Quantum Algorithms: Deutsch-Jozsa Algorithm, Grover's Search Algorithm and Shor's Algorithm

Monit Sharma
Indian Institute of Science Education And Research Mohali
(Dated: July 10, 2021)

The revelation that quantum computers disrupt the extended Church-Turing thesis [1]—by determining specific computational tasks with exponentially fewer steps than the best classical algorithm for the same task—shook up the grounds of computer science and opened the opportunity of an entirely new way of solving computational problems quickly. The realistic potential of quantum computers was shown soon after that when Peter Shor created quantum algorithms for factoring large numbers and computing discrete logarithms that were exponentially faster than any developed for a classical computer [2]. These quantum algorithms created severe interest in the security society since the classical hardness of these two problems rest at the heart of the public key "cryptosystems" that defend the vast majority of society's digital data. In this paper we will look at various approaches towards Quantum Algorithms and their uses.

I. INTRODUCTION

In quantum computing, a quantum algorithm is an algorithm that runs on a realistic model of quantum computing. A classical (or non-quantum) system is a finite sequence of instructions or a step-by-step procedure for solving a problem, where each step or instruction can be performed on a classical computer. A quantum algorithm is a step-by-step process, where each step can be done on a quantum computer. Although all classical algorithms can be implemented on a quantum computer, the term quantum algorithm is often used for those algorithms that seem to be of quantum nature or have few of the essential features of quantum computing, such as quantum superposition or quantum entanglement.

Quantum algorithms are typically defined, in the usually used circuit model of quantum computation, by a quantum circuit that acts on some input qubits and stops with a measurement. A quantum circuit consists of simple quantum gates which act on at most a fixed number of qubits. The number of qubits has to be fixed because a changing number of qubits implies non-unitary evolution. Quantum algorithms may also be stated in other models of quantum computation, such as the Hamiltonian oracle model.[3]

Quantum algorithms can be characterised by the main techniques used by the algorithm. Some commonly used techniques in quantum algorithms include **phase kick-back, phase estimation, the quantum Fourier transform, quantum walks, amplitude amplification and topological quantum field theory**. Quantum algorithms may also be grouped by the type of problem solved.

II. ALGORITHMS BASED ON QUANTUM FOURIER TRANSFORM

A. Quantum Teleportation: Deutsch Algorithm

Quantum teleportation is a way for transferring quantum information from a sender at one location to a receiver some distance away. Quantum teleportation only transfers quantum information. Furthermore, the sender may not know the recipient's location and does not know which particular quantum state will be transferred. Experimental determinations [4] of quantum teleportation have been made in information content - including photons, atoms, electrons, and superconducting circuits - as well as distance with 1,400 km (870 mi) being the longest distance of successful teleportation.

In 1985, David Deutsch proposed a straightforward algorithm that, nevertheless, hints at the capabilities of quantum computing. The problem it solves is only of theoretical relevance and was later generalized in joint work with Jozsa. We are given a circuit (an oracle) that implements a one-bit boolean function, and we are asked to determine whether the function is constant (returns the same value for all inputs) or balanced(returns 1 on one input and 0 on the other) Alternatively, we can think of the oracle as indexing a bit string of length two, and we are asked to compute the XOR of the bits of the string. In the classical case, we would need to consult the oracle twice to compute both function values. In the quantum case, we can make just one oracle call, but in superposition.

An oracle is treated as a black box, a circuit whose interior we cannot know. This circuit computes, in a reversible way, a specific function f. For the computation to be reversible, it uses as many inputs and "writes the result" with an XOR.

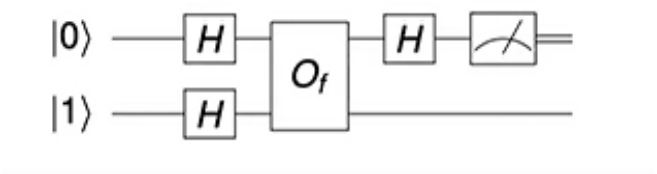

FIG. 1. Deutsch Circuit

1. *Analysis*

Let us analyze the algorithm to determine that it works correctly. Rather than taking the four cases separately, let us try to be more sophisticated and deal with the four cases all at the same time. The initial state is $|0\rangle, |1\rangle$, and so the state after the first two Hadamard transforms is

$$\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right)\left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right)$$

We can partially expand this state as

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

Performing the O_f operation, transforms this state to

$$\frac{1}{2}|0\rangle(|0\oplus f(0)\rangle - |1\oplus f(0)\rangle) + \frac{1}{2}|1\rangle(|0\oplus f(1)\rangle - |1\oplus f(1)\rangle)$$

$$= \left(\frac{1}{\sqrt{2}}(-1)^{f(0)}|0\rangle + \frac{1}{\sqrt{2}}(-1)^{f(1)}|1\rangle\right)\left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right)$$

The O_f transformation has apparently not changed the state of the second qubit; it has remained in the state

There has been an important effect, however, which is that the factors $-1^{f(0)}$ and $-1^{f(1)}$ have appeared in the state of the first qubit. This phenomenon is sometimes known as **phase kick-back**. It is a commonly used trick in quantum algorithms.

The second qubit is not important to us, We will deal with the first qubit only

$$\left(\frac{1}{\sqrt{2}}(-1)^{f(0)}|0\rangle + \frac{1}{\sqrt{2}}(-1)^{f(1)}|1\rangle\right)$$

which we can rewrite as

$$(-1)^{f(0)}\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}(-1)^{f(0)\oplus f(1)}|1\rangle\right)$$

Then, the final Hadamard transform take this state to

$$(-1)^{f(0)}|f(0)\oplus f(1)\rangle$$

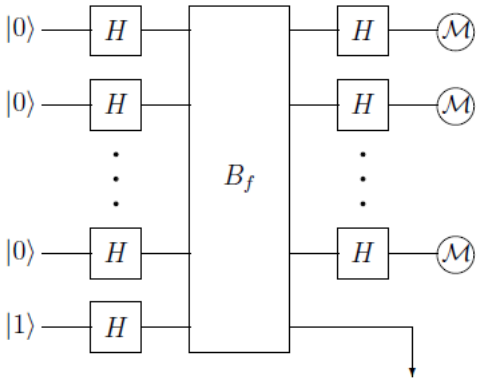

FIG. 2. Circuit Diagram for Deutsch-Jozsa

The measurement of this, therefore results in the value $(f(0) \oplus f(1))$. This value is 0 if f is constant and 1 if f is balanced.

B. A Simple Search Algorithm: The Deutsch-Jozsa Algorithm

It is an generalization of Deutsch's Algorithm called the Deutsch-Jozsa Algorithm. Although the computational problem being solved is still artificial and perhaps not particularly impressive.

This time, we assume that we are given a function $f : \{0,1\}^n \to \{0,1\}$, where n is some arbitrary positive integer, and we are promised that any of the two possibilities mentioned below holds:

- f is constant; $f(x) = 0$ for all $x \in \{0,1\}^n$ or $f(x) = 1$ for all $x \in \{0,1\}^n$
- f is balanced; This means that the number of inputs $x \in \{0,1\}^n$ for which it takes values 0 and 1 are the same:

As for the previous two algorithms, we assume that access to the function f is restricted to queries to a device corresponding to the transform B_f defined similarly to before:

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

for all $x \in \{0,1\}^n$ and $b \in \{0,1\}$.

It turns out that classically this problem is pretty easy given a small number of queries if we allow randomness and accept that there may be a small probability of error. Specifically we can randomly choose k inputs $x_1, ..., x_k \in \{0,1\}^n$, evaluate $f(x_i)$ for $i = 1, ..., k$ and answer constant if $f(x_1) = ... = f(x_k)$ and balanced otherwise. If the function was constant, this method will be correct every time, and if the function is balanced , this algorithm will be wrong with probability $2^-k - 1$.

In Quantum case, 1 query will be sufficient to determine with certainty whether the function is constant or balanced, and here is the algorithm, which is called **Deutsch-Jozsa Algorithm**

There are n bits resulting from the measurements. If all n measurement results are 0, we conclude that the function was constant. Otherwise, if at least one of the measurement outcomes is 1, we conclude that the function was balanced.

1. Analysis

We create the state $|0...0\rangle|1\rangle$ and after using Hadamard gate to create the superposition

$$\sum_{x\in\{0,1\}^n} \frac{1}{\sqrt{2^{n+1}}}|x\rangle(|0\rangle - |1\rangle)$$

We apply the oracle, getting

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

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

We apply again Hadamard gates to the n first qubits and we obtain. Finally we measure the n first qubits. If the function is constant we will obtain $|0\rangle$. Otherwise we will obtain $|1\rangle$

$$\sum_{y\in\{0,1\}^n}\sum_{x\in\{0,1\}^n} \frac{(-1)^{f(x)+x.y}}{2^n\sqrt{2}}|y\rangle(|0\rangle - |1\rangle)$$

For the Correctness of the Algorithm
The probability of measuring $|0\rangle$ is exactly

$$\left(\sum_{x\in\{0,1\}^n} \frac{(-1)^{f(x)+x.0}}{2^n}\right)^2 = \left(\sum_{x\in\{0,1\}^n} \frac{(-1)^{f(x)}}{2^n}\right)^2$$

- If f is constant , the sum is 1
- If f is balanced, the sum is 0

Check the Code for Deutsch-Jozsa Algorithm on Github.

III. ARITHMETIC AND NUMBER-THEORETIC PROBLEMS: REVERSIBLE COMPUTATION

A. Computational Number Theory

It is the branch of number theory concerned with finding and implementing efficient computer algorithms to solve various number theory problems. We will begin with basic definitions:

1. *Integer Addition*

Input: x, $y \in \mathbb{Z}$
Ouptut: $x + y$
The Boolean circuit complexity or the bit complexity of Integer addition for adding two n bit integer is $O(n)$

2. *Integer Multiplication*

Input: x, $y \in \mathbb{Z}$
Ouptut: xy
The Bit complexity of multyplying two n bit integer is $O(n^2)$

3. *Modular Exponentiation*

Input: A positive integer N, an integer $x \in \{0, ..., N-1\}$, and any integer k
Ouptut: $x^k (\text{mod } N)$
Using the Repeated Squaring method, the bit complexity is $O((lg\ k)(lg\ N)^2)$

4. *Primality Testing*

Input: A positive integer N
Ouptut: Prime if N is prime number, and Not Prime otherwise
Randomized algorithms having bit complexity $O((lg\ N)^3)$ that allow a probability 1/4, say, of making an error have been known for many years. In a breakthrough a few years ago, Agrawal, Kayal and Saxena [5] gave a deterministic prime testing algorithm that gives a $O((lg\ N)^6)$ deterministic algorithm for testing primality.

5. *Integer Factoring*

Input: A positive integer N
Ouptut: a prime factorization, $N = p_1^{a_1}....p_k^{a_k}$ of N

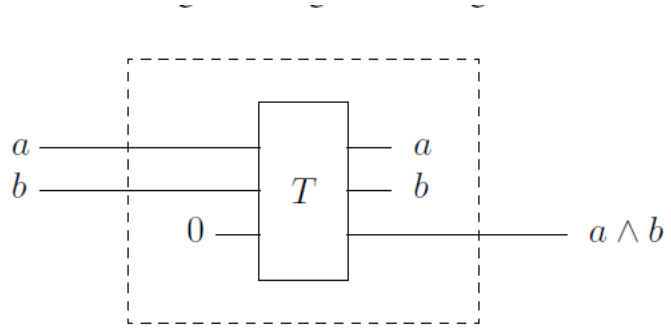

FIG. 3. AND gate using Toffoli

No classical algorithm is known to give a polynomial bit complexity for factoring- the best known algorithm asymptotically gives a bit complexity of

$$2^{O(log\ N)^{1/3}(log\ log\ N)^{2/3}}$$

B. Reversible Computation

To be considered efficient, one usually requires that the number of gates is polynomially related to the number of input qubits.Whereas there are a finite number of classical one and two bit operations, and it is well known that AND and NOT operations are sufficient to generate any such operation, there are infinitely many one and two qubit unitary operations.

So the Question here gets: **Whether an arbitrary efficient classical algorithm can be converted to an efficient quantum algorithm?**
Usually when we are talking about quantum gates and circuits that always map classical states to classical states, we refer to them as reversible gates or circuits. In other words, a reversible gate is a special type of quantum gate that maps classical states to classical states, or equivalently gives rise to a permutation matrix.

We can replace all the gates of the classical circuit with reversible gates.

- AND gate:
  We can simulate an AND gate using a Toffoli gate.
- NOT gate:
  These gates are already unitary,so we don't have to do anything.
- OR gate:
  These can be replaced by an AND gate and three NOT gates using DeMorgan's Law

Since NOT gates and Toffoli gates are their own inverses, we can simply take a "mirror image" of R to get R^{-1} which simply undoes the computation of R.

So , in summary if we have a classical Boolean circuit C for computing some function

$$f : \{0,1\}^n \to \{0,1\}^m$$

then the reversible computation can transform description of this circuit to a quantum circuit S_C composed only of Toffoli gates, NOT gates and controlled-NOT gates that satisfies

$$S_C|x\rangle|y\rangle|0^l\rangle = |x\rangle|y \oplus f(x)\rangle|0^l\rangle$$

C. Phase Estimation

Quantum phase estimation is one of the necessary subroutines in quantum computation. It serves as a central building block for many quantum algorithms. The Problem is the following:

Suppose that we have some Quantum Circuit Q acting on n qubits. Associated with this Quantum Circuit is a $2^n \times 2^n$ unitary matrix U. If n is reasonably large such as $n = 1,000$, it would be impossible to write down an explicit description of U because it is too large. Now, Since U is unitary, we know that it has a *complete, orthonormal collection of eigenvectors*

$$|\psi_1\rangle, ..., |\psi_N\rangle$$

where $N = 2^n$ and the associated eigenvalues having the form

$$e^{2\pi i\theta_1}, ..., e^{2\pi i\theta_N}$$

where $\theta_1, ...\theta_N \in [0, 1)$. This means that

$$U|\psi_j\rangle = e^{2\pi i\theta_j}|\psi_j\rangle$$

where the states are orthonormal.

The reason why each of the eigenvalues has the form $e^{2\pi i\theta}$, which is equivalent to saying that these eigenvalues are on the complex unit circle, is that U is unitary and therefore preserves Euclidean length.

1. *The Phase Estimation Procedure*

U is a unitary transformation acting on n-qubits, and suppose m is any positive integer. Then we let $\Lambda_m(U)$ denote the unique unitary transformation on $m+n$ qubits that satisfies:

$$\Lambda_m(U)|k\rangle|\phi\rangle = |k\rangle(U^k|\phi\rangle)$$

for all choices of $k \in \{0, ..., 2^m - 1\}$ and an arbitrary n-qubit vector $|\phi\rangle$.

In words, the first m qubits specify the number of times that U is to be applied to the remaining n qubits.

If someone gives us a quantum circuit Q that performs the unitary operation U, there is no guarantee that we will be able to build a quantum circuit that efficiently implements $\Lambda_m(U)$ for our choice of m.

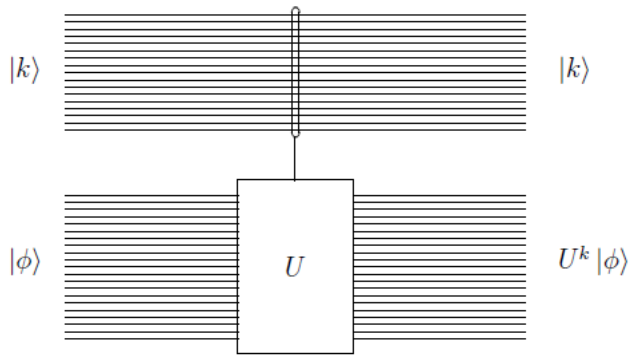

FIG. 4. Phase Estimation Circuit

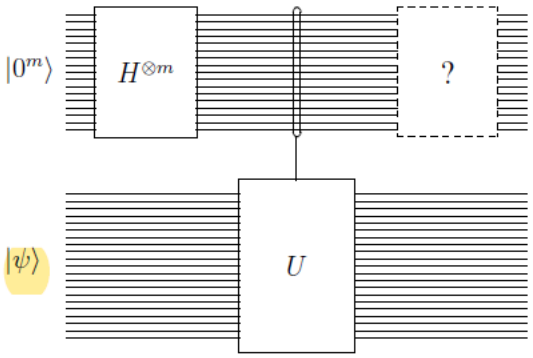

FIG. 5. Phase Estimation Circuit with Missing Gate

The initial state is $|0^m\rangle|\psi\rangle$ and after the Hadamard transforms are performed the state becomes

$$\frac{1}{2^{m/2}}\sum_{k=0}^{2^m-1}|k\rangle|\psi\rangle$$

Then the $\Lambda_m(U)$ transformation is performed, which performs that state to

$$\frac{1}{2^{m/2}}\sum_{k=0}^{2^m-1}|k\rangle|U^k\psi\rangle$$

Now, we use the fact $|\psi\rangle$ is an eigen vector of U.

$$\frac{1}{2^{m/2}}\sum_{k=0}^{2^m-1}|k\rangle|e^{2\pi ik\theta}\psi\rangle$$

$$=\frac{1}{2^{m/2}}\sum_{k=0}^{2^m-1}e^{2\pi ik\theta}|k\rangle|\psi\rangle$$

So, if we discard the last n qubits , we are left with the state

$$\frac{1}{2^{m/2}}\sum_{k=0}^{2^m-1}e^{2\pi ik\theta}|k\rangle$$

A Simple Case: $\theta = j/2^m$
Suppose for the moment that θ happens to have a special form

$$\theta=\frac{j}{2^m}$$

for some integer $j \in \{0,....,2^m-1\}$. We cannot assume the form of θ but it is helpful to consider this case first. Then the state can be written as

$$\frac{1}{2^{m/2}}\sum_{k=0}^{2^m-1}\omega^{jk}|k\rangle$$

for $\omega = e^{2\pi i/2^m}$
Now. let

$$|\phi\rangle=\frac{1}{2^{m/2}}\sum_{k=0}^{2^m-1}\omega^{jk}|k\rangle$$

for each $j \in \{0,...,2^m-1\}$. We know that the state of the first m qubits of our circuit is one of the states $|\phi_j\rangle : j = 0,....,2^m-1$ and the goal is to determine which one. Because once we know j we know θ as well.
We have

$$\langle\phi_j|\phi_{\bar{j}}\rangle=\frac{1}{2^m}\sum_{k=0}^{2^m-1}\left(\omega^{\bar{j}-j}\right)^k$$

Using the formula

$$\sum_{k=0}^{n-1}x^k=\frac{x^n-1}{x-1}$$

for $x \neq 1$, along with the observation that $\omega^{2ml} = 1$ for any integer l, we get the orthonormal states.
There is therefore a unitary transformation F that satisfies

$$F|j\rangle=|\phi_j\rangle$$

for $j = 0,....,2^m-1$. We can describe this matrix explicitly by allowing the vectors $|\phi_j\rangle$ to determine columns of F

$$F=\frac{1}{\sqrt{2^m}}\begin{pmatrix}1 & 1 & . & . & 1\\ 1 & \omega & \omega^2 & . & \omega^{2^m-1}\\ 1 & \omega^2 & \omega^4 & . & \omega^{2(2^m-1)}\\ . & . & . & . & .\\ . & . & . & . & .\\ 1 & \omega^{2^m-1} & \omega^{2(2^m-1)} & . & \omega^{(2^m-1)^2}\end{pmatrix}$$

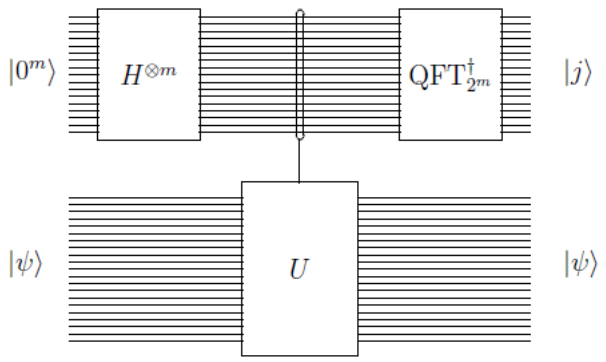

FIG. 6. Complete Phase Estimation Circuit

This matrix defines a linear transformation and is of much importance to us. It is the *discrete Fourier transform* and often time is denotes as QFT_{2^m}

Plugging the inverse of this transformation, in the circuit

Thus, measuring the first m qubits and dividing by 2^m tells us precisely the value θ.

Check the code for Quantum Phase Estimation Algorithm on Github

D. Shor's Algorithm

Any integer number has a unique decomposition into a product of primes, but finding the prime factors is believed to be a complex problem. The security of our online transaction rests on the assumption that factoring integers with a thousand or more digits is practically impossible. Shor's algorithm is the most dramatic example of how the paradigm of Quantum Computing changed our perception of which problem should be considered tractable.

Shor's algorithm is, probably the most famous quantum algorithm. It finds a factor of a n-bit integer in time $O(n^2(log\ \ n)(log\ \ log\ n))$. The best classical algorithm that we know of for the same task needs time $O(e^{cn^{\frac{1}{3}}(log\ n)^{\frac{2}{3}}})$

1. *Complexity of Factoring*

Suppose our task is to factor an integer N with d decimal digits. The brute force algorithm goes through all primes p up to $\sqrt{N}$ and checks whether p divides N. In the worst case, this would take time roughly $\sqrt{N}$, which is exponential in the number of digits d. A more efficient algorithm, known as the quadratic sieve, attempts to construct integers a, b such that $a^2 - b^2$ is a multiple of N. Once such a, b are found, one checks whether a+b have common factors with N. The quadratic sieve method has asymptotic runtime exponential in $\sqrt{d}$. The

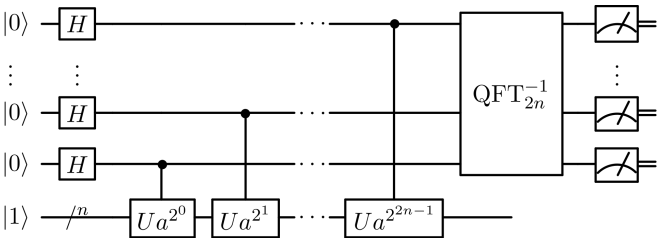

FIG. 7. Circuit For Shor's Algorithm

most efficient classical factoring algorithm, known as general number field sieve, achieves an asymptotic runtime exponential in $d^{1/3}$

The exponential runtime scaling limits the applicability of the classical factoring algorithms to numbers with a few hundred digits; the world record is $d = 232$ (which took roughly 2,000 CPU years!). In contrast, Shor's factoring algorithm has runtime polynomial in d. The version of the algorithm requires roughly $10d$ qubits and has runtime roughly $d^{1/3}$.

2. *Period Finding*

It has been known to mathematicians since the 1970s that factoring becomes easy if one can solve another hard problem: find a period of the modular exponential function. The period-finding problem is defined as follows. Given integers N and a, find the smallest positive integer r such that $a^r - 1$ is a multiple of N. The number r is called the period of a modulo$N$.

3. *Algorithm*

The Steps of Shor's Algorithm

1. Given N, check that N is not a prime or power of a prime. If it is, stop.
2. Choose $1 < a < N$ at random.
3. If $b = gcd(a, N) > 1$ output b and stop
4. Find the order of $a\ mod\ N$, that is, $r > 0$ such that $a^r = 1\ mod\ N$
5. If r is odd, go to 2
6. Compute

$$x = a^{\frac{r}{2}} + 1\ mod\ Ny = a^{\frac{r}{2}} - 1\ mod\ N$$

7. If $x = 0$, go to 2. If $y = 0$, take $r = \frac{r}{2}$ and go to 5.
8. Compute $p = gcd(x, N)$ and $q = gcd(y, N)$. At lease one of them will be a non-trivial factor of N.

For the Implementation of Shor's Algorithm every step but number 4 are carried out on a classical computer (efficient algorithms exists) and for step 4, there exists a quantum circuit with a number of gates that is polynomial of n (the number of bits N)

4. *Preparing a periodic sequence*

The first part of the circuit computes

$$\frac{1}{\sqrt{2^m}} \sum_{x=0}^{2^m-1} |x\rangle |a^x mod\ N\rangle$$

When we measure the bottom qubits, we obtain

$$\frac{1}{\sqrt{|C|}} \sum_{x \in C} |x\rangle |c\rangle$$

where c is some value in $\{0, ..., N-1\}$ and $C = \{x : a^x\ mod\ N = c\}$

For example, if $a = 2, N = 5, m = 4$ we would have

$$\frac{1}{4}(|0\rangle|1\rangle + |1\rangle|2\rangle + |2\rangle|4\rangle + |3\rangle|3\rangle + |4\rangle|1\rangle + ...|15\rangle|3\rangle)$$

and, when we measure we could obtain , for instance

$$\frac{1}{2}(|1\rangle|2\rangle + |5\rangle|2\rangle + |9\rangle|2\rangle + |13\rangle|2\rangle)$$

Notice that the values of the first register are exactly 4 units apart and that $2^4 = 1\ mod\ 5$ In general, we will obtain values that are r units apart, where $a^r = 1\ mod\ N$

5. *Measuring the Period*

For the Measurement of the period r we use the (inverse) of the Quantum Fourier Transform (QFT) Two properties of the QFT are central here:

- Shift-invariance (up to an unobservable phase)
- QFT transform sequences with period r into sequences with a period $\frac{M}{r}$(where $M = 2^m$)

After the use of the inverse QFT, we can measure a value of the form $\frac{Mc}{r}$ with high probability and, from it, obtain r

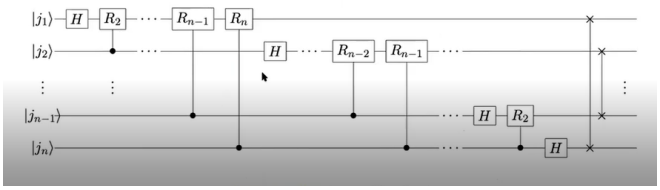

FIG. 8. Quantum Fourier Transform

6. *Quantum Fourier Transform*

The QFT of order m is the unitary transformation defined by

$$QFT|j\rangle = \frac{1}{\sqrt{2^m}} \sum_{k=0}^{2^m-1} e^{\frac{2\pi ijk}{2^m}} |k\rangle$$

The circuit in the figure implements the QFT. The R_k gates in the circuit are what called $R_Z(\frac{2\pi}{2^k})$ The number of gates is quadratic in m, an exponential speedup over the classical case (FFT) For Shor, m can be chosen to be about $2n$

7. *Using the QFT for phase estimation*

Suppose we are given a unitary operation U and one of its eigenvectors $|\psi\rangle$ We know that there exists $\theta \in [0, 1)$ such that $U|\psi\rangle = e^{2\pi i\theta}$ We can estimate θ ,With the first part, we will obtain $\frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{2\pi i\theta k}|k\rangle$, by using the inverse QFT we can measure $j \approx 2^n\theta$

8. *Shor's algorithm as a particular case of Quantum Phase Estimation*

Clearly, the circuit used in Shor's algorithm is a case of Quantum Phase estimation It can be shown that the unitary operation of modular multiplication by a has eigen values

$e^{2\pi i\frac{k}{r}}$, $k = 0, ..., r-1$

where r is the period of a

It is not easy to prepare one of the eigen vectors $|\psi_k\rangle$ of the unitary operation But we use the fact that

$|1\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} |\psi_k\rangle$

We will then measure a value close to $2^m k/r$ for some k

Check the code for Shor's Algorithm on Github

IV. ALGORITHMS BASED ON AMPLITUDE AMPLIFICATION

Amplitude amplification is a technique that allows the amplification of a chosen subspace of a quantum state.

Applications of amplitude amplification usually lead to quadratic speedups over the corresponding classical algorithms. It can be considered to be a generalization of *Grover's algorithm.*

A. Grover's Algorithm

Grover's algorithm also acknowledged as the quantum search algorithm, relates to a quantum algorithm for unstructured search that finds with high probability the unique input to a black box function that produces a particular output value, using just $O(\sqrt{N})$ evaluations of the function, where N is the size of the function's domain. Lov Grover devised it in 1996 [6]

A similar problem in classical computation cannot be solved in less than $O(N)$ evaluations (because, on average, one has to check half of the domain to get a 50% chance of finding the actual input). Since researchers commonly assume that NP-complete problems are complex because their search spaces have essentially no structure, the optimality of Grover's algorithm for unstructured search suggests (but does not prove) that quantum computers cannot solve NP-complete problems in polynomial time.

Unlike other quantum algorithms, which may present exponential speedup over their classical counterparts, Grover's algorithm provides only a quadratic speedup. Nevertheless, the quadratic speedup is substantial when N is significant, and Grover's algorithm can be applied to speed up broad classes of algorithms. Grover's algorithm could brute-force a 128-bit symmetric cryptographic key in roughly 264 iterations or a 256-bit key in roughly 2128 iterations. As a result, it is sometimes suggested that symmetric key lengths be doubled to protect against future quantum attacks.

1. *Statement*

Grover's Algorithm is used to solve search problems Imagine we have an unsorted list of N elements. One of them verifies a certain condition and we want to find it. Any classical algorithm requires $O(N)$ queries to the list in the worst case. Grover's algorithm can find the element with $O(\sqrt{N})$ queries.

2. *The Oracle*

As in Deutsch-Jozsa's algorithm, we will use an oracle. This oracle computes the function $f : \{0,1\}^n \implies \{0,1\}$, with $N = 2^n$ The element we want to find is the one that verifies $f(x) = 1$

Grover's algorithm is based on the idea of *inversion about the mean.*

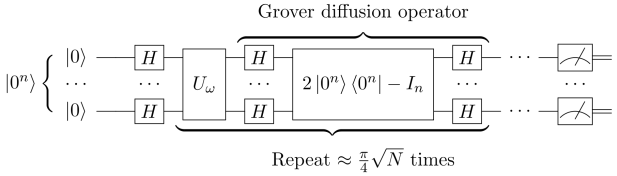

FIG. 9. Circuit for Grover's Algorithm

3. *Algorithm*

Grover's Algorithm performs $O(\sqrt{N})$ iterations , each one consisting in an oracle query and a call to Grover's Diffusion operator. The oracle "marks" those states that verify the condition. The diffusion operator "amplifies" the amplitudes of the marked state.

4. *Grover's algorithm as a Rotation*

Let us denote by $|x_1\rangle$ the marked element. Then, the initial state of the upper n qubits is

$$\sqrt{\frac{N-1}{N}}|x_0\rangle + \sqrt{\frac{1}{N}}|x_1\rangle$$

Where,

$$|x_0\rangle = \sum_{x\in\{0,1\}^n, x\neq x_1} \sqrt{\frac{1}{N-1}}|x\rangle$$

We can choose $\theta \in (0, \pi/2)$ such that

$$cos(\theta) = \sqrt{\frac{N-1}{N}} sin(\theta) = \sqrt{\frac{1}{N}}$$

Define D to be Grover's diffusion operator and $G = DO_f$ It can be shown that G acts on the 2-dimensional space spawned by $|x_0\rangle$ and $|x_1\rangle$ as a rotation of angle 2θ That is,

$$G|x_0\rangle = cos2\theta|x_0\rangle + sin2\theta|x_1\rangle G|x_1\rangle$$

$$= -sin2\theta|x_0\rangle + cos2\theta|x_1\rangle|x_0\rangle$$

$$= \sum_{x\in\{0,1\}^n, x\neq x_1} \sqrt{\frac{1}{N-1}}|x\rangle$$

Since the initial state is $cos\theta|x_0\rangle + sin\theta|x_1\rangle$, after m iterations we will have

$$cos(2m+1)\theta|x_0\rangle + sin(2m+1)\theta|x_1\rangle$$

In order to obtain $|x_1\rangle$ with high probability when we measure, we need

$$(2m+1)\theta \approx \frac{\pi}{2}$$

and this gives

$$m \approx \frac{\pi}{4\theta} - \frac{1}{2}$$

Since

$$sin\theta = \sqrt{\frac{1}{N}}$$

we will have

$$\theta \approx \sqrt{\frac{1}{N}}$$

and then we can choose

$$m = \left[\frac{\pi}{4}\sqrt{N}\right]$$

5. *The case with multiple marked elements*

If the number of marked elements is $k > 1$, a similar argument can be made by defining

$$|x_0\rangle = \sum_{f(x)=0} \sqrt{\frac{1}{N-k}}|x\rangle|x_1\rangle = \sum_{f(x)=1} \sqrt{\frac{1}{k}}|x\rangle$$

In this case

$$sin(\theta) = \sqrt{\frac{k}{N}}$$

and if $k << N$ we can choose

$$m = \left[\frac{\pi}{4}\sqrt{\frac{N}{k}}\right]$$

6. *The case with unknown number of marked elements*

If we do not know how many elements are marked, we can still use Grover's algorithm. We can use Grover's circuit combined with the Quantum Fourier Transform to estimate k. Or we can choose m at random. For instance: Uniformly from the set $\{0, ..., [\sqrt{N}+1\}$ With an incremental scheme, starting with an upper bound for m of $b = 1$ and increasing it exponentially up to $\sqrt{N}$ In all the cases, it can be shown that a marked element will be found with high probability with $O(\sqrt{N})$ queries to the oracle.

7. *Some comments on Grover's algorithm*

- When we measure, we will obtain x such that $f(x) = 1$ with probability depending on:
  1. The number m of iterations
  2. The fraction of values x that satisfy the condition
- If we perform too many iterations, we can overshoot and not find a marked element.
- On the other hand, if $k = \frac{N}{4}$ then one iteration will find a marked element with certainty.
- Grover's algorithm can be used to find minima of functions*(Durr-Hoyer's algorithm)*[7]
- It can be shown that no other quantum algorithm can obtain more than a quadratic speed-up over classical algorithm in the same setting.
- A generalization of Grover's algorithm called Amplitude Amplification can be used with states prepared by an arbitrary unitary A

Check the Code for Grover's Algorithm on Github

V. OUTLOOK

We read about many Quantum Algorithms, but many are still unknown, particularly those providing speedup over their classical counterparts.

One reason could be that we have prooves lower bounds on the computing power of quantum computation in the query complexity model.

For instance, the complexity delivered by Grover's algorithm cannot be increased by even one query while keeping the same success probability. To obtain an exponential speedup above classical computation in the query complexity model, there must be a promise on the input, The reason for Quantum Algorithms' success in cryptography is the hidden problem structures that quantum

computers can exploit in ways that classical computers cannot. Finding such hidden structure in other problems of practical interest remains a significant open problem.

As well as expanding new quantum algorithms, an essential objective for future research seems to be applying known quantum algorithms to new problem areas.

[1] Rabin, Michael O. (June 2012). Turing, Church, Gödel, Computability, Complexity and Randomization: A Personal View. Archived from the original on 2019-06-05. Retrieved 2012-07-14.

[2] Shor, P.W. (1994). "Algorithms for quantum computation: discrete logarithms and factoring". Proceedings 35th Annual Symposium on Foundations of Computer Science. IEEE Comput. Soc. Press: 124–134. doi:10.1109/sfcs.1994.365700. ISBN 0818665807.

[3] Farhi, E.; Goldstone, J.; Gutmann, S. (2007). "A Quantum Algorithm for the Hamiltonian NAND Tree". arXiv:quant-ph/0702144.

[4] Feng, Tianfeng , Xu, Qiao , Zhou, Linxiang , Maolin, Luo , Zhang, Wuhong. (2020). Teleporting an unknown quantum state to a photon with prior quantum information.

[5] Agrawal, Manindra; Kayal, Neeraj; Saxena, Nitin (2004). "PRIMES is in P" (PDF). Annals of Mathematics. 160 (2): 781–793. doi:10.4007/annals.2004.160.781. JSTOR 3597229.

[6] Grover, Lov K. (1996-07-01). "A fast quantum mechanical algorithm for database search". Proceedings of the twenty-eighth annual ACM symposium on Theory of Computing. STOC '96. Philadelphia, Pennsylvania, USA: Association for Computing Machinery: 212–219. doi:10.1145/237814.237866. ISBN 978-0-89791-785-8.

[7] arXiv:quant-ph/9607014v2 7 Jan 1999A Quantum algorithm for finding the minimum