QIST Project

Hafiz Muhammad Aleem Ullah
Roll no: 22120014
LUMS School of Science and Engineering

Friday, April, 07, 2023

1 Abstract

Simon's algorithm is one of the first quantum algorithms to demonstrate an exponential speedup over classical algorithms for a specific problem. In this report, we provide a comprehensive analysis of Simon's algorithm, including its theoretical background, implementation, and applications. We present the algorithm's complexity analysis and compare it to classical algorithms for solving the same problem. Our findings demonstrate the significant potential of Simon's algorithm in solving complex problems that are difficult to tackle with classical algorithms. We also discuss the quantum circuit implementation for Simon's algorithm. In addition, we present a detailed computational code for implementing this algorithm and analyze its time and space complexity.

2 Introduction

Quantum computing has emerged as a promising paradigm for solving computationally difficult problems in a wide range of fields, including computer science and cryptography. Among the many quantum algorithms that have been proposed, Simon's algorithm stands out for its ability to efficiently solve the hidden subgroup problem, which has important implications for a variety of applications.

The hidden subgroup problem (HSP) is a topic of research in mathematics and theoretical computer science. It is a problem of finding a subgroup of a given group that is hidden by a function. The HSP captures problems such as factoring, discrete logarithm, graph isomorphism, and the shortest vector problem.

In this report, we provide a comprehensive analysis of Simon's algorithm, including its theoretical foundations, computational implementation, and experimental results. We present here a detailed explanation of Simon's algorithm, including its input, output, and the steps involved in its computation. We then compare its performance to classical algorithms for solving the same problem. Finally, we present our findings on the experimental implementation of Simon's algorithm, including its limitations and potential future developments. Overall, this report aims to provide a comprehensive understanding of Simon's algorithm and its significance in the field of quantum computing.

3 Mathematical Formulation

3.1 Simon's Problem

We are given an unknown blackbox function $\boldsymbol{f}$, which is guaranteed to be either one-to-one (1 : 1) or two-to-one (2 : 1), where one-to-one and two-to-one functions have the following properties:

One-to-One: Maps exactly one unique output for every input. An example with a function that takes 4 inputs is:

$$f(1) \longrightarrow 1 \ , \ f(2) \longrightarrow 2 \ , \ f(3) \longrightarrow 3 \ , \ f(4) \longrightarrow 4$$

Two-to-One: Maps exactly two inputs to every unique output. An example with a function that takes 4 inputs is:

$$f(1) \longrightarrow 1 \ , \ f(2) \longrightarrow 2 \ , \ f(3) \longrightarrow 1 \ , \ f(4) \longrightarrow 2$$

This two-to-one mapping is according to a hidden bitstring, b, where:

given
$$x_1 \ , \ x_2 \ : \ f(x_1) \ = \ f(x_2)$$
in this case:
$$x_2 \ = \ x_1 \ \oplus \ b$$
it is guaranteed:
$$x_1 \ \oplus \ x_2 \ = \ b$$

Suppose we have a blackbox $\boldsymbol{f}$, how quickly can we tell if $\boldsymbol{f}$ is one-to-one or two-to-one? Then, if $\boldsymbol{f}$ turns out to be two-to-one, how quickly can we find out b? As it is clear, both cases boil down to the same query of finding b.

3.2 Classical Solution

Classically, if we want to measure b with 100% certainty for a given $\boldsymbol{f}$, we have to check our inputs up to $(2^{n-1} + 1)$ times, where n is the number of bits in our input. This means checking over half of all the possible inputs until we find two cases with the same output.

We could solve the problem with our first two tries only. But if we get an $\boldsymbol{f}$ that is one-to-one, or get really unlucky with an $\boldsymbol{f}$ that's two-to-one, then we have to

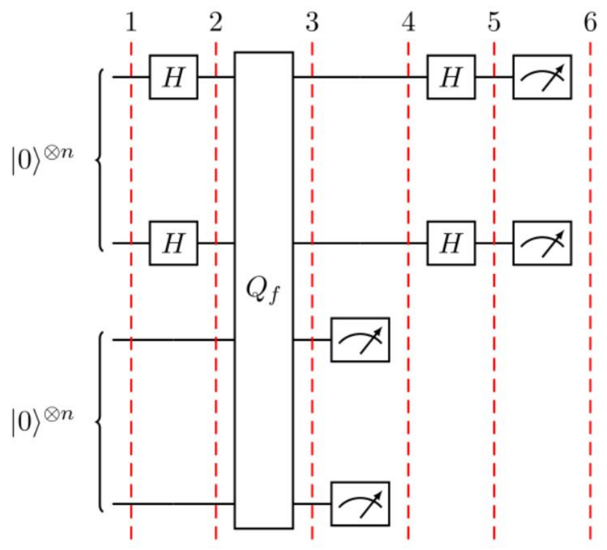


Figure 1: Quantum Circuit

go with the full $(2^{n-1}+1)$ check. There are known algorithms that can do the job in $O(2^{n/2})$ checks (Randomized algorithm), but generally speaking the complexity grows exponentially with n.

3.3 Quantum Solution

The quantum circuit that implements Simon's algorithm is shown in Figure (1). Where the query function, $\boldsymbol{Q_f}$ acts on two quantum registers as:

$$|x\rangle|a\rangle \longrightarrow |x\rangle|a \oplus f(x)\rangle$$

In the specific case that the second register is in the state $|0\rangle = |0\rangle|0\rangle....|0\rangle$ the above equation becomes

$$|x\rangle|0\rangle \longrightarrow |x\rangle|f(x)\rangle$$

The Simon's algorithm acts in the following steps [2]:

3.3.1 1st step

Two input registers, each of n-qubits length, are initialized to the zero state:

$$|\psi_1\rangle = |0\rangle^{\otimes n}|0\rangle^{\otimes n}$$

3.3.2 2nd step

Hadamard gate applies to the first register:

$$|\psi_2\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle|0\rangle^{\otimes n}$$

3.3.3 3rd step

Now we apply query function Q_f:

$$|\psi_3\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle|f(x)\rangle$$

3.3.4 4th step

We get a certain value of $f(x)$ after measuring the second register. Because of the value obtained on the second register, our first register must have some specific related values. If our second register gets $f(x)$ value, first register must possess two values

$$x \; and \; y$$

where

$$y = x \oplus b$$

Both of these values will be in equal superposition

$$|\psi_4\rangle = \frac{1}{\sqrt{2}}(|x\rangle + |y\rangle)$$

$$|\psi_4\rangle = \frac{1}{\sqrt{2}}(|x\rangle + |x \oplus b\rangle)$$

Notice that we have not written here the second register as it is already measured [1].

3.3.5 5th step

Now Hadamard gate is applied on the first register:

$$|\psi_5\rangle = \frac{1}{\sqrt{2^{n+1}}} \sum_{z\in\{0,1\}^n} [(-1)^{x.z} + (-1)^{y.z}]|z\rangle$$

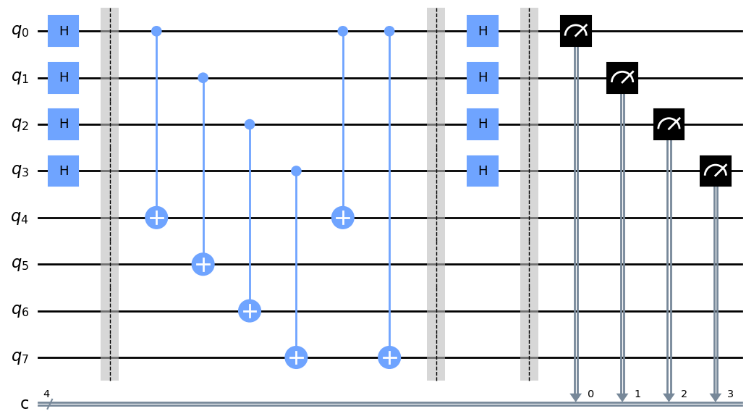


Figure 2: Quantum Circuit

3.3.6 6th step

Now comes the measurement of first register and it will give us result only if:

$$x.z = y.z$$

$$x.z = (x \oplus b).z$$

$$x.z = x.z \oplus b.z$$

$$b.z = 0(\text{mod } 2)$$

We will get certain value of z whose dot product with the hidden variable b is zero. By repeating this algorithm n times, we will get n different values of z.

To measure b, we only need $(n-1)$ linearly independent values of z (n is the number of qubits here). By doing simple linear algebra of these $(n-1)$ linearly independent equations, we can find b (our hidden string).

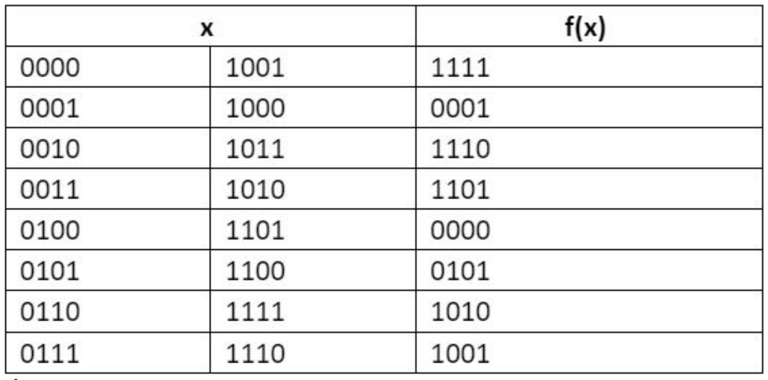
3.4 Example

Let's now see an example of simon's algorithm for 4 qubits. Our secret bitstring b in this case is 1001. Quantum circuit for this example is shown in Figure (2).

3.4.1 1st step

Two 4-qubit registers are initialized by an input state ket zero.

$$|\psi_1\rangle \;\; = \;\; |0000\rangle|0000\rangle$$



x		f(x)
0000	1001	1111
0001	1000	0001
0010	1011	1110
0011	1010	1101
0100	1101	0000
0101	1100	0101
0110	1111	1010
0111	1110	1001

Figure 3: Corresponding outputs for all possible inputs

3.4.2 2nd step

Hadamard gates are applied to the first register only.

$$|\psi_2\rangle \;=\; \frac{1}{\sqrt{2^4}} \sum_{x\in\{0,1\}^4} |x\rangle|0000\rangle$$

$$|\psi_2\rangle \;=\; \frac{1}{4}(|0000\rangle + |0001\rangle + |0010\rangle + |0100\rangle + ... + |1111\rangle) \otimes |0000\rangle$$

3.4.3 3rd step

Now this state $|\psi_2\rangle$ will pass through the oracle and we will have the following state as a result:

$$|\psi_3\rangle = \frac{1}{4} \sum_{x\in\{0,1\}^4} |x\rangle|f(x)\rangle$$

Oracle outputs for all the possible inputs is shown in Table (3). Following is the corresponding output for all possible inputs:

$$|\psi_3\rangle = \frac{1}{4}[|0000\rangle|1111\rangle + |0001\rangle|0001\rangle + |0010\rangle|1110\rangle + ... + |0111\rangle|1001\rangle + |1111\rangle|1010\rangle]$$

3.4.4 4th step

Now we measure our second register. For each measured state on the second register, there will be only two corresponding states (in equal superposition) on the first register. It is because of the fact that for Two-to-One function, two input states can produce the same output state.

$$|\psi_4\rangle = \frac{1}{\sqrt{2}}(|x\rangle + |x \oplus b\rangle) \otimes |f_(x)\rangle$$

If our second register is measured to be 1010 then ψ_4 will be

$$|\psi_4\rangle = \frac{1}{\sqrt{2}}(|0110\rangle + |0110 \oplus 1001\rangle) \otimes |1010\rangle$$

$$|\psi_4\rangle = \frac{1}{\sqrt{2}}(|0110\rangle + |1111\rangle) \otimes |1010\rangle$$

We can ignore second register from now onward as it has been measured.

$$|\psi_4\rangle = \frac{1}{\sqrt{2}}(|0110\rangle + |1111\rangle)$$

3.4.5 5th step

$$|\psi_5\rangle = \frac{1}{\sqrt{2^{n+1}}} \sum_{z\in\{0,1\}^n} [(-1)^{x.z} + (-1)^{y.z}]|z\rangle$$

$$|\psi_5\rangle = \frac{1}{\sqrt{2^{n+1}}} \sum_{z\in\{0,1\}^n} [(-1)^{x.z} + (-1)^{(x \,\oplus\, b).z}]|z\rangle$$

Hadamard gate is applied on the first register in this step

$$|\psi_5\rangle = H^{\otimes n}(\frac{1}{\sqrt{2}}(|0110\rangle + |1111\rangle))$$

After applying Hadamard gate and doing further simplifications, we get

$$|\psi_5\rangle = \frac{1}{\sqrt{2^3}}(|0000\rangle - |0010\rangle - |0100\rangle + |0110\rangle + |1001\rangle - |1011\rangle - |1101\rangle + |1111\rangle)$$

These are the values of z that satisfy the condition

$$b.z = 0(\text{mod } 2)$$

For all values of z, we can write that

$$b.z^1 = 0$$

$$b.z^2 = 0$$

$$b.z^3 = 0$$

$$.$$

$$.$$

$$.$$

$$b.z^8 = 0$$

Here z^1, z^2,...,z^8 are first, second, ..., eigth values of z respectively.

We need here atleast three linearly independent values of z to find b. In this case three linearly independent values of z are $(0010, 0100, 1001)$. We can also write them in equation form as follows (subscript shows integer number from left to right in the bit-string):

$$z_1^1.b_1 + z_2^1.b_2 + z_3^1.b_3 + z_4^1.b_4 = 0$$

$$z_1^2.b_1 + z_2^2.b_2 + z_3^2.b_3 + z_4^2.b_4 = 0$$

$$z_1^3.b_1 + z_2^3.b_2 + z_3^3.b_3 + z_4^3.b_4 = 0$$

We now write them in matrix form to solve for b:

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Swapping R_1 and R_3:

$$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

b_4 is free variable here. Suppose $b_4 = 1$

$$b_2 = 0$$

$$b_3 = 0$$

$$b_1 + b_4 = 0$$

$$b_1 = -b_4$$

$$b_1 = -1$$

As we are doing mod2 calculation, so;

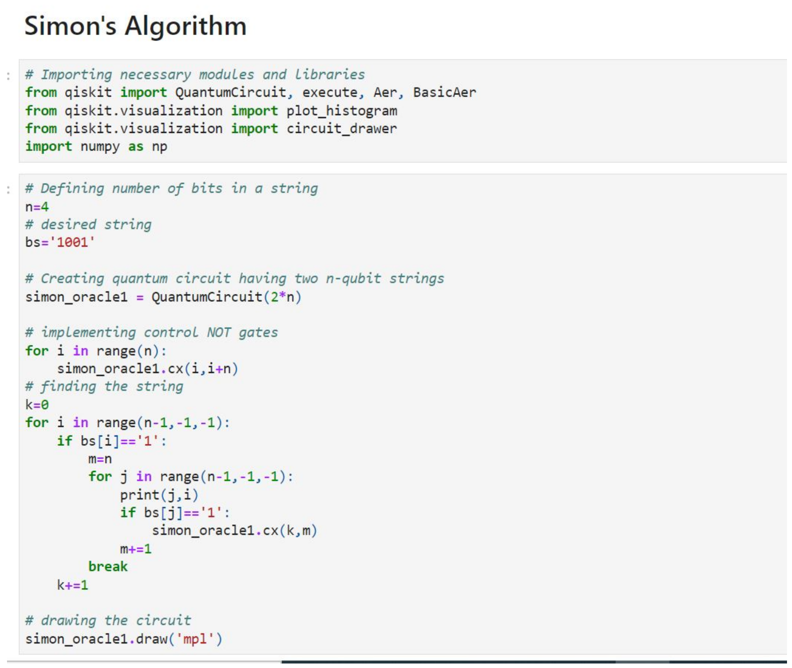
$$b_1 = 1$$

Hence, our secret bit-string b is (1001).

We have solved our problem in just $(n-1)$ iterations here. In worst case, it would take maximum of n iteration to find b. It clearly shows that Simon's algorithm provides exponential speed up over classical solution.

4 Implementation Methodology

We now implement simon's algorithm on python for the example used above. Each step of implementong the algorithm is shown in the Figures below. Final calculation for the secret bit-string b is after getting z values is the same as done in the example.



Simon's Algorithm

```
# Importing necessary modules and libraries
from qiskit import QuantumCircuit, execute, Aer, BasicAer
from qiskit.visualization import plot_histogram
from qiskit.visualization import circuit_drawer
import numpy as np
```

```
# Defining number of bits in a string
n=4
# desired string
bs='1001'

# Creating quantum circuit having two n-qubit strings
simon_oracle1 = QuantumCircuit(2*n)

# implementing control NOT gates
for i in range(n):
    simon_oracle1.cx(i,i+n)
# finding the string
k=0
for i in range(n-1,-1,-1):
    if bs[i]=='1':
        m=n
        for j in range(n-1,-1,-1):
            print(j,i)
            if bs[j]=='1':
                simon_oracle1.cx(k,m)
            m+=1
        break
    k+=1

# drawing the circuit
simon_oracle1.draw('mpl')
```

Figure 4: Defining quantum circuit and implementing the oracle

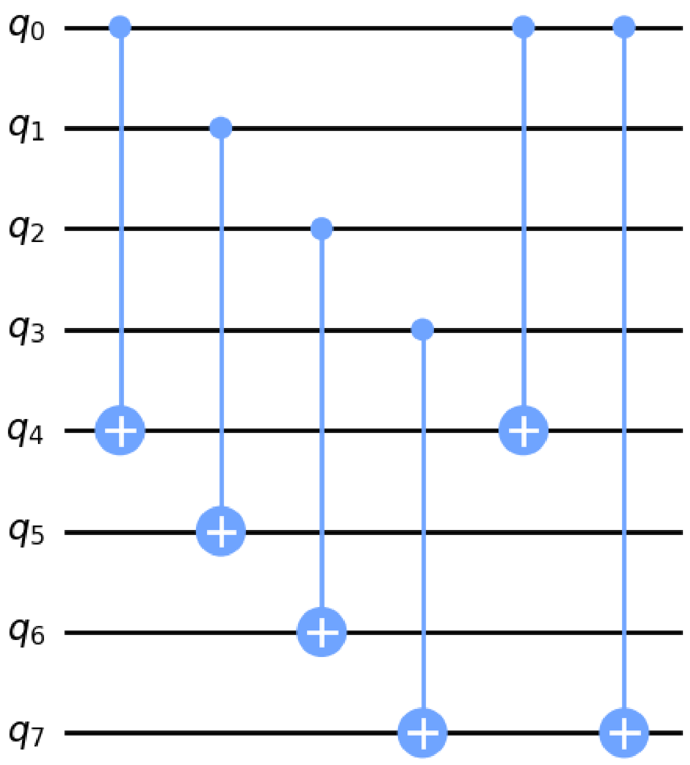
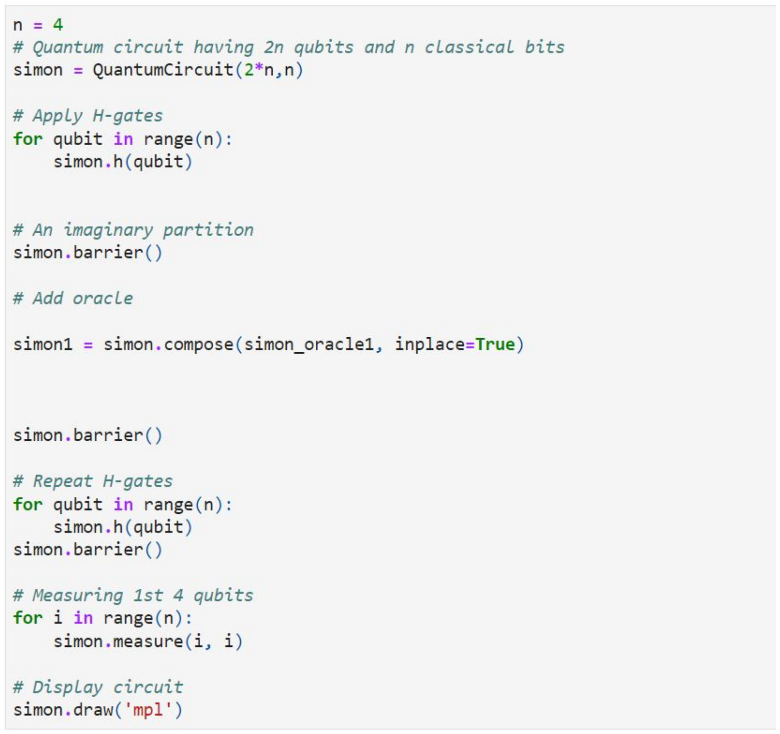


Figure 5: Oracle Diagram



```
n = 4
# Quantum circuit having 2n qubits and n classical bits
simon = QuantumCircuit(2*n,n)

# Apply H-gates
for qubit in range(n):
    simon.h(qubit)


# An imaginary partition
simon.barrier()

# Add oracle

simon1 = simon.compose(simon_oracle1, inplace=True)



simon.barrier()

# Repeat H-gates
for qubit in range(n):
    simon.h(qubit)
simon.barrier()

# Measuring 1st 4 qubits
for i in range(n):
    simon.measure(i, i)

# Display circuit
simon.draw('mpl')
```

Figure 6: Implementing complete Quantum Circuit

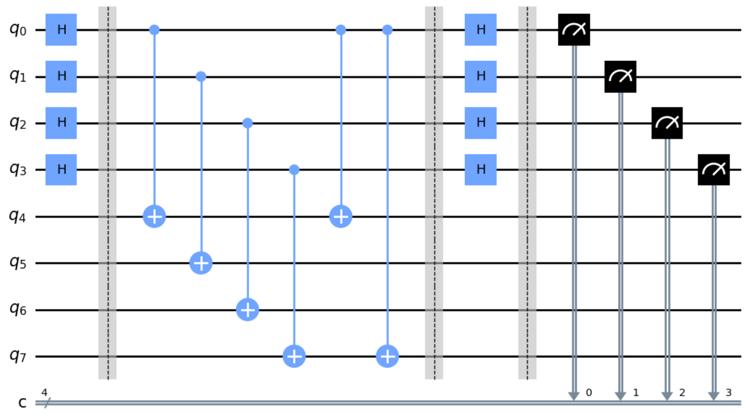
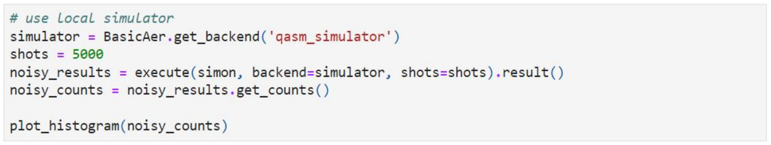


Figure 7: Quantum Circuit



```
# use local simulator
simulator = BasicAer.get_backend('qasm_simulator')
shots = 5000
noisy_results = execute(simon, backend=simulator, shots=shots).result()
noisy_counts = noisy_results.get_counts()

plot_histogram(noisy_counts)
```

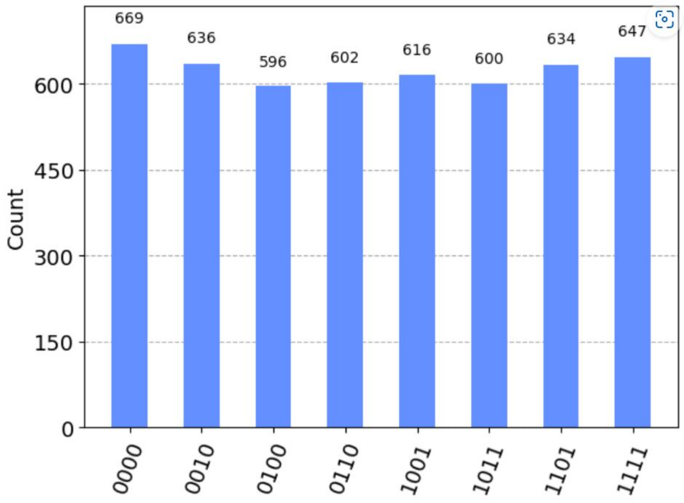


Figure 8: Simulating the results

5 Applications

Simon's algorithm has several applications in various fields, particularly in computer science and cryptography. Here are some key applications:

5.1 Cryptography

Simon's algorithm has significant implications for cryptography. It can be used to break certain types of cryptographic protocols that rely on the difficulty of finding the period of a function. By efficiently determining the period, Simon's algorithm can undermine the security of these protocols, highlighting the need for more robust cryptographic schemes.

5.2 Group Theory

Simon's algorithm has applications in group theory, a branch of mathematics that studies symmetries and abstract algebraic structures. It can help identify hidden subgroups within groups, enabling the analysis of group properties and aiding in solving computational problems related to group theory.

5.3 Quantum Algorithm Design

Simon's algorithm serves as a foundation for the development of more advanced quantum algorithms. It provides valuable insights into the use of quantum techniques, such as quantum parallelism and interference, in designing efficient algorithms for solving problems that involve hidden subgroups.

6 Conclusion

In conclusion, Simon's algorithm has proven to be a powerful tool for solving the hidden subgroup problem in quantum computing. We have seen that how can it provide us exponential benefit over the classical algorithm. Simon's algorithm also paved the way for the development of the most famous Shor's algorithm.As the field of quantum computing progresses, further research and exploration of Simon's algorithm and its applications will undoubtedly contribute to the advancement of computational capabilities and information theory.

References

[1] Qiskit.

[2] Daniel R. Simon. On the power of quantum computation. *SIAM Journal on Computing*, 26(5):1474–1483, 1997.