of $\langle 0|f(k)\rangle$. Extending this to functions with multi-bit outputs, we can apply the same argument for $|\langle 2m|\psi\rangle|^2$, $|\langle 3m|\psi\rangle|^2 ... |\langle (2^l-1)m|\psi\rangle|^2$. So, if $p(m) = \sum_{i=0}^{2^l-1} |(\langle i| \otimes \langle m|)|\psi\rangle|^2$ is non-zero then we know $\frac{mr}{2^n}$ must be an integer.

Since we have a value of $m$ where $p(m) \neq 0$, obtained from measurement of the first $n$ qubits, we can deduce that $r$ must be an integer multiple of $\frac{2^n}{m}$. By taking several measurements of $m$ (which is done by repeating the calculation with a set of 'fresh' qubits each time) we can then find a good estimate of $r$ to be the least common denominator of the set of fractions $\frac{m}{2^n}$.

### 1.3.4 Shor's Algorithm

Each number has a unique expansion as a product of prime numbers. Finding the prime factors of large numbers is usually achievable on a classical computer, but for larger numbers, such as RSA-220 which is a 220 digit semi-prime (a number with only 2 prime factors), this can take significant time and resources[3]. RSA encryption relies on the fact that is is easy to check whether a number is a factor of a large semi-prime, but difficult to work out what those factors are.

Miller's algorithm is a method of finding factors of a semi-prime (or any composite) number $N = pq$ by finding the smallest nonzero value of $r$ such that

$$a^r = 1 \bmod N, \tag{22}$$

where $a$ is any number where $a < r$ and $gcd(a, N) = 1$[4]. Once $r$ is known (and assuming $r$ is even), the factors of $N$ ($p$ and $q$) are found using the following equation

$$\begin{aligned}
a^r &= 1 \bmod N \\
a^r - 1 &= 0 \bmod N \\
(a^{r/2} - 1)(a^{r/2} + 1) &= 0 \bmod (pq) \\
(a^{r/2} - 1)(a^{r/2} + 1) &= Mpq, M \in \mathbb{Z} \\
\frac{(a^{r/2} - 1)(a^{r/2} + 1)}{pq} &\in \mathbb{Z} \\
gcd(a^{r/2} - 1, N) &= p \\
gcd(a^{r/2} + 1, N) &= q.
\end{aligned} \tag{23}$$

The difficult part of factorising large numbers now becomes finding the value of $r$. Considering the equation $f(x) = a^x \bmod n$, we have

$$\begin{aligned}
f(x + yr) &= a^{x+yr} \bmod n \\
&= (a^{yr} \bmod n)(a^x \bmod n) \bmod n \\
&= 1^y (a^x \bmod n) \\
&= f(x).
\end{aligned} \tag{24}$$

So $r$ must be the period of $f$. We can use the period finding algorithm described in the previous section to find $r$ in polynomial time for any value of $N$ with an amount of qubits which grows linearly with $n$, the binary length of $N$. This is what gives Shor's algorithm the title of a 'Quantum Killer App', and is what lead to the development of the field of quantum computing today.

## 1.4 Grover's Search Algorithm

Another app which helped kick-start the field of quantum computing is Grover's Search Algorithm[9]. We are given a classical gate ('the oracle') with $n$ binary inputs, which outputs the value 1 only when given a particular set of inputs, and zero otherwise. This is analogous to machine which can scan items one at a time until it finds the correct one. On average, it takes $N/2$ tries to find the correct item (where $N = 2^n$).

This number can be reduced if we replace the classical oracle with a quantum gate with the same behaviour. We define $\xi$ as a number between 0 and $N-1$, and $f(x)$ as a function $f : \{0,1\}^n \to \{0,1\}$, with $f(\xi) = 1$, and $f(x) = 0$ otherwise. The oracle can now be expressed as the quantum gate $\mathbf{U}_f$, which is defined as before, with $n$ qubits in the control register and one target qubit, with

$$\mathbf{U}_f \left|x\right\rangle_n \otimes \left|y\right\rangle = \left|x\right\rangle_n \otimes \left|f(x) \oplus y\right\rangle . \tag{25}$$

If we initialise the target register with the state $\mathbf{H}\left|1\right\rangle$, this equation becomes

$$\begin{aligned} \mathbf{U}_f \left|x\right\rangle_n \otimes \mathbf{H}\left|1\right\rangle &= \left|x\right\rangle_n \otimes (\left|f(x)\right\rangle \oplus \frac{1}{\sqrt{2}}(\left|0\right\rangle - \left|1\right\rangle)) \\ &= \begin{cases} \left|x\right\rangle_n \otimes \frac{1}{\sqrt{2}}(\left|0\right\rangle - \left|1\right\rangle), & x \neq \xi \\ \left|x\right\rangle_n \otimes \frac{1}{\sqrt{2}}(\left|1\right\rangle - \left|0\right\rangle), & x = \xi \end{cases} \end{aligned} \tag{26}$$

These results are the same up to a change in phase by a factor of -1. This allows us to rewrite the result as

$$\mathbf{U}_f(\left|x\right\rangle_n \otimes \mathbf{H}\left|1\right\rangle) = (-1)^{f(k)} \left|x\right\rangle_n \otimes \mathbf{H}\left|1\right\rangle . \tag{27}$$

We now define an operator $\mathbf{V}$ which acts solely on the control register, where

$$\mathbf{V}\left|x\right\rangle_n = (-1)^{f(k)} \left|x\right\rangle_n . \tag{28}$$

With this, we obtain the relation

$$\mathbf{U}_f(\left|x\right\rangle_n \otimes \mathbf{H}\left|1\right\rangle) = (\mathbf{V}\left|x\right\rangle_n) \otimes \mathbf{H}\left|1\right\rangle . \tag{29}$$

Applying $\mathbf{H}$ gates to the control register, we obtain the superposition of all states, $|\phi\rangle$, as described in Equation 11.

$$\mathbf{U}_f(\mathbf{H}^{\otimes n} |0\rangle_n \otimes \mathbf{H} |1\rangle) = \mathbf{V} |\phi\rangle \otimes \mathbf{H} |1\rangle . \tag{30}$$

We introduce the diffusion operator $\mathbf{W}$, which is defined as follows. We also re-express $\mathbf{V}$ in Dirac's bra-ket notation.

$$\begin{aligned} \mathbf{V} &= \mathbb{1} - 2 |\xi\rangle \langle\xi| \\ \mathbf{W} &= 2 |\phi\rangle \langle\phi| - \mathbb{1} \\ \mathbf{G} &= \mathbf{WV} \end{aligned} \tag{31}$$

Here we have defined the Grover operator, $\mathbf{G}$, as the application of the $\mathbf{V}$ and $\mathbf{W}$ gates on the control register. We repeat this operation $R$ times, so the final state to be measured is

$$\mathbf{G}^R |\phi\rangle \otimes \mathbf{H} |1\rangle . \tag{32}$$

One iteration of the Grover operator affects the control register in the following way.

$$\begin{aligned} \mathbf{V} : a|\phi\rangle + b|\xi\rangle &\mapsto \begin{pmatrix} a & b \end{pmatrix} \begin{pmatrix} 1 & -\frac{2}{\sqrt{N}} \\ 0 & -1 \end{pmatrix} \begin{pmatrix} |\phi\rangle \\ |\xi\rangle \end{pmatrix} \\ \mathbf{W} : a|\phi\rangle + b|\xi\rangle &\mapsto \begin{pmatrix} a & b \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \frac{2}{\sqrt{N}} & -1 \end{pmatrix} \begin{pmatrix} |\phi\rangle \\ |\xi\rangle \end{pmatrix} \\ \mathbf{VW} = \mathbf{G} : a|\phi\rangle + b|\xi\rangle &\mapsto \begin{pmatrix} a & b \end{pmatrix} \begin{pmatrix} 1 & -\frac{2}{\sqrt{N}} \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \frac{2}{\sqrt{N}} & -1 \end{pmatrix} \begin{pmatrix} |\phi\rangle \\ |\xi\rangle \end{pmatrix} \\ \mathbf{G} : a|\phi\rangle + b|\xi\rangle &\mapsto \begin{pmatrix} a & b \end{pmatrix} \begin{pmatrix} 1 - \frac{4}{N} & \frac{2}{\sqrt{N}} \\ -\frac{2}{\sqrt{N}} & 1 \end{pmatrix} \begin{pmatrix} |\phi\rangle \\ |\xi\rangle \end{pmatrix} \end{aligned} \tag{33}$$

To illustrate the effect of this operation on the probability of a successful measurement, we define

$$\begin{aligned}
|\phi\rangle &= \frac{\sqrt{N-1}}{\sqrt{N}}\,|\eta\rangle + \frac{1}{\sqrt{N}}\,|\xi\rangle \\
\Leftrightarrow |\eta\rangle &= \frac{\sqrt{N}}{\sqrt{N-1}}\,|\phi\rangle - \frac{1}{\sqrt{N-1}}\,|\xi\rangle\,,
\end{aligned} \tag{34}$$

where $|\eta\rangle$ is the superposition of all states except $|\xi\rangle$. The definition of $\mathbf{G}$ then becomes

$$\begin{aligned}
\mathbf{G}\,|\xi\rangle &= -\frac{2}{\sqrt{N}}\,|\phi\rangle + |\xi\rangle \\
&= -\frac{2\sqrt{N-1}}{N}\,|\eta\rangle + \frac{N-2}{N}\,|\xi\rangle \\
\mathbf{G}\,|\phi\rangle &= \left(1-\frac{4}{N}\right)|\phi\rangle + \frac{2}{\sqrt{N}}\,|\xi\rangle \\
&= \left(1-\frac{4}{N}\right)\left(\frac{\sqrt{N-1}}{\sqrt{N}}\,|\eta\rangle + \frac{1}{\sqrt{N}}\,|\xi\rangle\right) + \frac{2}{\sqrt{N}}\,|\xi\rangle \\
&= \frac{\sqrt{N-1}}{N\sqrt{N}}(N-4)\,|\eta\rangle + \frac{1}{N\sqrt{N}}\,(3N-4)\,|\xi\rangle \\
\mathbf{G}\,|\eta\rangle &= \frac{\sqrt{N}}{\sqrt{N-1}}\mathbf{G}\,|\phi\rangle - \frac{1}{\sqrt{N-1}}\mathbf{G}\,|\xi\rangle \\
&= \frac{N-4}{N}\,|\eta\rangle + \frac{3N-4}{N\sqrt{N-1}}\,|\xi\rangle + \frac{2}{N}\,|\eta\rangle - \frac{N-2}{N\sqrt{N-1}}\,|\xi\rangle \\
&= \frac{N-2}{N}\,|\eta\rangle + \frac{2N-2}{N\sqrt{N-1}}\,|\xi\rangle \\
&= \frac{N-2}{N}\,|\eta\rangle + \frac{2\sqrt{N-1}}{N}\,|\xi\rangle
\end{aligned} \tag{35}$$

$$\mathbf{G} : a\,|\eta\rangle + b\,|\xi\rangle \mapsto \begin{pmatrix} a & b \end{pmatrix} \begin{pmatrix} \frac{N-2}{N} & \frac{2\sqrt{N-1}}{N} \\ -\frac{2\sqrt{N-1}}{N} & \frac{N-2}{N} \end{pmatrix} \begin{pmatrix} |\eta\rangle \\ |\xi\rangle \end{pmatrix} \tag{36}$$

Since $\left(\frac{N-2}{N}\right)^2 + \left(\frac{2\sqrt{N-1}}{N}\right)^2 = \frac{N^2}{N^2} = 1$, we can rewrite these values as $\sin\theta$ and $\cos\theta$.

$$\mathbf{G} = \begin{pmatrix} \frac{N-2}{N} & \frac{2\sqrt{N-1}}{N} \\ -\frac{2\sqrt{N-1}}{N} & \frac{N-2}{N} \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}. \tag{37}$$

Repeated iterations of the Grover operator now gives the result

$$\mathbf{G}^R = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}^R = \begin{pmatrix} \cos R\theta & \sin R\theta \\ -\sin R\theta & \cos R\theta \end{pmatrix}. \tag{38}$$

To find the value $R$ which maximises the probability of measuring $|\xi\rangle$, we first calculate $p(\xi) = |\langle\xi|\mathbf{G}^R|\phi\rangle|^2$. Since $\mathbf{G}^R$ is a unitary operator in the space spanned by $|\eta\rangle$ and $|\xi\rangle$, the operator $|\eta\rangle\langle\eta| + |\xi\rangle\langle\xi|$ results in the identity operator. We split this equation into

$$\begin{aligned}
p(\xi) &= |\langle\xi|\mathbf{G}^R\mathbb{1}|\phi\rangle|^2 \\
&= |\langle\xi|\mathbf{G}^R|\eta\rangle\langle\eta|\phi\rangle + \langle\xi|\mathbf{G}^R|\xi\rangle\langle\xi|\phi\rangle|^2 \\
&= \left|\frac{\sqrt{N-1}}{\sqrt{N}}\langle\xi|\mathbf{G}^R|\eta\rangle + \frac{1}{\sqrt{N}}\langle\xi|\mathbf{G}^R|\xi\rangle\right|^2 \\
&= \frac{1}{N}\left|\sqrt{N-1}sin(R\theta) + cos(R\theta)\right|^2 \\
&\to \frac{N-1}{N} \text{ as } R\theta \to \frac{\pi}{2} \\
&\to 1 \text{ as } N \to \infty
\end{aligned} \tag{39}$$

And since $\sin\theta = \frac{2\sqrt{N-1}}{N} \to \frac{2}{\sqrt{N}}$ and $\sin\theta \to \theta$ as $N \to \infty$, we have $\theta$ approximately equal to $\frac{2}{\sqrt{N}}$. With this, we find the optimal value of $R$ to be

$$R = \frac{\pi}{4}\sqrt{N}. \tag{40}$$

## 2 Methods

### 2.1 Shor's Algorithm

#### 2.1.1 Simulating Shor's Algorithm Classically

We use Fast Fourier Transform to simulate the quantum step of Shor's algorithm. The code factors a semi-prime number $N$ by taking the discrete Fourier transform of the function $f(x) = a^x \bmod N$, for some $a$ that does not share any factors with $N$. The result of this is the graph shown in Figure 3 when we set $N = 33$ and $a = 5$. This code emulates the quantum circuit diagram shown in Figure 2, where the results of the modular exponentiation gates and the Fourier transform gate are calculated classically.

The size of the input register, $n$, will determine the maximum value that can be obtained for the output of the Fourier transform, $2^n$. The length of the target register, $l$, is some number such that $N < 2^l$. We generally choose both $l$ and $n$ to be the smallest such number, but $n$ can be decreased if we know the value of the period is small, or

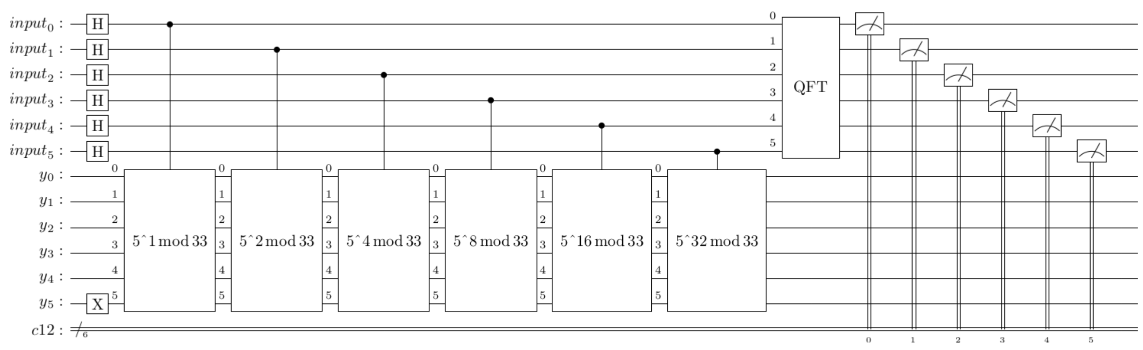


Figure 2: Circuit diagram to find the period of $5^x$ mod 33 with 6 qubits in the input and target registers

increased if we want the the peaks of the graph to be more pronounced and give less chance of finding an incorrect value of $r$.

Next, we choose a value where the Fourier transform is nonzero and let it equal $m2^n/r$ for some integer $m$. We divide this by $2^n$ and use convergent fractions to obtain the denominator of $m/r$, which is the period of the function. If r is odd then we select a new point on the graph. We select a new seed value $x$ if no valid period is found for $a^x$ mod $N$.

Once we have $r$, we use equation 23 to obtain the factors of $N$, and verify $pq = N$. If no factors are found, we repeat with a new value of $x$.

In the example case where $N = 33$, we find the period of $5^r$ mod 33 to be 10, and $5^{10/2} - 1 = 3124$, $5^{10/2} + 1 = 3126$. Since $3124 = 2 * 2 * 11 * 71$, we find its greatest common divisor with 33 is 11 using the Euclidean algorithm, which is indeed a factor of 33. Similarly, $3126 = 2 * 3 * 521$, so we find the second factor is 3.

Since we use the full results of a classical discrete Fourier transform to simulate this, this code is able to scale with $Nlog(N)$ as we factorise larger numbers, so it takes over twice as long to factorise a number which is twice as large.

### 2.1.2 Simulating Shor's Algorithm with Qiskit

Qiskit is an open-source SDK for working with quantum computers at the level of pulses, circuits, and application modules[5]. Using Qiskit, we can build up the the gates required to run Shor's algorithm on a quantum computer, or on classical hardware using a quantum computer simulator such as the Qiskit Aer simulator. We also use Qiskit to create the

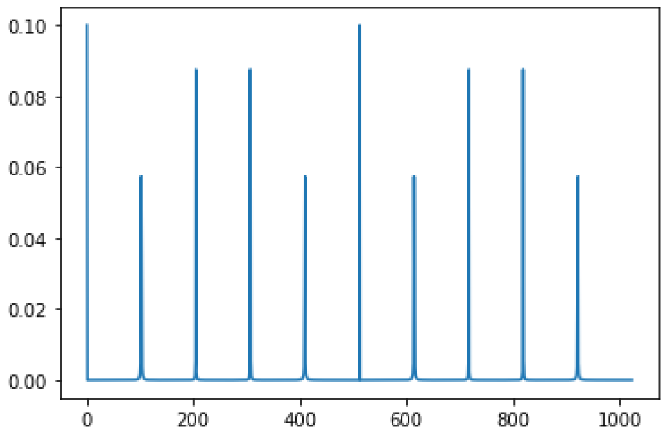


Figure 3: Simulated Fourier transform of $5^x$ mod 33 with 10 input qubits

circuit diagrams used in this report. We recreate the modular exponentiation gates needed for Shor's algorithm using the method outlined by Vedral[6].

### 2.1.3 Constructing the Modular Exponentiation Gate

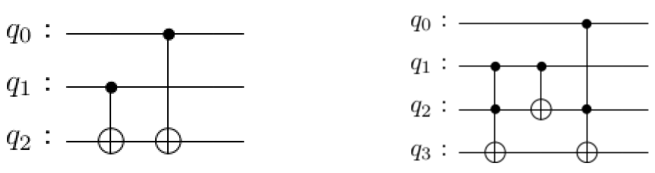


Figure 4: Sum and Carry circuits used in the adder gate

The first step to preforming arithmetic with a quantum computer is to define addition of two binary numbers $a$ and $b$. This can be done by defining the bit-wise sum and carry gates shown in Figure 4, and chaining them together using $n$ temporary qubits, and then making sure to undo the carry operation on the temporary qubits once they have been used. We do this by taking the inverse of the carry gate, which is obtained by processing

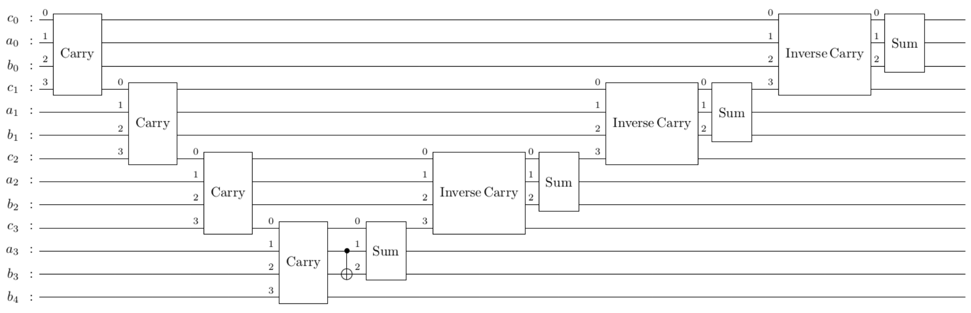


Figure 5: Adder Circuit for 4 qubits, with 4 temporary qubits

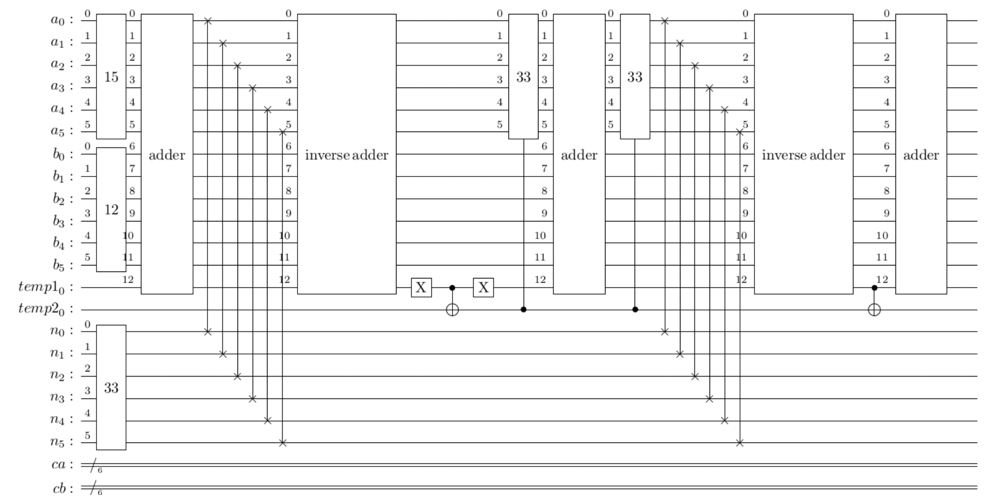


Figure 6: Modular Addition circuit for $15 + 12 \bmod 33$

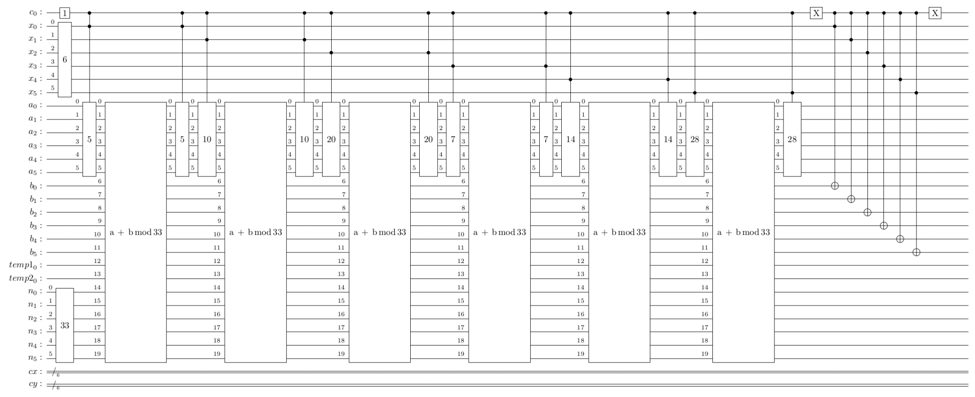


Figure 7: Controlled Multiplication circuit for $6 \cdot 5 \bmod 33$

each gate in reverse order. This composition is shown in Figure 5. This adder gate uses $3n + 1$ qubits, where $n$ is the binary length of $a$ and $b$. While this gate can be used to build up the other gates, it is inefficient in the number of qubits it uses and also is computationally expensive to run due to the double controlled X-gates (or Toffoli gates) in each of the carry gates. For this reason, we use the Draper QFT Adder[7], which require no temporary qubits.

Once we have this gate, we also obtain its inverse in the same way as before. We construct the modular addition gate, whose input is ($a$,$b$,$N$), and output is ($a$, $a + b$ mod $N$, $N$). It requires $3n$ qubits to store the length of $a$,$b$ and $N$, along with 2 temporary qubits. We then construct the circuit shown in Figure 6 and simulate several values of $a$,$b$ and $N$ to verify the results. We also obtain the inverse of this gate.

Next, we create the controlled modular multiplication gate shown in Figure 7. Its inputs are ($c$,$x$,0,$N$) with $c$ length one, $x$,0,$N$ each length $n$, and $n+2$ temporary qubits. The $a$ register is used for temporary storage, so its input and output are both 0. The numbered gates in this register signify controlled bit-wise addition of the values $a2^p \bmod N$ for $p$ between 1 and $n$, which are calculated classically. This gate returns ($c$, $x$, $ax \bmod N$, $N$) when $c$ is 1, and ($c$, $x$, $x$, $N$) otherwise. This gate works by calculating the values of $a2^p$ for each $p$ in the binary expansion of $x$, and summing them with modular addition gates to obtain $ax \bmod N$. We obtain the inverse of this gate as before. The inverse gate outputs ($c$, $x$, 0, $N$) when $c$ is one and the factor $a$ defined for this gate is the inverse of $x \bmod N$, which is found classically with the extended Euclidean algorithm. Additionally, the second and third inputs must differ by a factor of $x$, so the input must be of the form ($c$, $bx \bmod N$, $b$, $N$).

With these gates, we construct the modular exponentiation gates required by Shor's algorithm. To do this, we start with the inputs ($c$, 1, 0, $N$), and add a controlled multiplication gate with factor $a^{2^p}$ for some $p$. We then take the second and third output and swap them, so the result is ($c$, $a^{2^p} \bmod N$, 1, $N$). This is then passed through an inverse controlled multiplication gate with factor $1/a^{2^p}$ modulo $N$. This gate will return ($c$, $a^{2^p} \bmod N$, 0, $N$), freeing up the second input for use. Taking $n$ copies of this circuit in series controlled by the binary expansion of some number $x$, we use this to obtain the value of $a^x \bmod N$, as required by Shor's algorithm. These circuits are shown in Figures 8 and 9.

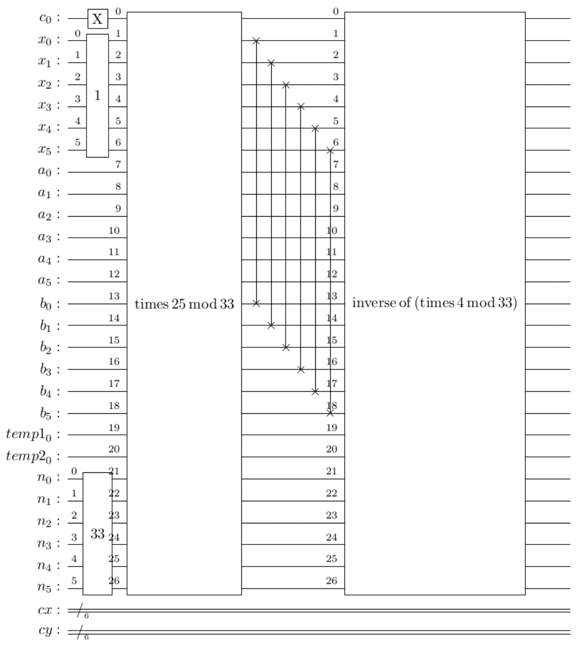


Figure 8: Modular Exponentiation circuit for $5^{2^1} \bmod 33$

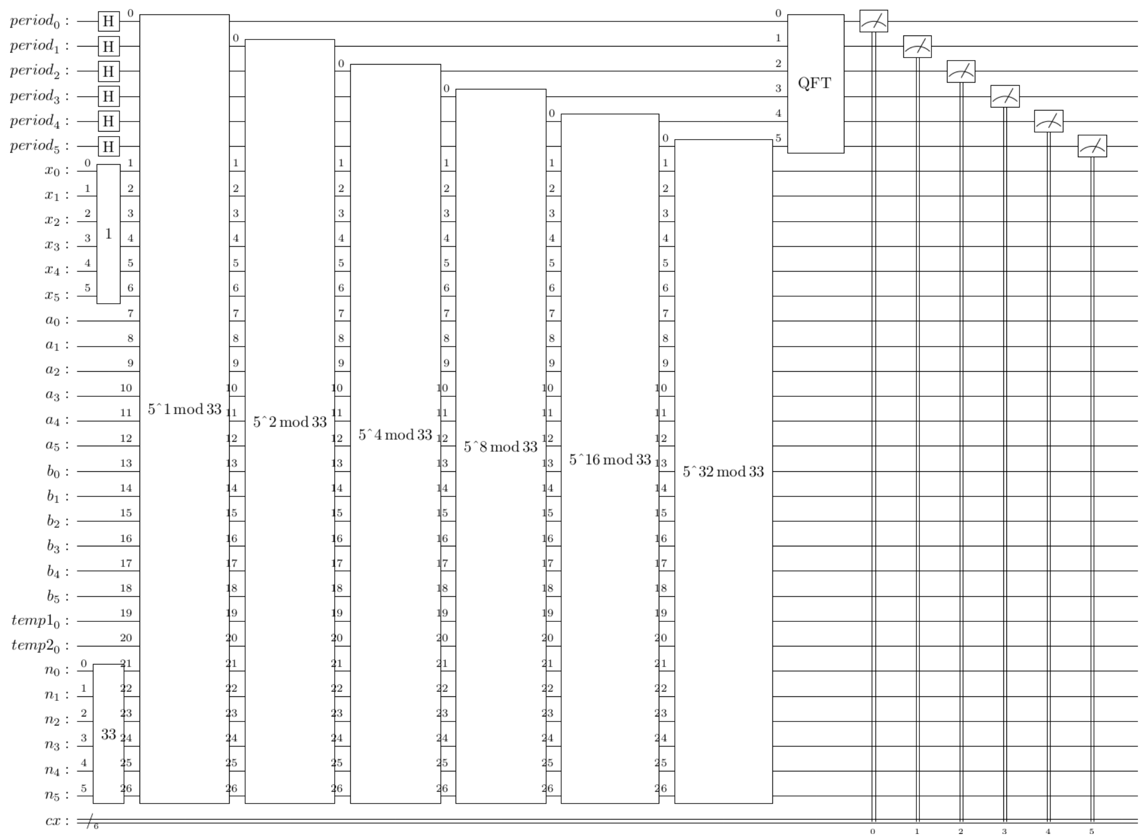


Figure 9: Completed circuit for factorising 33 with Shor's Algorithm with 32 qubits

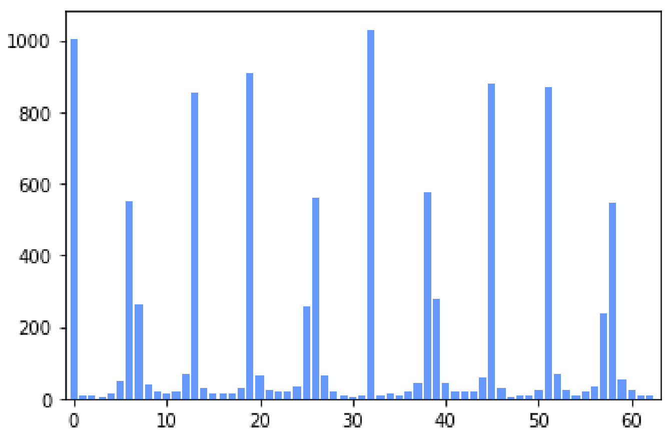


Figure 10: Fourier transform of $5^x$ mod 33 with 6 input qubits using Qiskit Aer

### 2.1.4 Results of Shor's Algorithm

We set up a circuit to find the factors of some $N < 2^n$ by taking $5n + 2$ qubits, and applying $\mathbf{H}$ gates to each qubit of the input register, and x gates on the other registers according to Figure 9, so the $x$-register is in the state $|1\rangle_n$ and the $n$-register is in the state $|N\rangle_n$. We then use $n$ controlled modular exponentiation gates, controlled by the input register. We pass the input register though a quantum Fourier transform gate and measure the results. This circuit is equivalent to the period finding circuit shown in Figure 1, with the $x$-register taking the place of the output of $\mathbf{U}_f$. Using the Qiskit Aer simulator on this circuit, we obtain the results in Figure 10, which agree with the results from Section 2.1. Applying the same method, we find the period of $5^x$ mod 33 is 10, and the factors of 33 are 3 and 11.

### 2.1.5 Running Shor's Algorithm on a quantum computer

IBM's Quantum services allow us to submit circuits created in Qiskit as jobs to be be run remotely on a quantum computer. We run Shor's algorithm on a 7 qubit quantum computer, IBM_Lagos, and use it to factor $N = 15$. We construct a circuit with 4 qubits in the target register, which is the minimum number of qubits since $15 = 2^4 - 1$.

The modular exponentiation gate $2^x$ mod 15 can be constructed by inspecting the binary expansion of the output of repeated application of multiplication by 2 mod 15. In this case, we have the series $(2, 4, 8, 1)$, which is $(0010, 0100, 1000, 0001)$ in binary. From this, we find that multiplication by 2 modulo 15 is equivalent to cycling each bit of the

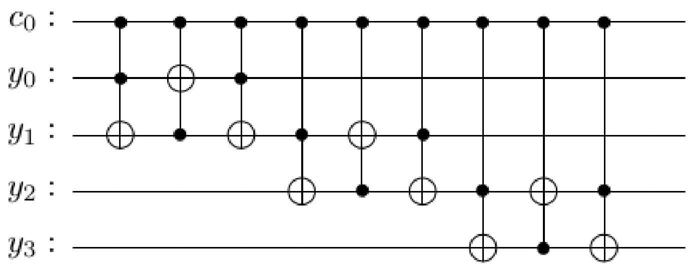


Figure 11: Gate for controlled multiplication by 2 mod 15

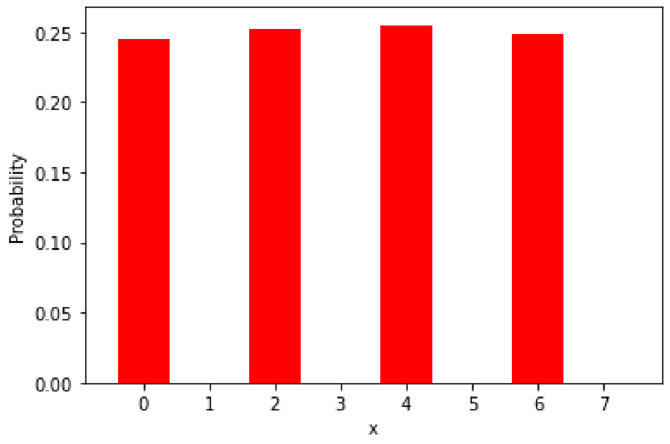


Figure 12: Fourier transform of $2^x$ mod 15 with 3 input qubits with Qiskit Aer

4-bit output once to the left. This can be done with a series of swap gates, as shown in Figure 11. $2^x$ mod 15 is found by repeating this circuit $x$ times. Construction of circuits in this way requires knowing the full sequence $(a^x \bmod N)_x$, which also means we need to know the period of the function beforehand. For this reason we are not able to generalise Shor's algorithm to one which would require a total of $2n$ qubits for larger numbers with this method.

We use 3 qubits in the input register, which would normally make it more difficult to find the correct value of $r$, but in this case the procedure is not changed. We first simulate this circuit with Qiskit Aer. The results are shown in Figure 12. The probabilities of measurement are non zero at $m = 0, 2, 4, 6$, so $r$ is the denominator of the fractions $0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}$. Taking $r = 4$, we find $2^{4/2} + 1 = 5$ and $2^{4/2} - 1 = 3$, the factors of 15.

When we run this circuit on a quantum computer, we obtain the results shown in Figure 13. These results are less obvious than the simulated run because of the noise associated with running circuits on physical quantum computers. This effect is explained in Section 3.7. Shor's algorithm gives results which indicate that $r = 4$ 50% of the time

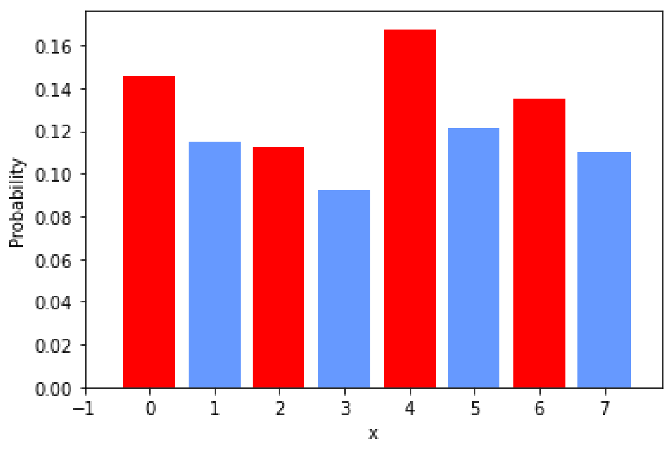


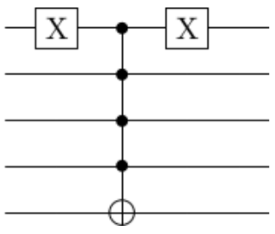
Figure 13: Results of Shor's Algorithm on IBM_Lagos

in the ideal case, when either 2 or 6 are measured (measuring 0 gives us no information about the period and measuring 4 gives $r = 2$). In this run, 2 or 6 were measured 24.8% of the time, which is slightly lower than the chance of picking these from a list of 8 numbers at random. When we run the circuit a second time, this number becomes 26.2%. This circuit is near the limit of what is achievable on a quantum computer of this noise level without using error correction. To remedy this, we could use the Shor Code, which uses approximately $9k$ qubits to ensure a $k$-qubit system is protected against single qubit-flip and phase-flip errors[8].

## 2.2 Grover's Search Algorithm

### 2.2.1 Simulating Grover's Search Algorithm with Qiskit

We simulate Grover's search algorithm for the case where $n = 4$ and $\xi = 7$. The oracle and gate and diffusion gate are constructed as shown in Figure 14. The oracle is constructed such that it only affects the target register when it receives $|7\rangle_4$ (or $|0111\rangle$). The diffusion gate is constructed using an $(n-1)$-controlled Z gate at the centre (expressed here as an $(n-1)$-controlled X gate in between two **H** gates). This gate is constructed in the same way for any $n$ greater than 1[10]. $R$ is found to be $\frac{4\pi}{4} = \pi \approx 3$, so three copies of each gate is used. We simulate this circuit with the Aer simulator and obtain the results shown in Figure 16. We run this circuit 10000 times, and we measure the output as $\xi$


(a) The oracle

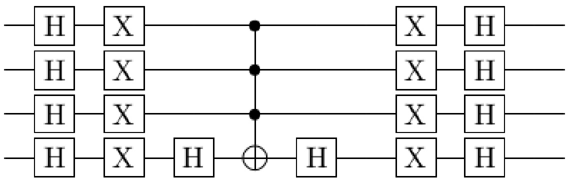

(b) The diffusion gate

Figure 14: The oracle and diffusion gate for $\xi = 7$ and $n = 4$

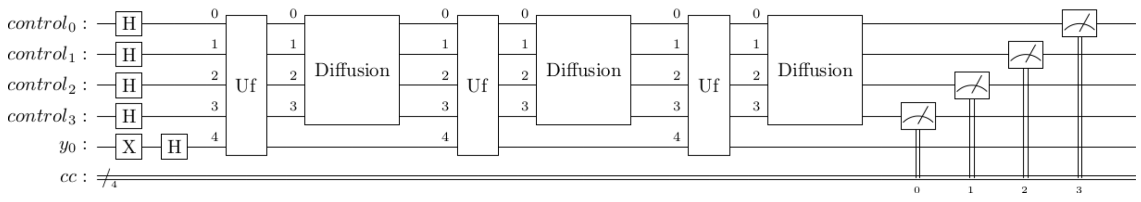


Figure 15: Circuit for Grover's search algorithm with $n = 4$

96% of the time. Since we took $R$ to be 3, $\mathbf{G}^R$ becomes

$$\mathbf{G}^3 = \begin{pmatrix} \frac{N-2}{N} & \frac{2\sqrt{N-1}}{N} \\ -\frac{2\sqrt{N-1}}{N} & \frac{N-2}{N} \end{pmatrix}^3 = \begin{pmatrix} 0.055 & 0.999 \\ 0.999 & 0.05f \end{pmatrix} \tag{41}$$

$\mathbf{G}^3$ acts on $|\phi\rangle$ as

$$\mathbf{G}^3 : \frac{\sqrt{15}}{4}|\eta\rangle + \frac{1}{4}|\xi\rangle \mapsto \begin{pmatrix} 0.968 & 0.25 \end{pmatrix} \begin{pmatrix} 0.055 & 0.999 \\ 0.999 & 0.055 \end{pmatrix} \begin{pmatrix} |\eta\rangle \\ |\xi\rangle \end{pmatrix} = 0.1967\,|\eta\rangle + 0.9804\,|\xi\rangle \,. \tag{42}$$

The expected probability of measuring $\xi$ from this state is $0.9804^2 = 0.9613$ according to Born's rule, which agrees with the simulated results.

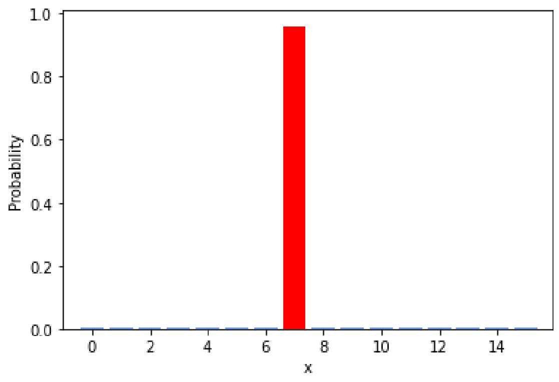


Figure 16: Results of Grover's search algorithm with the Aer simulator with $\xi = 7$

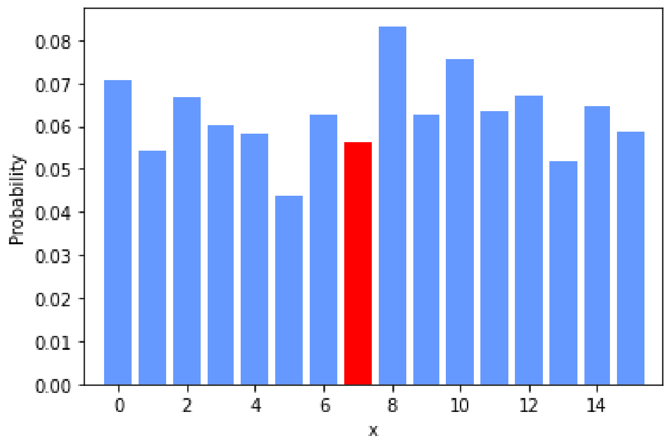


Figure 17: Results of Grover's search algorithm on IBMQ_Quito

### 2.2.2 Running Grover's Algorithm on a quantum computer

We run Grover's search algorithm for $n = 4$, $\xi = 7$ on the 5 qubit quantum computer run by IBM in Quito. The results of this are shown in Figure 17.

Physical quantum computers are limited in which qubits are allowed to interact with each other, so when a job is submitted, additional swap gates are added to ensure the circuit is able to run. The result of these additional gates is the relatively simple circuit described in Figure 15 now taking 670 lines of code corresponding to 649 quantum gates and 4 measurement gates. Additionally, each quantum computer has a listed "CNOT error," which is the chance for an error to occur each time a qubit passes through a CNOT gate. These errors are typically around 1%, and since the quantum circuit we defined now contains 467 CNOT gates, the chance of no CNOT errors occurring in a run is about 1%. 10000 runs are taken each time a circuit is run, so any noise-free runs are drowned out by runs which may be incorrect, leading to the random nature of these results.

## 3 Conclusion

The aim of this project was to investigate two of the quantum algorithms that led to the rapid development of the field of quantum computing, Shor's algorithm and Grover's search algorithm. We first looked at the derivations for each of these algorithms.

We simulated Shor's algorithm classically using the discrete Fourier transform, and then using the quantum simulator Qiskit Aer. We constructed the modular exponentiation gate required for this circuit using Vedral's method for arithmetic on quantum circuits, and compared our results with those found classically. We ran a simplified version of this circuit to find the factors of 15 with 7 qubits on the quantum computer IBM_Lagos, and obtained results which are affected by noise, leading to some incorrect measurements. The chance of obtaining the factors of 15 from one run was 24.8%, as opposed to the probability when run without noise, 50%.

We simulated Grover's search algorithm for $N = 16$, and constructed a circuit which identified the correct state 96% of the time when simulated using Aer. We then ran this on the 5-qubit quantum computer IBMQ_Quito and obtained a circuit which identified the correct state less than 1 out of 16 times. This was due to the relatively high chance of error for each CNOT gate in the circuit, as well as the large number of CNOT gates required.

We demonstrated these quantum algorithms giving the expected results on Aer, a noise-free simulator. This indicates that the circuits we constructed are correctly preforming Shor's algorithm and Grover's search algorithm, and would give the correct results on a noise-free quantum computer.

Circuits run on IBM's quantum computers and other publicly available quantum computers are highly prone to errors as a result of noise[11]. We have shown that without specific measures to prevent these errors such as quantum error correction, the probability of a run with no errors from noise is low, even for low-qubit circuits such as these.

A current challenge for the field quantum computing is reducing this gap between simulated results and actual results of running quantum circuits. This could be achieved with advances in the construction of quantum computers which are less prone to noise, or with circuits which try to account for this noise, but require additional qubits.

## 4 Appendix

### 4.1 Code used

The code for simulating Shor's algorithm and Grover's search algorithm is available at https://github.com/AislingHeanue/Quantum-Computing-Circuits

## References

[1] Zygelman, Bernard (2018) A first Introduction to Quantum Computing and Information. Springer International Publishing.

[2] Cooley, James W.; Lewis, Peter A. W.; Welch,Peter D. (1967), Historical notes on the fast Fourier transform. IEEE Transactions on Audio and Electroacoustics. **15**(2), 76–79.

[3] Zimmermann, Paul (2016) Factorisation of RSA-220 with CADO-NFS. Cado-nfs-discuss, https://lists.gforge.inria.fr/pipermail/cado-nfs-discuss/2016-May/000626.html

[4] Shor, Peter W. (1999) Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM review **41**(2), 303-332.

[5] MD Sjaid Anis et al. (2021) Qiskit: An Open-source Framework for Quantum Computing. https://qiskit.org

[6] Vedral, V.; Barenco, A.; Ekert, A. (1996) Quantum networks for elementary arithmetic operations. Physical Review A, **54**(1), 147.

[7] Draper, Thomas G. (2000), Addition on a quantum computer. arXiv preprint quant-ph/0008033

[8] Shor, P. W. (1995). Scheme for reducing decoherence in quantum computer memory. Physical review A, **52**(4), R2493.

[9] Grover, L.K. (1996) A fast quantum mechanical algorithm for database search. In Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, 212-219.

[10] Figgatt, C., Maslov, D., Landsman, K.A. et al. (2017) Complete 3-Qubit Grover search on a programmable quantum computer. Nat Commun **8**, 1918. https://doi.org/10.1038/s41467-017-01904-7

[11] Johnstun, S.; Van Huele, J. F. (2021). Understanding and compensating for noise on IBM quantum computers. American Journal of Physics, 89(10), 935-942.