Scientific Research Publishing

Journal of Computer and Communications, 2025, 13(2), 126-141
https://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

Numbering and Generating Quantum Algorithms

Mohamed A. El-Dosuky[1,2]

[1]Faculty of Computers and Information, Mansoura University, Mansoura, Egypt
[2]Arab East Colleges, Riyadh, Saudi Arabia
Email: maldosuky@arabeast.edu.sa

How to cite this paper: El-Dosuky, M.A. (2025) Numbering and Generating Quantum Algorithms. *Journal of Computer and Communications*, **13**, 126-141.
https://doi.org/10.4236/jcc.2025.132008

Received: January 22, 2025
Accepted: February 21, 2025
Published: February 24, 2025

Abstract

Quantum computing offers unprecedented computational power, enabling simultaneous computations beyond traditional computers. Quantum computers differ significantly from classical computers, necessitating a distinct approach to algorithm design, which involves taming quantum mechanical phenomena. This paper extends the numbering of computable programs to be applied in the quantum computing context. Numbering computable programs is a theoretical computer science concept that assigns unique numbers to individual programs or algorithms. Common methods include Gödel numbering which encodes programs as strings of symbols or characters, often used in formal systems and mathematical logic. Based on the proposed numbering approach, this paper presents a mechanism to explore the set of possible quantum algorithms. The proposed approach is able to construct useful circuits such as Quantum Key Distribution BB84 protocol, which enables sender and receiver to establish a secure cryptographic key via a quantum channel. The proposed approach facilitates the process of exploring and constructing quantum algorithms.

Keywords

Quantum Algorithms, Numbering Computable Programs, Quantum Key Distribution

1. Introduction

Quantum computing, utilizing quantum superposition, offers unprecedented computational power, enabling simultaneous computations beyond traditional computers [1]. This power opens up applications like drug discovery [2] and accelerated artificial intelligence [3].

Quantum software architecture is the design and structure of quantum computing systems, encompassing the organization of components, information flow, and framework for developing and deploying applications [4]. Key elements include quantum hardware abstraction [5], a quantum programming language [6], quantum algorithms [7], quantum error correction [8], optimizers [9], quantum simulators [10], and cloud-based quantum computing services [11]. Quantum hardware abstraction provides a unified interface for interacting with quantum devices, while quantum programming languages provide necessary abstractions and tools. Quantum algorithms may also be included in libraries or modules, enabling developers to leverage existing quantum algorithms for various applications. Integration between classical and quantum computing is crucial for realizing hybrid quantum-classical algorithms [12].

Quantum programming languages and frameworks are specialized tools for writing quantum algorithms and working with quantum computers [6]. Popular examples include Qiskit, Cirq, Quil, ProjectQ, and Q#. Qiskit is an open-source framework developed by IBM, providing a Python-based interface for working with quantum circuits, simulators, and devices. Cirq is a flexible framework by Google, while Quil is hardware-agnostic. ProjectQ supports multiple quantum backends. Q#, pronounced Q sharp, is a high-level language developed by Microsoft for writing quantum algorithms and interacting with quantum simulators and hardware. QASM is proposed to be an open quantum assembly language [13]. Quipper is a scalable quantum programming language [14]. Recent advancements in software framework development aim to compile quantum algorithms from high-level descriptions to physical gates for fault-tolerant quantum computers [15].

Tables 1-3 list the building blocks of quantum computing, known as quantum gates. Beside the name, each gate has a symbol and a matrix notation. Controlled-X is known as CNOT gate. SWAP gate can be decomposed of 3 CNOT gates.

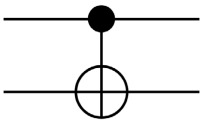
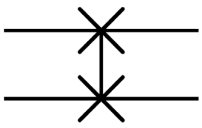
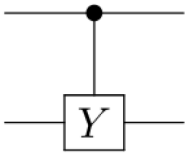
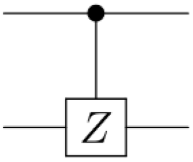
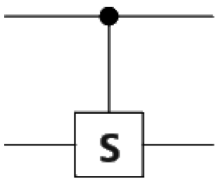
Table 1. Quantum gates operating on 1 qubit.

Gate Name	Gate Symbol	Matrix Representation
Pauli-X	X	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y	Y	$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli-Z	Z	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard	H	$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Rotation-X	$R_x(\theta)$	$\begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) \\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{bmatrix}$

Continued

Rotation-Y	$R_y(\theta)$	$\begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{bmatrix}$
Rotation-Z	$R_z(\theta)$	$\begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix}$
S	S	$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$
T	T	$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{bmatrix}$
R4	$R4$	$\begin{bmatrix} 0 & 1 \\ 1 & e^{i\frac{\pi}{8}} \end{bmatrix}$
Measure		Qubit to Bit

Table 2. Quantum gates operating on 2 qubits.

Gate Name	Gate Symbol	Matrix Representation
CNOT		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
SWAP		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Controlled-Y		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -i \\ 0 & 0 & i & 0 \end{bmatrix}$
Controlled-Z		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
Controlled-S		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & i \end{bmatrix}$

Continued

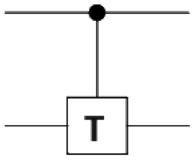
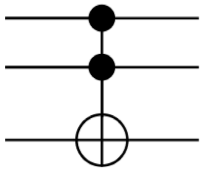
Controlled-T		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\frac{\pi}{4}} \end{bmatrix}$

Table 3. Quantum gates operating on 3 qubits.

Gate Name	Gate Symbol	Matrix Representation
Toffoli		$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$

The Gödel numbering is a widely used numbering scheme for computable functions, assigning a unique natural number to each function [16]. This method involves encoding the description of a function as a string and converting it into a number. The process involves creating a list of possible symbols, assigning a unique number to each symbol, and encoding the description of a function as a string.

This paper extends the numbering of programs as presented in [17] to be applied in the quantum computing context. Then and based on the proposed numbering approach, this paper presents a mechanism to explore the set of possible quantum algorithms in two ways, namely the forward way and the backward way. Finally, the paper validates the proposed numbering approach.

The contributions of this paper can be listed as follows:

- Introducing the concept of numbering of programs in the quantum computing context,
- Presenting a mechanism to explore the set of possible quantum algorithms in two ways,
- The forward way of exploring quantum algorithms generates code of quantum circuits,
- The backward way of exploring quantum algorithms construct quantum circuits from corresponding numbers, and
- The proposed approach is able to construct useful circuits such as Quantum Key Distribution BB84 protocol.

The rest of this paper is structured as follows. Section 2 presents two of the most reputable quantum algorithms to show a glimpse of the specific nature of quantum algorithms as compared to classical ones. In Section 3, the proposed numbering approach is provided. Section 4 contains the conclusion and recommendations for further work.

2. Quantum Algorithms

2.1. Grover's Quantum Search Algorithm

Grover's quantum search algorithm, proposed by Lov Grover in 1996 [18], is a quantum algorithm for searching specific items in unsorted databases. It uses quantum operations, such as the Hadamard transform and phase inversion, to amplify the target item's amplitude. The algorithm runtime is proportional to the square root of the database size, faster than classical algorithms. However, it has limitations, such as a quadratic speedup and requires a quantum computer.

First, we define the initial state of the quantum system as:

$$|\psi_0\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle \tag{1}$$

where N is the size of the database and $|x\rangle$ represents the state of the quantum system corresponding to the item x in the database.

Next, we define the quantum oracle that marks the target item as:

$$U_\omega |x\rangle = (-1)^{f(x)} |x\rangle \tag{2}$$

where $f(x)=1$ if x is the target item and $f(x)=0$ otherwise.

We then apply the Grover iteration, which consists of two steps:

Apply the quantum oracle:

$$|\psi_1\rangle = U_\omega |\psi_0\rangle \tag{3}$$

Apply the Grover diffusion operator:

$$|\psi_2\rangle = U_s |\psi_1\rangle \tag{4}$$

where the Grover diffusion operator is defined as:

$$U_s = 2|\psi_0\rangle\langle\psi_0| - I \tag{5}$$

and I is the identity operator.

The Grover iteration is repeated k times, where k is approximately equal to:

$$k \approx \frac{\pi}{4}\sqrt{N} \tag{6}$$

After k iterations, the quantum system is measured, and the target item is found with high probability.

2.1.1. Shor's Quantum Algorithm for Factoring Integers

Shor's quantum algorithm, proposed by Peter Shor in 1994 [19], efficiently factored large composite numbers into prime factors using quantum modular exponentiation. The algorithm uses a random number a between 1 and $N-1$, and a quantum circuit constructed using quantum Fourier transform and modular exponentiation. The runtime is $O\left((\log N)^3\right)$, faster than classical algorithms but requires a large number of qubits.

Here are some of the key equations involved in the algorithm:

The quantum Fourier transform is used to convert the period finding problem into a phase estimation problem. The quantum Fourier transform of a state $|x\rangle$

is defined as:

$$\mathcal{F}\left|x\right\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \mathrm{e}^{2\pi i x y/N} \left|y\right\rangle \tag{7}$$

where i is the imaginary unit and N is the size of the quantum register.

The quantum modular exponentiation is used to compute $a^x \bmod N$ efficiently using a quantum computer. It is defined as:

$$U_f \left|x\right\rangle \left|y\right\rangle = \left|x\right\rangle \left|y \oplus f\left(x\right)\right\rangle \tag{8}$$

where $f\left(x\right) = a^x \bmod N$ and $\oplus$ represents bitwise addition modulo 2.

The phase estimation algorithm is used to estimate the phase of the eigenvalue of the unitary operator U_f. It is based on the application of the quantum Fourier transform to a superposition of eigenstates of U_f:

$$\left|\psi\right\rangle = \frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} \left|u^j\right\rangle \tag{9}$$

where u is the eigenvector of U_f corresponding to the eigenvalue $\mathrm{e}^{2\pi i\theta}$.

The period r of the function $f\left(x\right) = a^x \bmod N$ can be obtained from the phase estimate using continued fractions:

$$\theta \approx \frac{x}{q} = \left[0; a_1, a_2, \cdots, a_k\right] \tag{10}$$

where q is a power of 2 and the coefficients a_i are obtained from the convergents of the continued fraction expansion of x/q.

Once the period r is obtained, N can be factored using classical methods. The factors are given by:

$$N = \gcd\left(a^{r/2} \pm 1, N\right) \times \gcd\left(a^{r/2} \mp 1, N\right) \tag{11}$$

where gcd denotes the greatest common divisor.

3. Proposed Numbering Approach

3.1. Numbering the Set of Quantum Gates

To prove that the set of quantum gates, denoted by Q, is effectively denumerable, we need to show that there exists a bijective mapping from the set of quantum gates to the set of natural numbers. In other words, we need to show that we can assign a unique natural number to each quantum gate. The set of quantum gates typically consist of a finite number of gates, such as H, X, Y, Z, CNOT, etc. Each gate can be uniquely identified by its name and the qubits it operates on. To establish a bijective mapping, we can assign a unique natural number to each possible combination of gate name and qubits.

For numbering the X gate:

$$\beta\left(X\left(n\right)\right) = 19\left(n-1\right) + 0 \tag{12}$$

For numbering the Y gate:

$$\beta\left(Y\left(n\right)\right)=19\left(n-1\right)+1 \tag{13}$$

For numbering the Z gate:

$$\beta\left(Z\left(n\right)\right)=19\left(n-1\right)+2 \tag{14}$$

For numbering the H gate:

$$\beta\left(H\left(n\right)\right)=19\left(n-1\right)+3 \tag{15}$$

For numbering the S gate:

$$\beta\left(S\left(n\right)\right)=19\left(n-1\right)+4 \tag{16}$$

For numbering the T gate:

$$\beta\left(T\left(n\right)\right)=19\left(n-1\right)+5 \tag{17}$$

For numbering the R4 gate:

$$\beta\left(R4\left(n\right)\right)=19\left(n-1\right)+6 \tag{18}$$

For numbering the Measure gate:

$$\beta\left(Measure\left(n\right)\right)=19\left(n-1\right)+7 \tag{19}$$

For numbering the $R_x\left(\theta\right)$ gate:

$$\beta\left(R_x\left(n,\theta\right)\right)=19\Pi\left(n-1,\theta\right)+8 \tag{20}$$

For numbering the $R_y\left(\theta\right)$ gate:

$$\beta\left(R_y\left(n,\theta\right)\right)=19\Pi\left(n-1,\theta\right)+9 \tag{21}$$

For numbering the $R_z\left(\theta\right)$ gate:

$$\beta\left(R_z\left(n,\theta\right)\right)=19\Pi\left(n-1,\theta\right)+10 \tag{22}$$

For numbering the CNOT gate:

$$\beta\left(\mathrm{CNOT}\left(m,n\right)\right)=19\Pi\left(m-1,n-1\right)+11 \tag{23}$$

For numbering the SWAP gate:

$$\beta\left(\mathrm{SWAP}\left(m,n\right)\right)=19\Pi\left(m-1,n-1\right)+12 \tag{24}$$

For numbering the Controlled-Y gate:

$$\beta\left(\mathrm{CY}\left(m,n\right)\right)=19\Pi\left(m-1,n-1\right)+13 \tag{25}$$

For numbering the Controlled-Z gate:

$$\beta\left(\mathrm{CZ}\left(m,n\right)\right)=19\Pi\left(m-1,n-1\right)+14 \tag{26}$$

For numbering the Controlled-S gate:

$$\beta\left(\mathrm{CS}\left(m,n\right)\right)=19\Pi\left(m-1,n-1\right)+15 \tag{27}$$

For numbering the Controlled-T gate:

$$\beta\left(\mathrm{CT}\left(m,n\right)\right)=19\Pi\left(m-1,n-1\right)+16 \tag{28}$$

For numbering the Toffoli gate:

$$\beta\left(\mathrm{Toffoli}\left(k,m,n\right)\right)=19\zeta\left(k,m,n\right)+17 \tag{29}$$

where

$$\Pi\left(m,n\right)=2^{m}\left(2n+1\right)-1 \tag{30}$$

and

$$\zeta\left(k,m,n\right)=\Pi\left(\Pi\left(k-1,m-1\right),n-1\right) \tag{31}$$

To find $\beta^{-1}\left(q\right)$, find u and r such that $q=19u+r$ with $0\le r<19$. This value of r indicates which kind of gate β^{-1} is.

$$\text{if } r=0, \text{ then } \beta^{-1}\left(q\right)=X\left(u+1\right) \tag{32}$$

$$\text{if } r=1, \text{ then } \beta^{-1}\left(q\right)=Y\left(u+1\right) \tag{33}$$

$$\vdots$$

$$\text{if } r=17, \text{ then } \beta^{-1}\left(q\right)=\mathrm{Toffoli}\left(k,m,n\right) \tag{34}$$

where

$$\left(k,m,n\right)=\zeta^{-1}\left(u\right) \tag{35}$$

and

$$\zeta^{-1}\left(q\right)=\left(\Pi_1\left(\Pi_1\left(q\right)\right)+1,\Pi_2\left(\Pi_1\left(q\right)\right)+1,\Pi_2\left(q\right)+1\right) \tag{36}$$

where Π_1 and Π_2 are computable functions defined as follows.

$$\Pi_1\left(q\right)=\left(q+1\right)_1 \tag{37}$$

and

$$\Pi_2\left(q\right)=\frac{1}{2}\left(\frac{q+1}{2^{\Pi_1(q)}}-1\right) \tag{38}$$

Note that at $r=18$ is kept unused for future usages.

3.2. Numbering the Set of Quantum Circuits

To prove that the set C of all quantum circuits is effectively denumerable, we need to show that there exists a bijective mapping from the set of quantum circuits to the set of natural numbers. In other words, we need to show that we can assign a unique natural number to each quantum circuit. A quantum circuit consists of a sequence of quantum gates. To establish a bijective mapping, we can use a standard encoding scheme, such as the Gödel numbering. If circuit C is a sequence of gates: $G_1,G_2,\cdots,G_s$ then

$$\gamma\left(C\right)=\tau\left(\beta\left(G_1\right),\beta\left(G_2\right),\cdots,\beta\left(G_s\right)\right) \tag{39}$$

where

$$\tau\left(v_1,\cdots,v_k\right)=2^{v_1}+2^{v_1+v_2+1}+\cdots+2^{v_1+v_2+\cdots+k-1}-1 \tag{40}$$

Calculating $\tau^{-1}(c)$, relies on the fact that every natural number can be expressed as a binary number. Given c, we can find a unique numbers k and $0 \le u_1 < u_2 < \cdots < u_k$ such that

$$c+1 = 2^{u_1} + 2^{u_2} + \cdots + 2^{u_k} \tag{41}$$

So, $\tau^{-1}(c) = (v_1, v_2, \cdots, v_k)$ where $v_1 = u_1$ and

$$v_{i+1} = u_{i+1} - u_i - 1 \tag{42}$$

where $1 \le i \le k$

For a circuit C, the number $\gamma(C)$ is called the code or the number of that circuit. We can say that C_n = the circuit with code number $n = \gamma^{-1}(n)$. We say that C_n is the nth circuit.

Note that if $m \ne n$, then C_m is different from C_n even that they may compute the same operation.

3.3. Exploring Quantum Algorithms: The Forward Way

Figure 1 shows the circuit of EPR creation. This circuit consists of Hadamard and CNOT gates. Let us calculate the number or the code of this circuit.

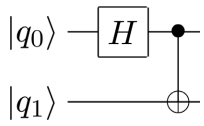


Figure 1. EPR creation.

It can be listed as:

```
H(1)
CNOT (1, 2)
```

For numbering the H gate:

$$\beta(H(n)) = 19(n-1)+3 = 3$$

For numbering the CNOT gate:

$$\beta(\text{CNOT}(m,n)) = 19\Pi(m-1, n-1)+11$$

At $m = 1$ and $n = 2$, let us first calculate the Π function

$$\Pi(0,1) = 2^0(2\times 1+1)-1 = 2$$

For numbering the CNOT gate:

$$\beta(\text{CNOT}(1,2)) = 19\times 2+11 = 49$$

Now, let us compute the code of that circuit.

$= \tau(2, 49)$

$= 2^2 + 2^{2+49+1} - 1$

$= 4 + 2^{52} - 1$

$= 4{,}503{,}599{,}627{,}370{,}499$

Figure 2 shows the circuit of SWAP operation. This circuit consists of three CNOT gates. Let us calculate the number or the code of this circuit.

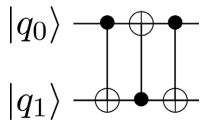


Figure 2. SWAP circuit.

It can be listed as:

```
CNOT (1, 2)
CNOT (2, 1)
CNOT (1, 2)
```

We previously calculated CNOT(1, 2) = 49. For numbering the CNOT(2, 1) gate

$$\beta\left(\mathrm{CNOT}\left(m,n\right)\right)=19\Pi\left(m-1,n-1\right)+11$$

At $m=2$ and $n=1$, let us first calculate the Π function

$$\Pi\left(1,0\right)=2^{1}\left(2\times 0+1\right)-1=1$$

For numbering the CNOT gate:

$$\beta\left(\mathrm{CNOT}\left(2,1\right)\right)=19\times 1+11=30$$

Now, let us compute the code of that circuit.

$= \tau(49, 30, 49)$

$= 2^{49}+2^{49+30+1}+2^{49+30+49+2}-1$

$= 2^{49}+2^{80}+2^{130}-1$

= 562,949,953,421,312

+ 1,208,925,819,614,629,174,706,176

+ 1,361,129,467,683,753,853,853,498,429,727,072,845,824

− 1

= 1,645,504,558,087,453,812,587,913,611,736,524,018,557,

890,457,442,949,424,440,410,111

Note that this code for numbering the SWAP gate is the equivalent to

$$\beta\left(\mathrm{SWAP}\left(1,2\right)\right)=19\Pi\left(0,1\right)+12=50$$

based on Π

$$\Pi\left(0,1\right)=2^{0}\left(2\times 1+1\right)-1=2$$

and $\tau\left(50\right)=2^{50}-1=1125899906842623$

Figure 3 shows the teleportation circuit. This circuit consists of Hadamard, CNOT, Measure, Controlled-Z, and Controlled- X gates. Let us calculate the number or the code of this circuit.

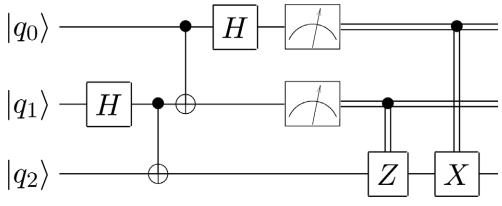


Figure 3. Teleportation circuit.

It can be listed as:

```
H(2)
CNOT (2, 3)
CNOT (1, 2)
H(1)
Measure (1)
Measure (2)
Controlled-Z (2, 3)
Controlled-X (1, 3)
```

Let us code each of the gates as follows.

$$\beta\left(\mathrm{H}\left(2\right)\right)=19\left(1\right)+3=22$$

$$\beta\left(\mathrm{CNOT}\left(2,3\right)\right)=19\Pi\left(1,2\right)+11=19\times 9+11=182$$

$$\beta\left(\mathrm{CNOT}\left(1,2\right)\right)=19\Pi\left(0,1\right)+11=19\times 2+11=49$$

$$\beta\left(\mathrm{H}\left(1\right)\right)=19\left(0\right)+3=3$$

$$\beta\left(\mathrm{Measure}\left(1\right)\right)=19\left(0\right)+7=7$$

$$\beta\left(\mathrm{Measure}\left(2\right)\right)=19\left(1\right)+7=26$$

$$\beta\left(\mathrm{Controlled\text{-}Z}\left(2,3\right)\right)=19\Pi\left(1,2\right)+14=19\times 9+14=185$$

$$\beta\left(\mathrm{Controlled\text{-}X}\left(1,3\right)\right)=19\Pi\left(0,2\right)+11=19\times 4+11=87$$

The number of the circuit can then be calculated.

$= \tau(22, 182, 49, 3, 7, 26, 185, 87)$

$= 2^{22}+2^{22+182+1}+2^{22+182+49+2}+2^{22+182+49+3+3}$

$+ 2^{22+182+49+3+7+4}+2^{22+182+49+3+7+26+5}$

$+ 2^{22+182+49+3+7+26+185+6}+2^{22+182+49+3+7+26+185+87+7}$

-1

$= 2^{22}+2^{205}+2^{255}+2^{259}+2^{267}+2^{294}+2^{480}+2^{568}-1$

= 4,194,304

+ 51,422,017,416,287,688,817,342,786,954,917,203,
280,710,495,801,049,370,729,644,032

+ 57,896,044,618,658,097,711,785,492,504,343,953,
926,634,992,332,820,282,019,728,792,003,956,564,819,
968

+ 926,336,713,898,529,563,388,567,880,069,503,262,
826,159,877,325,124,512,315,660,672,063,305,037,119,
488

+ 237,142,198,758,023,568,227,473,377,297,792,835,
283,496,928,595,231,875,152,809,132,048,206,089,502,
588,928

+ 31,828,687,130,226,345,097,944,463,881,396,533,
766,429,193,651,030,253,916,189,694,521,162,207,

808,802,136,034,115,584
+ 3,121,748,550,315,992,231,381,597,229,793,166,
305,748,598,142,664,971,150,859,156,959,625,371,
738,819,765,620,120,306,103,063,491,971,159,826,
931,121,406,622,895,447,975,679,288,285,306,290,
176
+ 966,134,380,754,314,586,173,837,972,732,996,836,
074,731,832,426,608,749,664,308,812,862,879,785,
572,390,106,134,048,441,645,480,644,490,615,904,
007,875,544,294,341,269,665,260,746,913,935,727,
168,366,770,187,174,245,203,705,856
− 1
= 966,134,380,754,314,586,173,837,975,854,745,386,
390,724,063,808,205,979,457,475,118,611,477,928,237,
361,257,025,034,088,639,205,159,924,866,743,949,573,
929,210,916,805,793,710,442,371,339,067,812,831,970,
988,587,388,382,478,335

Figure 4 shows the circuit of Quantum Fourier Transform (QFT). This circuit consists of Hadamard, Controlled-S, Controlled-T, and SWAP gates. Let us calculate the number or the code of this circuit.

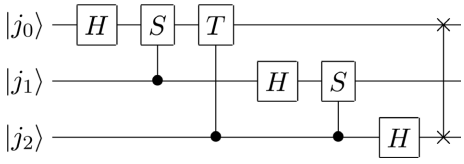


Figure 4. Quantum Fourier transform.

It can be listed as:

```
H(1)
Controlled-S (2, 1)
Controlled-T (3, 1)
H(2)
Controlled-S (3, 2)
H(3)
SWAP (1, 3)
```

Let us code each of the gates as follows.

$$\beta\left(\mathrm{H}(1)\right)=19(0)+3=3$$

$$\beta\left(\text{Controlled-S}(2,1)\right)=19\Pi(1,0)+15=19\times1+15=34$$

$$\beta\left(\text{Controlled-T}(3,1)\right)=19\Pi(2,0)+16=19\times3+16=73$$

$$\beta\left(\mathrm{H}(2)\right)=19(1)+3=22$$

$$\beta\left(\text{Controlled-S}(3,2)\right)=19\Pi(2,1)+15=19\times11+15=224$$

$$\beta(\mathrm{H}(3)) = 19(2) + 3 = 41$$

$$\beta(\mathrm{SWAP}(1,3)) = 19\Pi(0,2) + 12 = 19 \times 4 + 12 = 88$$

The number of the circuit can then be calculated.

$= \tau(3, 34, 73, 22, 224, 41, 88)$

$= 2^3 + 2^{3+34+1} + 2^{3+34+73+2} + 2^{3+34+73+22+3}$

$+ 2^{3+34+73+22+224+4} + 2^{3+34+73+22+224+41+5}$

$+ 2^{3+34+73+22+224+41+88+6}$

$- 1$

$= 2^3 + 2^{38} + 2^{112} + 2^{135} + 2^{360} + 2^{402} + 2^{491} - 1$

= 8 + 274,877,906,944

+ 5,192,296,858,534,827,628,530,496,329,220,096

+ 43,556,142,965,880,123,323,311,949,751,266,331,

066,368

+ 2,348,542,582,773,833,227,889,480,596,789,337,

027,375,682,548,908,319,870,707,290,971,532,209,

025,114,608,443,463,698,998,384,768,703,031,934,976

+ 10,328,999,512,347,634,358,623,676,688,012,047,497,

318,823,171,316,894,051,322,637,426,162,590,488,067,

364,778,518,581,413,120,551,325,743,612,687,890,989,

973,504

+ 6,393,341,031,047,152,089,869,511,126,616,404,594,173,

128,996,177,860,916,959,553,453,312,761,321,102,879,990,

006,386,899,074,031,556,935,325,554,936,640,763,689,877,

454,191,182,408,307,282,280,448

− 1

= 6,393,341,031,047,152,089,869,511,136,945,404,106,

523,111,897,384,311,438,199,490,431,228,373,829,447,

149,723,877,932,645,107,329,335,974,222,586,036,484,

943,430,874,337,072,758,146,938,842,382,343

3.4. Exploring Quantum Algorithms: The Backward Way

The backward way of exploring quantum algorithms is shown in **Figure 5**. The algorithm takes the initial and last values of numbering. It starts with setting empty set to Cs which is the list of valid quantum circuits. It loops until it reaches a stopping condition or the loop iterator reaches the last value of numbering. At each iteration, it calculates the valid quantum circuit that is associated with a number equals to the iterator. Validity of a quantum circuit means that it performs a desirable operation based on evaluation of an expert. The author played the role of expert for recognizing useful quantum circuits.

One of useful applications of quantum circuits is Quantum Key Distribution (QKD) BB84 protocol [20] which is proved to be secure [21]. Quantum Key

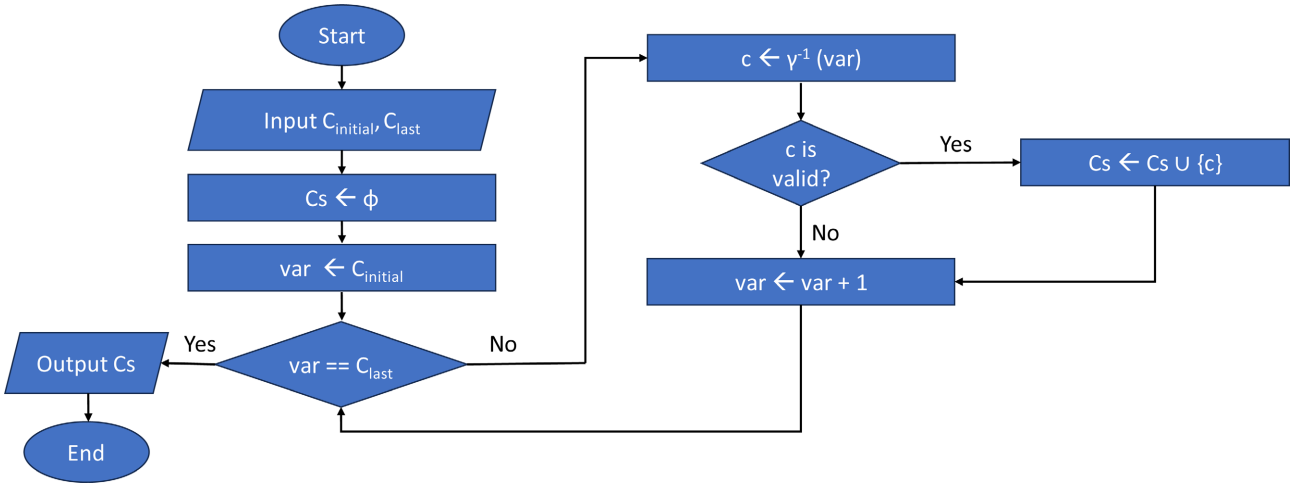


Figure 5. Exploring quantum algorithms.

Distribution (QKD) is a cryptographic protocol that uses quantum mechanics to establish a shared key between Alice and Bob. The BB84 protocol, named after its inventors Charles Bennett and Gilles Brassard, involves initialization, transmission, basis selection, measurement, public announcement, key sifting, error estimation, privacy amplification, and key generation [22]. A full software stack to integrate QKD in a cloud context was proposed in a recent publication [23]. Another recent paper presented a simulation of the QKD BB84 protocol using IBM's Qiskit quantum computing platform [24]. The simulation involves implementing the protocol with and without an eavesdropper Eve.

Table 4 shows the quantum part of the channel between Alice and Bob implementing BB84 Protocol on 5 Qubits. There is also a classical channel between them. The proposed approach can generate the circuits of BB84 protocol. But it depends on the ability of the implementation runner to recognize the circuits.

Table 4. BB84 Protocol on 5 Qubits.

Alice	Eve	Bob
X H		
H		H
	H	H
X	H	
H	H	

4. Conclusion and Future Work

This paper reports the construction of an effective numbering of quantum gates

and circuits. Then based on this numbering, quantum algorithms can be explored. There is a caveat in the implementation of the proposed numbering. As you may have noticed, the code or number of a quantum circuits grows very rapidly. Python's ability to handle large numbers has limitations, including memory consumption and slower computation times, making it impractical for performance and resource usage.

A possible future work can be seeking an efficient implementation of precise large numbers. Another future direction may be to extend the proposed numbering approach to other layers of quantum software architecture.

Conflicts of Interest

The author declares no conflicts of interest regarding the publication of this paper.

References

[1] Gyongyosi, L. and Imre, S. (2019) A Survey on Quantum Computing Technology. *Computer Science Review*, **31**, 51-71. https://doi.org/10.1016/j.cosrev.2018.11.002

[2] Cao, Y., Romero, J. and Aspuru-Guzik, A. (2018) Potential of Quantum Computing for Drug Discovery. *IBM Journal of Research and Development*, **62**, 6:1-6:20. https://doi.org/10.1147/jrd.2018.2888987

[3] Ayoade, O., Rivas, P. and Orduz, J. (2022) Artificial Intelligence Computing at the Quantum Level. *Data*, **7**, Article 28. https://doi.org/10.3390/data7030028

[4] Khan, A.A., Ahmad, A., Waseem, M., Liang, P., Fahmideh, M., Mikkonen, T., *et al.* (2023) Software Architecture for Quantum Computing Systems—A Systematic Review. *Journal of Systems and Software*, **201**, Article 111682. https://doi.org/10.1016/j.jss.2023.111682

[5] Khalid, M., Mujahid, U., Jafri, A., Choi, H. and Muhammad, N.U.I. (2021) An FPGA-Based Hardware Abstraction of Quantum Computing Systems. *Journal of Computational Electronics*, **20**, 2001-2018. https://doi.org/10.1007/s10825-021-01765-w

[6] Heim, B., Soeken, M., Marshall, S., Granade, C., Roetteler, M., Geller, A., *et al.* (2020) Quantum Programming Languages. *Nature Reviews Physics*, **2**, 709-722. https://doi.org/10.1038/s42254-020-00245-7

[7] Montanaro, A. (2016) Quantum Algorithms: An Overview. *npj Quantum Information*, **2**, Article No. 15023. https://doi.org/10.1038/npjqi.2015.23

[8] Roffe, J. (2019) Quantum Error Correction: An Introductory Guide. *Contemporary Physics*, **60**, 226-245. https://doi.org/10.1080/00107514.2019.1667078

[9] Quetschlich, N., Burgholzer, L. and Wille, R. (2023) Compiler Optimization for Quantum Computing Using Reinforcement Learning. 2023 60*th ACM/IEEE Design Automation Conference* (*DAC*), San Francisco, 9-13 July 2023, 1-6. https://doi.org/10.1109/dac56929.2023.10248002

[10] Altman, E., Brown, K.R., Carleo, G., Carr, L.D., Demler, E., Chin, C., *et al.* (2021) Quantum Simulators: Architectures and Opportunities. *PRX Quantum*, **2**, Article 017003. https://doi.org/10.1103/prxquantum.2.017003

[11] Soeparno, H. and Perbangsa, A.S. (2021) Cloud Quantum Computing Concept and Development: A Systematic Literature Review. *Procedia Computer Science*, **179**, 944-954. https://doi.org/10.1016/j.procs.2021.01.084

[12] Endo, S., Cai, Z., Benjamin, S.C. and Yuan, X. (2021) Hybrid Quantum-Classical

Algorithms and Quantum Error Mitigation. *Journal of the Physical Society of Japan*, **90**, Article 032001. https://doi.org/10.7566/jpsj.90.032001

[13] Cross, A., Bishop, L., Smolin, J. and Gambetta, J. (2017) Open Quantum Assembly Language.

[14] Green, A.S., Lumsdaine, P.L., Ross, N.J., Selinger, P. and Valiron, B. (2013) Quipper. *Proceedings of the* 34*th ACM SIGPLAN Conference on Programming Language Design and Implementation*, Seattle, 6-19 June 2013, 333-342. https://doi.org/10.1145/2491956.2462177

[15] Roetteler, M., Svore, K.M., Wecker, D. and Wiebe, N. (2017) Design Automation for Quantum Architectures. 2017 *Design*, *Automation & Test in Europe Conference & Exhibition* (*DATE*), Lausanne, 27-31 March 2017, 1312-1317. https://doi.org/10.23919/date.2017.7927196

[16] Gödel, K. (1931) Godel's Theorem in Focus. In: *On Formally Undecidable Propositions of Principia Mathematica and Related Systems*, Dover Publications, 17-47.

[17] Cutland, N. (1980) Computability. Cambridge University Press. https://doi.org/10.1017/cbo9781139171496

[18] Grover, L.K. (1996) A Fast Quantum Mechanical Algorithm for Database Search. *Proceedings of the* 28*th Annual ACM Symposium on Theory of Computing—STOC*'96, Philadelphia, 22-24 May 1996, 212-219. https://doi.org/10.1145/237814.237866

[19] Shor, P.W. (1994) Algorithms for Quantum Computation: Discrete Logarithms and Factoring. *Proceedings of the* 35*th Annual Symposium on Foundations of Computer Science*, Santa Fe, 20-22 November 1994, 124-134. https://doi.org/10.1109/sfcs.1994.365700

[20] Chong, S. and Hwang, T. (2010) Quantum Key Agreement Protocol Based on BB84. *Optics Communications*, **283**, 1192-1195. https://doi.org/10.1016/j.optcom.2009.11.007

[21] Shor, P.W. and Preskill, J. (2000) Simple Proof of Security of the BB84 Quantum Key Distribution Protocol. *Physical Review Letters*, **85**, 441-444. https://doi.org/10.1103/physrevlett.85.441

[22] Bennett, C.H. and Brassard, G. (1984) An Update on Quantum Cryptography. In: *Lecture Notes in Computer Science*, Springer, 475-480. https://doi.org/10.1007/3-540-39568-7_39

[23] Pedone, I., Atzeni, A., Canavese, D. and Lioy, A. (2021) Toward a Complete Software Stack to Integrate Quantum Key Distribution in a Cloud Environment. *IEEE Access*, **9**, 115270-115291. https://doi.org/10.1109/access.2021.3102313

[24] Saeed, M.H., Sattar, H., Durad, M.H. and Haider, Z. (2022) Implementation of QKD BB84 Protocol in Qiskit. 2022 19*th International Bhurban Conference on Applied Sciences and Technology* (*IBCAST*), Islamabad, 16-20 August 2022, 689-695. https://doi.org/10.1109/ibcast54850.2022.9990073