

Minicurso: Introducción al cómputo cuántico.

Taller de tecnologías cuánticas UAMI

Alejandro Kunold¹

¹UAM A, México

31 de julio de 2024

Programa del minicurso

Introducción

Historia y aplicaciones

Computadoras cuánticas

Compuertas y algoritmos cuánticos

Circuitos cuánticos

Compuertas unitarias

Qbits

Compuertas lógicas

Implementación de algoritmos

Instalación de Qiskit

Algoritmos en Qiskit

Introducción

Historia y aplicaciones

Computadoras cuánticas

Compuertas y algoritmos cuánticos

Circuitos cuánticos

Compuertas unitarias

Qbits

Compuertas lógicas

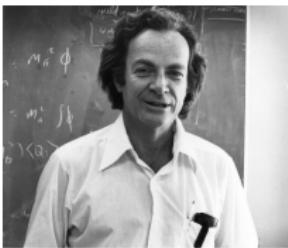
Implementación de algoritmos

Instalación de Qiskit

Algoritmos en Qiskit

Introducción Historia y Aplicaciones

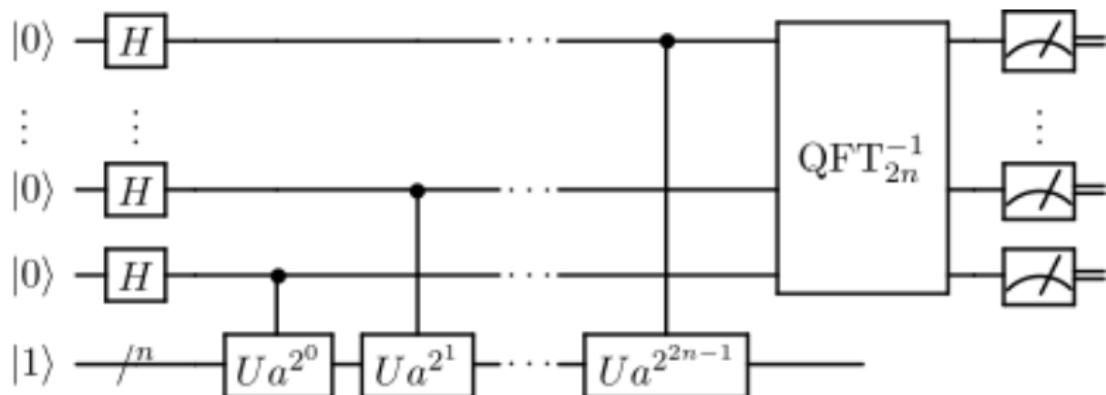
- ▶ Tres artículos fundacionales de Feynmann¹:
 1. Simulating physics with computers (1982): artículo principal con aspectos abstractos sobre como simular sistemas cuánticos pero sin detalles de como sería una computadora cuántica o un algoritmo cuántico.
 2. There's Plenty of Room at the Bottom (1960). Relacionado con la nanotecnología.
 3. Quantum Mechanical Computers (1985): Detalles en como funcionaría una computadora cuántica y cómo sería un algoritmo cuántico. No habla ni de qbits ni de quantum gates.



¹<https://jackkrupansky.medium.com/feynmans-three-papers-related-to-quantum-computing-dd6f9847e6ad>

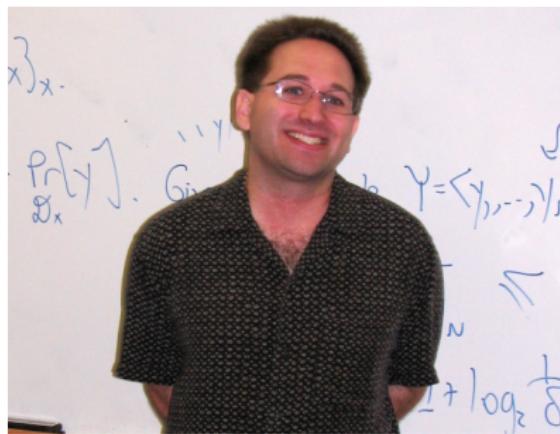
Introducción Historia y Aplicaciones

► Algoritmo de Shor²



Introducción Historia y Aplicaciones

- ▶ Qué ha pasado 40 años después Preskill³.
- ▶ Presposterous universe, Mindscape Preskill⁴
- ▶ Scott Aaronson⁵, Quantum Computing: Dismantling the Hype



³<https://arxiv.org/pdf/2106.10522>

⁴<https://www.preposterousuniverse.com/podcast/2021/06/28/>

153-john-preskill-on-quantum-computers-and-what-theyre-good-for/

⁵<https://www.youtube.com/watch?v=qs0D9sdbKPU&t=5218s>

Introducción Historia y Aplicaciones

Qiskit | Ecosystem
Qiskit Finance 0.4.0

Search
Descripción General
Primeros Pasos
Tutoriales
Referencia de la API
Notas de la Versión
GitHub

Descripción general de Qiskit Finance

Descripción General

Qiskit Finance es un framework de código abierto que contiene componentes de incertidumbre para problemas de acciones/valores, aplicaciones para problemas financieros, como la optimización de cartera, y proveedores de datos para obtener datos reales o aleatorios para experimentos financieros.

IBM Quantum Documentation
Overview Start Build Transpile Verify Run API reference v0.32.1

You are viewing the API reference for an old version of Qiskit SDK. Switch to latest version

Qiskit's optimization module

qiskit.optimization

qiskit.optimization

Qiskit's optimization module covers the whole range from high-level modeling of combinatorial optimization problems to the direct conversion of problems to different required representations, to a suite of easy-to-use quantum optimization algorithms that are ready to run on classical simulators, as well as on real quantum devices via Qiskit.

Qiskit SDK
v0.32.1

API index
Circuit construction
Quantum information (qiskit.quantum_info)
Transpilation
Primitives and providers
Results and visualizations
Optflow
Generalization/generalization

You are viewing the API reference for an old version of Qiskit SDK. Switch to latest version

Qiskit's chemistry module

qiskit.chemistry

This is Qiskit's chemistry module that provides for experimentation with chemistry domain problems such as ground state energy and excited state energies of molecules.

IBM Quantum Documentation
Overview Start Build Transpile Verify Run API reference v0.32.1

You are viewing the API reference for an old version of Qiskit SDK. Switch to latest version

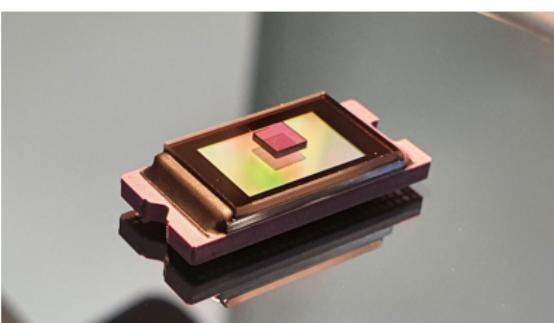
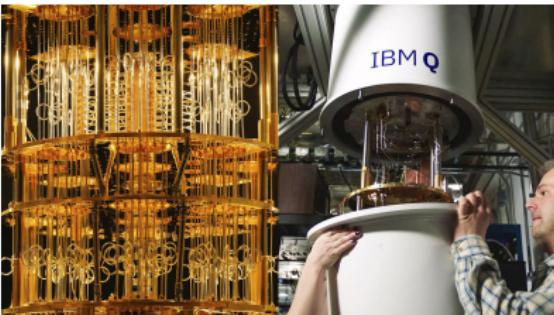
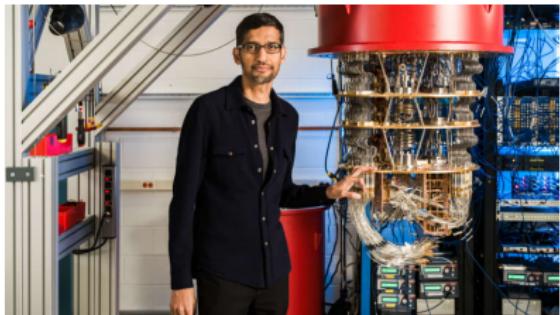
Qiskit's Machine Learning module

qiskit.ml

This is the Qiskit's machine learning module. There is an initial set of function here that will be built out over time. At present it has sample sets that can be used with Aqua's classifiers.

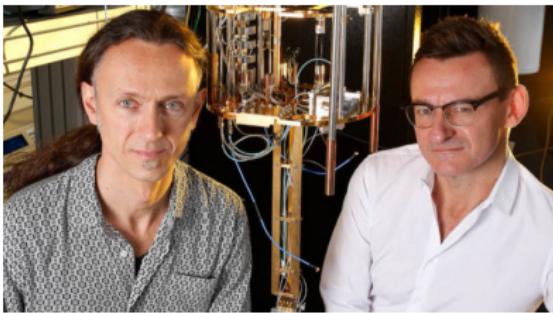
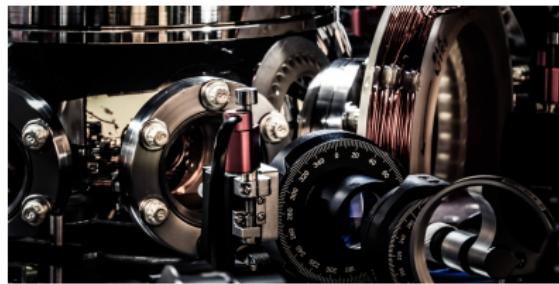
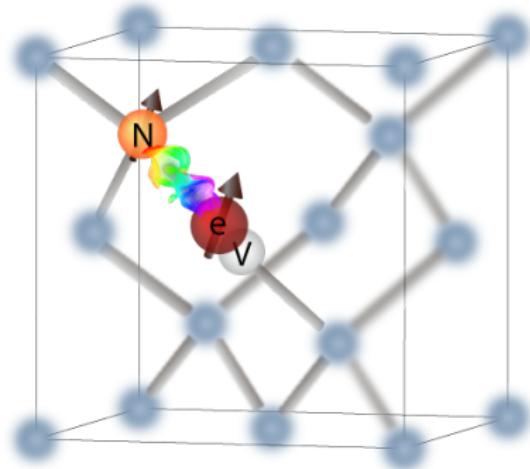
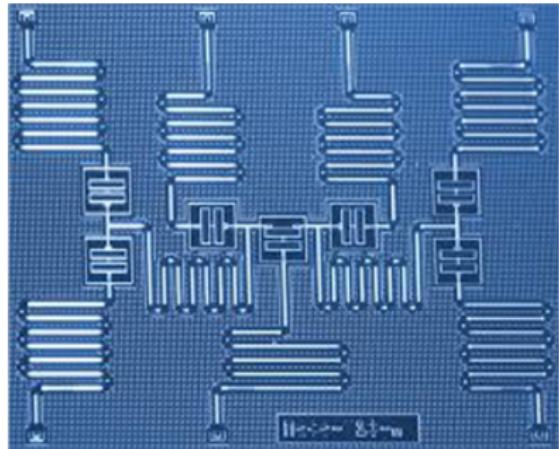
... y simulación cuántica cómo proponía Feynmann.

Introducción Computadoras cuánticas



y hay muchas más ...

Introducción Computadoras cuánticas

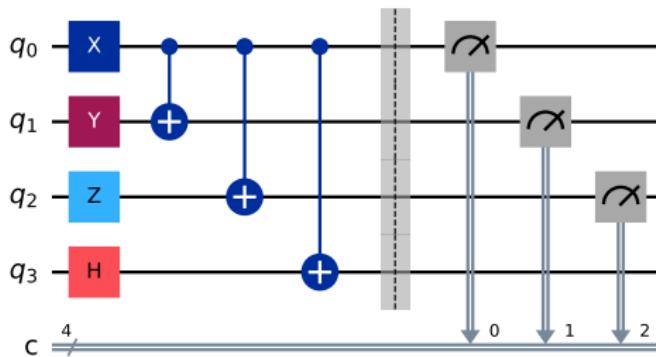


Introducción Compuertas y algoritmos cuánticos

- ▶ Las computadoras cuánticas sólo pueden resolver un problema

$$i\hbar \frac{d}{dt} |\psi(t)\rangle = H |\psi(t)\rangle$$

- ▶ Todas las compuertas son transformaciones unitarias excepto la medición.
- ▶ Síntesis cuántica:convertir en un conjunto de compuertas.



Introducción Compuertas y algoritmos cuánticos

- ▶ Puedo entender a las compuertas como matrices

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix},$$

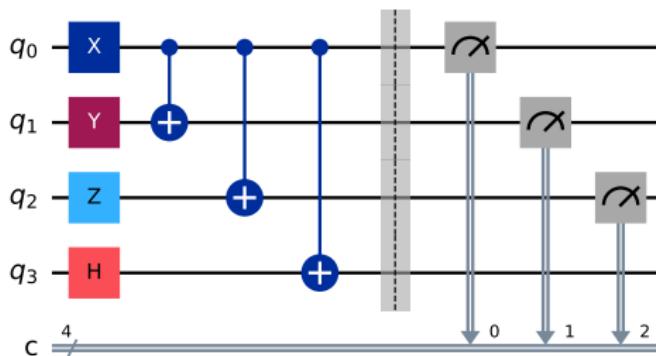
$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad T = \begin{pmatrix} 1 & 0 \\ 0 & i\pi/4 \end{pmatrix}$$

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Introducción Compuertas y algoritmos cuánticos

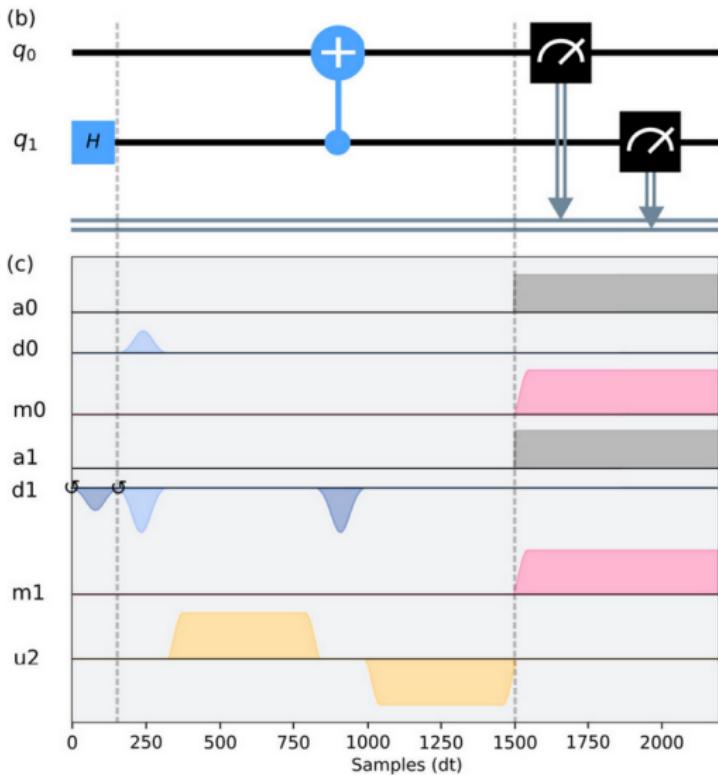
- ▶ Los circuitos cuánticos se pueden entender como productos de operadores

$$\begin{aligned} X \otimes Y \otimes Z \otimes H & [(P_0 \otimes I + P_1 \otimes X) \otimes I \otimes I] \\ & \times [(P_0 \otimes I \otimes I + P_1 \otimes I \otimes X) \otimes I] \\ & \times [(P_0 \otimes I \otimes I \otimes I + P_1 \otimes I \otimes I \otimes X)] \end{aligned}$$



Introducción Compuertas y algoritmos cuánticos

- ▶ Transpilación: Se convierten las compuertas en un protocolo de pulsos electromagnéticos.



Introducción

Historia y aplicaciones

Computadoras cuánticas

Compuertas y algoritmos cuánticos

Circuitos cuánticos

Compuertas unitarias

Qbits

Compuertas lógicas

Implementación de algoritmos

Instalación de Qiskit

Algoritmos en Qiskit

Circuitos cuánticos Compuertas unitarias

- ▶ La ecuación de Schrödinger

$$i\hbar \frac{d}{dt} |\psi(t)\rangle = H |\psi(t)\rangle$$

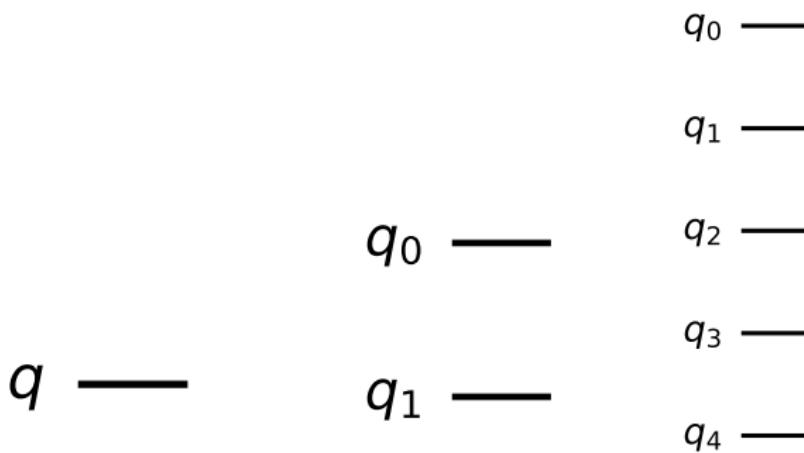
- ▶ Operador de evolución temporal es unitario

$$i\hbar \frac{d}{dt} U(t) = HU(t)$$

- ▶ Las compuertas son en realidad operadores $U(t)$
- ▶ Hay compuertas de 1 qbit o compuertas de n qbits

Circuitos cuánticos Qbits

- ▶ 1 qbit $\rightarrow |q_1\rangle$ donde $q_1 = 0, 1 \rightarrow 2$ estados
- ▶ n qbit $\rightarrow |q_n\rangle \otimes \dots \otimes |q_2\rangle \otimes |q_1\rangle = |q_n, \dots, q_2, q_1\rangle = |q\rangle$ donde $q_1, q_2, \dots, q_n = 0, 1 \rightarrow 2^n$ estados
- ▶ $q = (q_n, \dots, q_2, q_1)$ es una cadena binaria
- ▶ Los qbits siempre están inicializados a $|0\rangle$
- ▶ n qbits inicialmente $|0\rangle \otimes |0\rangle \dots |0\rangle = |0, 0, \dots, 0\rangle$



Circuitos cuánticos Qbits

Ejemplo: 1 qbit

q —

- ▶ Hay $2^1 = 2$ estados posibles son

$$|0\rangle \rightarrow \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$|1\rangle \rightarrow \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Circuitos cuánticos Qbits

Ejemplo: 2 qbits

q_0 —

q_1 —

- ▶ Hay $2^2 = 4$ estados posibles son

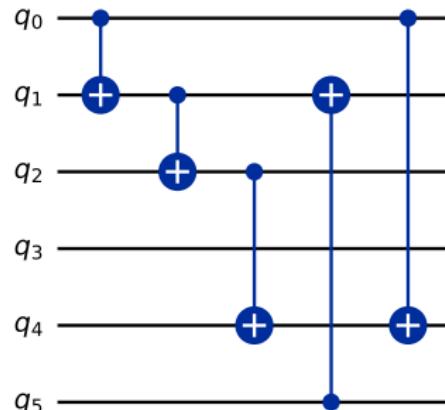
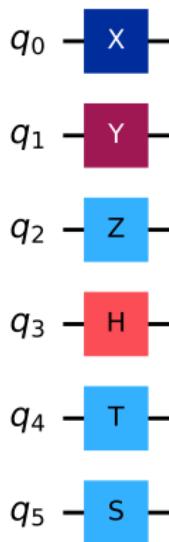
$$\begin{array}{ccc} |0\rangle \otimes |0\rangle & \rightarrow & \begin{bmatrix} 1 & \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} \\ 0 & \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \\ |0,0\rangle & & \end{array}$$
$$\begin{array}{ccc} |0\rangle \otimes |1\rangle & \rightarrow & \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \\ |0,1\rangle & & \end{array}$$

$$\begin{array}{ccc} |1\rangle \otimes |0\rangle & \rightarrow & \begin{bmatrix} 0 & \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} \\ 1 & \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \\ |1,0\rangle & & \end{array}$$
$$\begin{array}{ccc} |1\rangle \otimes |1\rangle & \rightarrow & \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \\ |0,1\rangle & & \end{array}$$

Circuitos cuánticos Compuertas lógicas

Compuertas implementadas

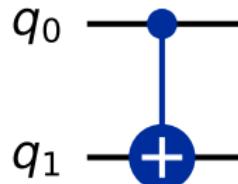
- ▶ En una CQ digital sólo son necesarios dos tipos de compuertas
- ▶ Compuertas de 1 qbit $U(\theta, \phi, \lambda)$
- ▶ Compuertas de 2 qbits $CNOT$



Circuitos cuánticos Compuertas lógicas

Compuertas de 1 qbit unitarias

- ▶ Tienen la forma general

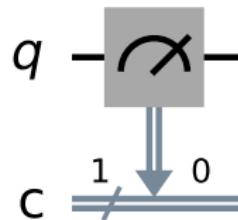


$$U(\theta, \phi, \lambda) = \begin{bmatrix} \cos(\theta/2) & -e^{i\lambda} \sin(\theta/2) \\ e^{i\phi} \sin(\theta/2) & e^{i(\lambda+\phi)} \cos(\theta/2) \end{bmatrix}$$

- ▶ $U(\theta, \phi, \lambda)$ es casi la matriz unitaria de 2×2 más general

Compuertas de 1 qbit no unitarias

- ▶ Las mediciones no son unitarias



$$M = |\alpha_0|^2 P_0, |\alpha_1|^2 P_1$$

si el qbit se encuentra en el estado

$$|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle$$

Circuitos cuánticos Compuertas lógicas

Ejemplo: La compuerta X (NOT)

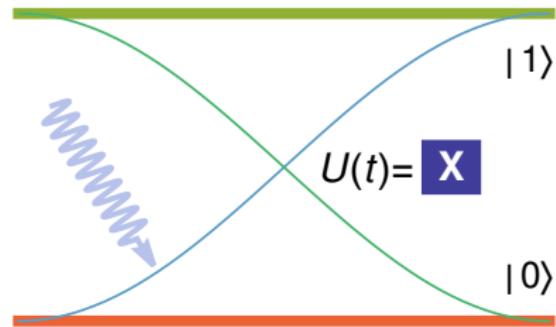
- ▶ La compuerta X se puede escribir como una matriz

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

- ▶ Su efecto es

$$X|0\rangle = |1\rangle, \quad X|1\rangle = |0\rangle, \quad X(\alpha|0\rangle + \beta|1\rangle) = (\alpha|1\rangle + \beta|0\rangle)$$

- ▶ Se puede entender como una oscilación de Rabi



Circuitos cuánticos Compuertas lógicas

Ejemplo: La compuerta Y (Pauli– Y)

- ▶ La compuerta Y se puede escribir como

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

- ▶ Su efecto es

$$\begin{aligned} Y|0\rangle &= i|1\rangle, \quad Y|1\rangle = -i|0\rangle, \\ Y(\alpha|0\rangle + \beta|1\rangle) &= (i\alpha|1\rangle - i\beta|0\rangle) \end{aligned}$$

- ▶ Es unitaria

$$\begin{aligned} YY^\dagger &= YY \\ &= [-i|0\rangle\langle 1| + i|1\rangle\langle 0|] [-i|0\rangle\langle 1| + i|1\rangle\langle 0|] \\ &= |0\rangle\langle 0| + |1\rangle\langle 1| = 1 \end{aligned}$$

Circuitos cuánticos Compuertas lógicas

Ejemplo: La compuerta H (Hadamard)

- ▶ La compuerta H se puede escribir como

$$Y = \frac{1}{\sqrt{2}} |0\rangle [\langle 0| + \langle 1|] + \frac{1}{\sqrt{2}} |1\rangle [\langle 0| + \langle 1|] \rightarrow \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

- ▶ Su efecto es

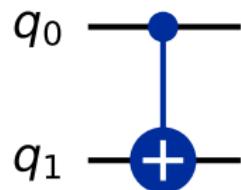
$$H|0\rangle = \frac{1}{\sqrt{2}} [|0\rangle + |1\rangle] = |+\rangle,$$

$$H|1\rangle = \frac{1}{\sqrt{2}} [|0\rangle - |1\rangle] = |-\rangle.$$

Circuitos cuánticos Compuertas lógicas

Compuertas de 2 qbits

- ▶ Hay muchas pero la más importante es



$$CNOT = CX = P_0 \otimes I + P_1 \otimes X$$

donde

$$P_0 = |0\rangle\langle 0|, \quad I = |0\rangle\langle 0| + |1\rangle\langle 1|$$

$$P_1 = |1\rangle\langle 1|, \quad X = |0\rangle\langle 1| + |1\rangle\langle 0|$$

- ▶ Consiste en una compuerta X (NOT) controlada

$$CNOT |0\rangle \otimes |0\rangle = |0\rangle \otimes |0\rangle \quad CNOT |0\rangle \otimes |1\rangle = |0\rangle \otimes |1\rangle$$

$$CNOT |1\rangle \otimes |0\rangle = |1\rangle \otimes |1\rangle \quad CNOT |1\rangle \otimes |1\rangle = |1\rangle \otimes |0\rangle$$

Circuitos cuánticos Compuertas lógicas

Compuerta *CNOT* y entrelazamiento cuántico

- ▶ Se pueden producir los 4 estados de Bell con *CNOT*

$$|\phi^+\rangle = \frac{1}{\sqrt{2}} \left[|0\rangle \otimes |0\rangle + |1\rangle \otimes |1\rangle \right],$$

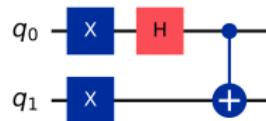
$$|\phi^-\rangle = \frac{1}{\sqrt{2}} \left[|0\rangle \otimes |0\rangle - |1\rangle \otimes |1\rangle \right],$$

$$|\psi^+\rangle = \frac{1}{\sqrt{2}} \left[|0\rangle \otimes |1\rangle + |1\rangle \otimes |0\rangle \right],$$

$$|\psi^-\rangle = \frac{1}{\sqrt{2}} \left[|0\rangle \otimes |1\rangle - |1\rangle \otimes |0\rangle \right].$$

Circuitos cuánticos Compuertas lógicas

$|\psi^-\rangle$ y $CNOT$



$$\begin{aligned} |\psi^-\rangle &= CNOT HX \otimes X |0\rangle \otimes |0\rangle \\ &= \frac{1}{\sqrt{2}} [|0\rangle \otimes |1\rangle - |1\rangle \otimes |0\rangle] \end{aligned}$$

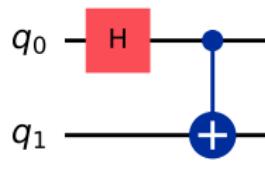
Demostración

- Sustituyendo las formas explícitas de las compuertas

$$\begin{aligned} CNOT[HX \otimes X] |0\rangle \otimes |0\rangle &= CNOT HX |0\rangle \otimes X |0\rangle \\ &= CNOT H |1\rangle \otimes |1\rangle = CNOT \frac{1}{\sqrt{2}} [|0\rangle - |1\rangle] \otimes |1\rangle \\ &= CNOT \frac{1}{\sqrt{2}} [|0\rangle \otimes |1\rangle - |1\rangle \otimes |1\rangle] \\ &= [P_0 \otimes I + P_1 \otimes X] \frac{1}{\sqrt{2}} [|0\rangle \otimes |1\rangle - |1\rangle \otimes |1\rangle] \end{aligned}$$

Circuitos cuánticos Compuertas lógicas

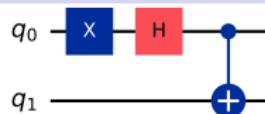
$|\phi^+\rangle$ y $CNOT$



Quantum circuit diagram showing two qubits, q_0 and q_1 . q_0 passes through a Hadamard gate (H). The two qubits are then coupled via a CNOT gate, represented by a blue circle with a plus sign (+). A right-pointing arrow leads to the mathematical expression for the state.

$$|\phi^+\rangle = CNOT H \otimes I |0\rangle \otimes |0\rangle$$
$$= \frac{1}{\sqrt{2}} [|0\rangle \otimes |0\rangle + |1\rangle \otimes |0\rangle]$$

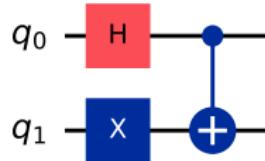
$|\phi^-\rangle$ y $CNOT$



Quantum circuit diagram showing two qubits, q_0 and q_1 . q_0 passes through a NOT gate (X) followed by a Hadamard gate (H). The two qubits are then coupled via a CNOT gate, represented by a blue circle with a plus sign (+). A right-pointing arrow leads to the mathematical expression for the state.

$$|\phi^-\rangle = CNOT HX \otimes I |0\rangle \otimes |0\rangle$$

$|\psi^+\rangle$ y $CNOT$



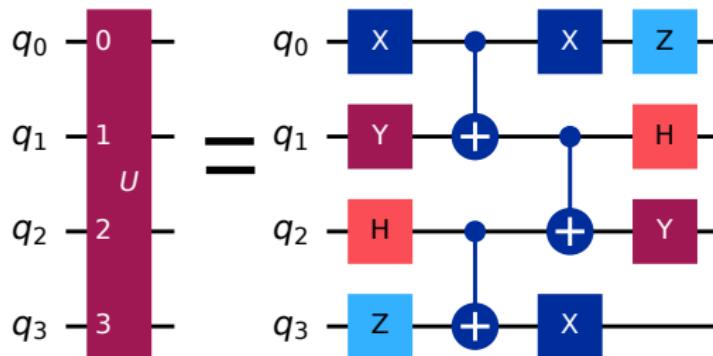
Quantum circuit diagram showing two qubits, q_0 and q_1 . q_0 passes through a Hadamard gate (H). q_1 passes through a NOT gate (X). The two qubits are then coupled via a CNOT gate, represented by a blue circle with a plus sign (+). A right-pointing arrow leads to the mathematical expression for the state.

$$|\psi^+\rangle = CNOT HX \otimes I |0\rangle \otimes |0\rangle$$

Circuitos cuánticos Compuertas lógicas

Teorema de Universalidad

- ▶ Cualquier compuerta de n qbits se puede implementar con una combinación de compuertas de 1 qbit y $CNOTs$



Introducción

Historia y aplicaciones

Computadoras cuánticas

Compuertas y algoritmos cuánticos

Circuitos cuánticos

Compuertas unitarias

Qbits

Compuertas lógicas

Implementación de algoritmos

Instalación de Qiskit

Algoritmos en Qiskit

Implementación de algoritmos Instalación de Qiskit

python

Instalar Python

env

```
$ python3 -m venv /path/to/qiskit-1.0-venv
```

```
$ source /qiskit/bin/activate
```

```
(qiskit)$
```

Implementación de algoritmos Instalación de Qiskit

Youtube ⁶



Qiskit

```
(qiskit)$ pip install 'qiskit>=1'  
(qiskit)$ pip install qiskit_aer  
(qiskit)$ pip install matplotlib  
(qiskit)$ pip install pylatexenc
```

⁶https://www.youtube.com/watch?v=dZWz4Gs_BuI

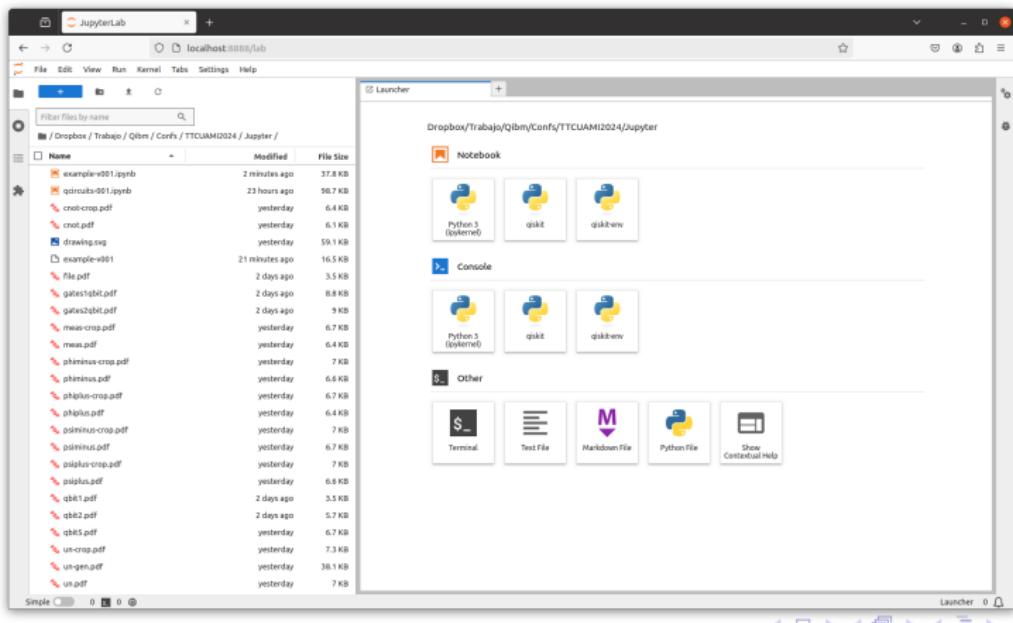
Implementación de algoritmos Instalación de Qiskit

jupyter

```
(qiskit)$ pip install jupyter lab
```

```
(qiskit)$ python -m ipykernel install --user --name=qiskit
```

```
(qiskit)$ jupyter lab
```

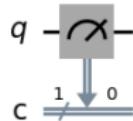


Implementación de algoritmos Algoritmos en Qiskit

Círculo 1, Creando el circuito

```
[1]: import numpy as np  
import matplotlib.pyplot as plt  
  
[2]: from qiskit import QuantumCircuit  
from qiskit import transpile  
  
from qiskit_aer import AerSimulator  
  
from qiskit.visualization import plot_histogram  
  
[3]: qc = QuantumCircuit(1,1)  
qc.measure(0,0)  
qc.draw(output='mpl')
```

[3]:



Teoría

- ▶ El estado inicial es por default $|0\rangle$.
- ▶ La probabilidad de estar en $|0\rangle$ es 1 y nunca sale $|1\rangle$.

Implementación de algoritmos Algoritmos en Qiskit

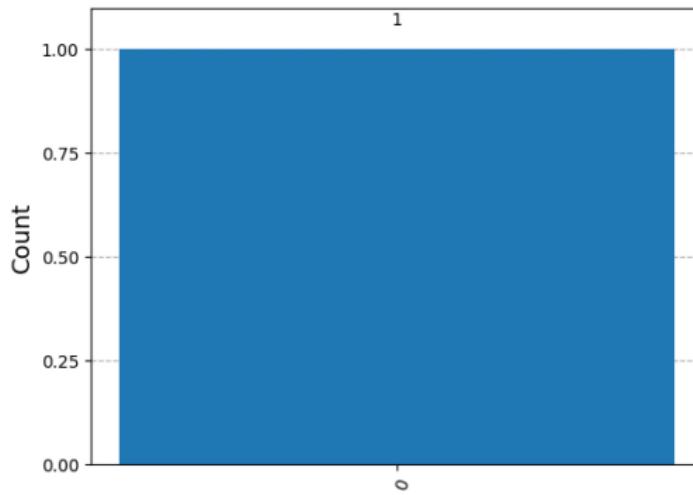
Ejecutando el circuito

```
[4]: backend = AerSimulator()
tran_qc = transpile(qc, backend=backend)
shots = 1
result = backend.run(tran_qc, shots=shots).result()
counts = result.get_counts()
counts
```

```
[4]: {'0': 1}
```

```
[5]: plot_histogram(counts)
```

```
[5]:
```



Implementación de algoritmos Algoritmos en Qiskit

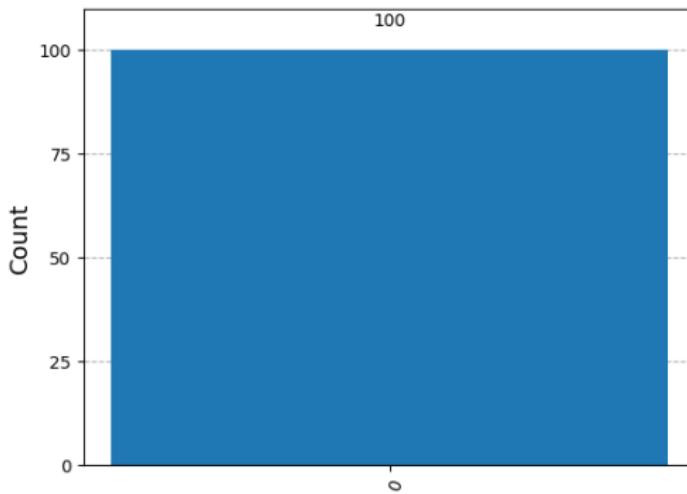
Otros experimentos

```
[6]: shots = 100  
result = backend.run(tran_qc, shots=shots).result()  
counts = result.get_counts()  
counts
```

```
[6]: {'0': 100}
```

```
[7]: plot_histogram(counts)
```

```
[7]:
```

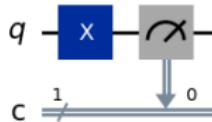


Implementación de algoritmos Algoritmos en Qiskit

Círculo 2

```
[8]: qc = QuantumCircuit(1,1)
qc.x(0)
qc.measure(0,0)
qc.draw(output='mpl')
```

[8]:



Teoría

- ▶ El estado inicial es por default $|0\rangle$.
- ▶ $X|0\rangle = |1\rangle$
- ▶ La probabilidad de estar en $|1\rangle$ es 1 y nunca sale $|0\rangle$.

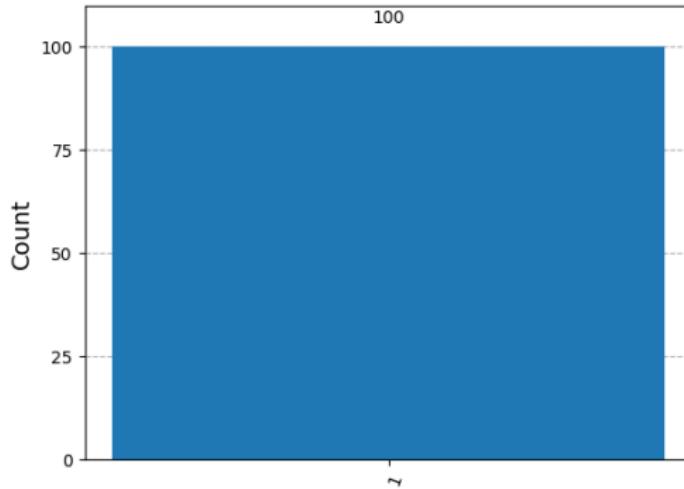
Implementación de algoritmos Algoritmos en Qiskit

Círculo 2

```
[9]: tran_qc = transpile(qc, backend=backend)

[10]: shots = 100
       result = backend.run(tran_qc, shots=shots).result()
       counts = result.get_counts()
       plot_histogram(counts)
```

[10]:

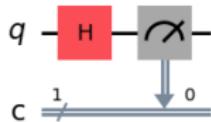


Implementación de algoritmos Algoritmos en Qiskit

Circuito 3

```
[11]: qc = QuantumCircuit(1,1)
qc.h(0)
qc.measure(0,0)
qc.draw(output='mpl')
```

[11]:



Teoría

- ▶ $H|0\rangle = \frac{1}{\sqrt{2}}[|0\rangle + |1\rangle]$.
- ▶ La probabilidad de estar en $|0\rangle$ o en $|1\rangle$ es $1/2$.
- ▶ Los dos salen el *mismo* número de veces.

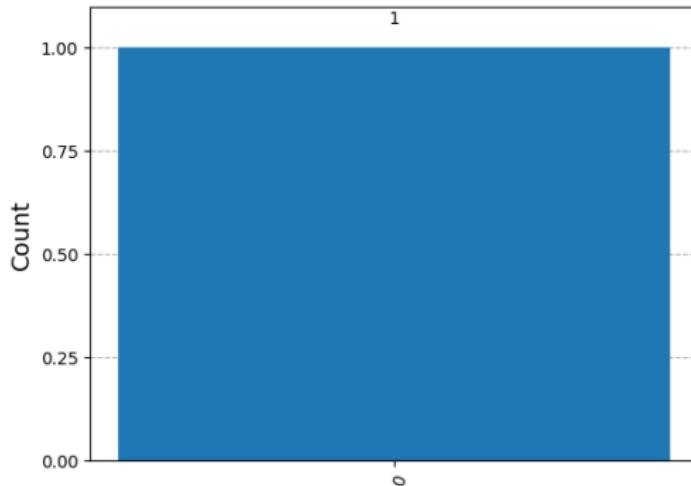
Implementación de algoritmos Algoritmos en Qiskit

Círculo 3

```
[12]: tran_qc = transpile(qc, backend=backend)

[15]: shots = 1
      result = backend.run(tran_qc, shots=shots).result()
      counts = result.get_counts()
      plot_histogram(counts)
```

[15]:

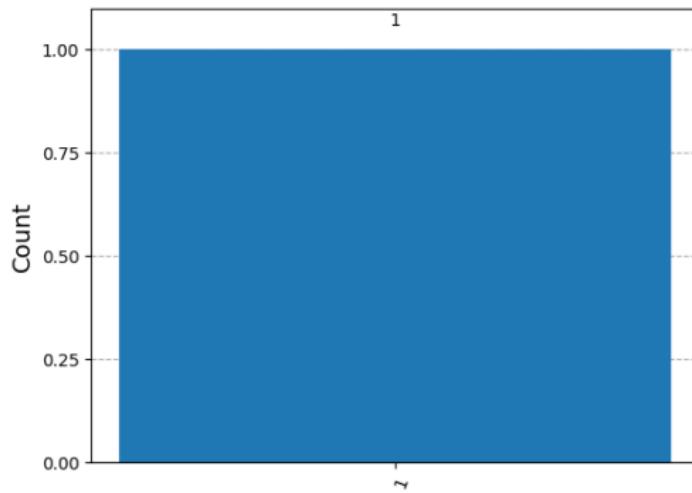


Implementación de algoritmos Algoritmos en Qiskit

Circuito 3

```
[13]: shots = 1
result = backend.run(tran_qc, shots=shots).result()
counts = result.get_counts()
plot_histogram(counts)
```

[13]:

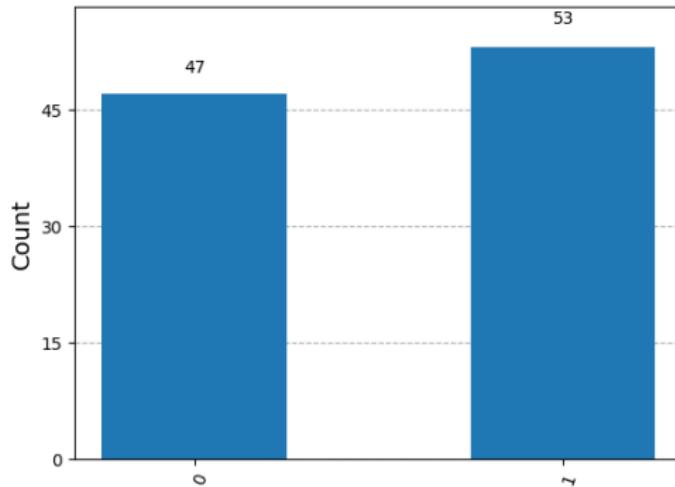


Implementación de algoritmos Algoritmos en Qiskit

Círculo 3

```
[16]: shots = 100  
result = backend.run(tran_qc, shots=shots).result()  
counts = result.get_counts()  
plot_histogram(counts)
```

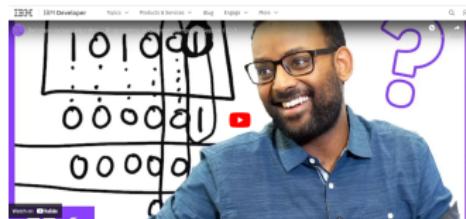
[16]:



[]:

Implementación de algoritmos Algoritmos en Qiskit

Círculo 4, Algoritmo de Bernstein-Vazirani⁷



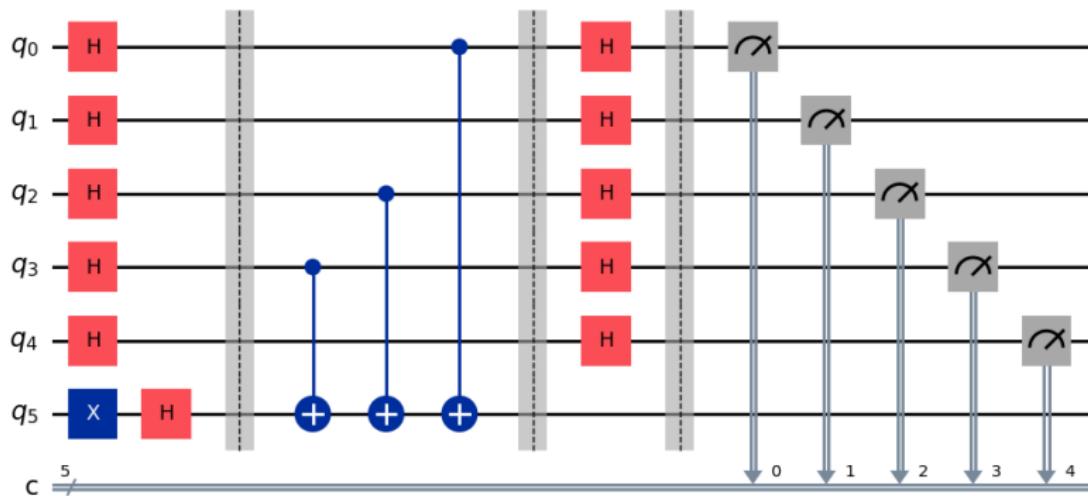
Teoría

- ▶ Cadena binaria de longitud n ($0, 1, 1, 0, 0, 1$)
- ▶ Una computadora clásica necesita n intentos para conocer todos los números de la cadena.
- ▶ El algoritmo de algoritmo de Bernstein-Vazirani sólo requiere un intento.
- ▶ Este algoritmo cuántico supera a su contraparte clásica.

⁷<https://developer.ibm.com/videos/programming-on-quantum-computers-pt-6/>

Implementación de algoritmos Algoritmos en Qiskit

Circuito del algoritmo de Bernstein-Vazirani



Implementación de algoritmos Algoritmos en Qiskit

Círculo del algoritmo en Qiskit

```
[40]: num = '01101'  
dim = len(num)
```

```
[41]: qc = QuantumCircuit(dim + 1, dim)  
qc.h(range(dim))  
qc.x(dim)  
qc.h(dim)  
qc.barrier()  
  
# codificacion del numero con CNOTs  
for ii in range(dim):  
    if num[ii] == '1':  
        qc.cx(dim - 1 - ii, dim)  
  
qc.barrier()  
  
qc.h(range(dim))  
qc.barrier()  
  
# medicion  
qc.measure(range(dim), range(dim))
```

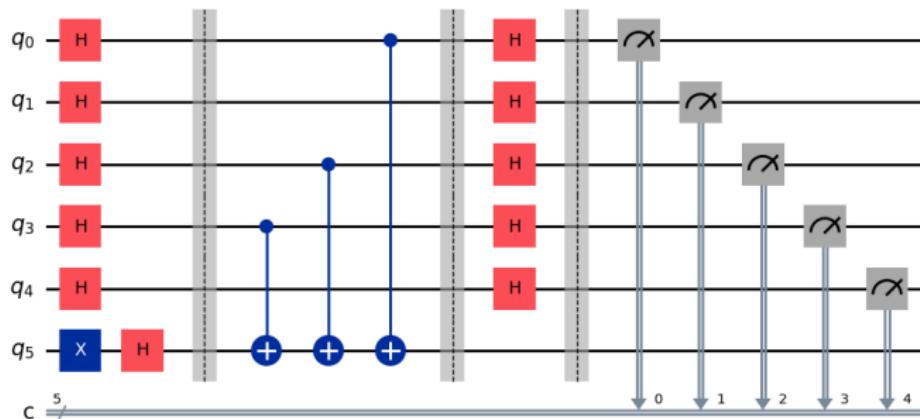
```
[41]: <qiskit.circuit.instructionset.InstructionSet at 0x7fb456be2a90>
```

Implementación de algoritmos Algoritmos en Qiskit

Circuito del algoritmo en Qiskit

```
[48]: qc.draw(output='mpl', filename='./bernstein-vazirani.png')
```

```
[48]:
```



Implementación de algoritmos Algoritmos en Qiskit

Círculo del algoritmo en Qiskit

```
[35]: backend = AerSimulator()
```

```
[44]: tran_qc = transpile(qc, backend=backend)
```

```
[45]: shots = 1
result = backend.run(tran_qc, shots=shots).result()
counts = result.get_counts()
counts
```

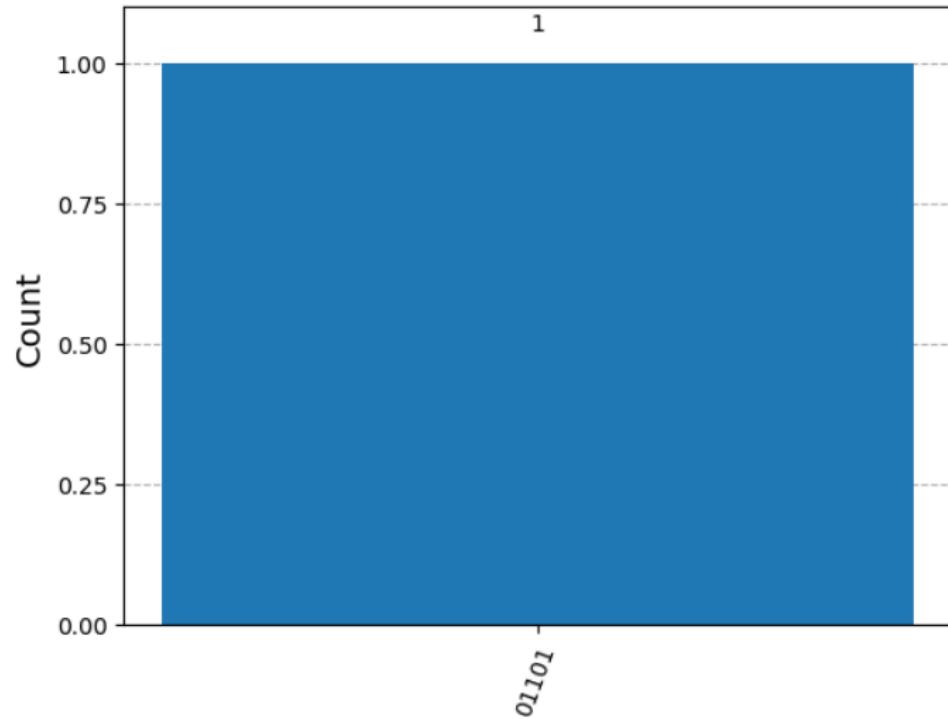
```
[45]: {'01101': 1}
```

Implementación de algoritmos Algoritmos en Qiskit

Círculo del algoritmo en Qiskit

```
[46]: plot_histogram(counts)
```

[46]:



GRACIAS