FCFM
FACULTAD DE CIENCIAS FISICO MATEMÁTICAS

Aplicaciones de la mecánica cuántica
Algoritmo de Shor

Giovanni Gamaliel López Padilla, Ivan Arturo Pla Gúzman

20 de noviembre de 2020

Introducción - Historia de las matemáticas

Teorema

Cada entero positivo tiene una única descomposición en números primos.

Introducción - Complejidad algorítmica

Notación	Nombre	Ejemplo de algoritmo
$O(1)$	Constante	Acceso a un elemento de un vector
$O(logN)$	Logarítmica	Búsqueda binaria
$O(N)$	Lineal	Búsqueda secuencial
$O(NlogN)$	Lineal-Logarítmica	Algoritmo de ordenamiento *quicksort*
$O(N^2)$	Cuadrática	Algoritmo de ordenamiento simple
$O(N^3)$	Cúbica	Multiplicación de matrices
$O(2^N)$	Exponencial	Partición de conjuntos

Tabla 1: Ejemplos de algoritmos numéricos con su clasificación O de complejidad

Introducción - Complejidad algorítmica

Complejidad algortímica del algoritmo Criba General del Cuerpo de Números.

$$O\left(exp\left[cn^{1/3}\left(logn\right)^{2/3}\right]\right) \tag{1}$$

Introducción - Origen de la computación cuántica

Tesis (de Church)

La clase de las funciones que pueden ser calculadas mediante un algoritmo coincide con la clase de las funciones recursivas.

Tesis (de Turing)

La clase de las funciones que pueden ser calculadas mediante un método definido coincide con la clase de las funciones calculables mediante una Máquina de Turing.

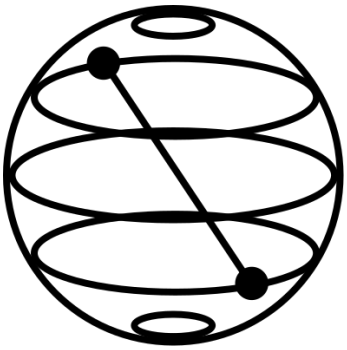
Introducción - Algoritmo de Shor

Complejidad algorítmica del algoritmo de Shor

$$O(n^2(logn)(log(logn))) \tag{2}$$

Introducción - Qiskit-IBM

(a)



(b)

In [7]:
```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from qiskit.tools.visualization import circuit_drawer
import numpy as np

qr = QuantumRegister(2)
cr = ClassicalRegister(2)
qp = QuantumCircuit(qr,cr)

qp.rx( np.pi/2,qr[0])
qp.cx(qr[0],qr[1])

qp.measure(qr,cr)

circuit_drawer(qp)
```

Out[7]:

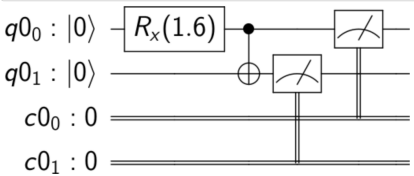


Figura 1: (a) Logo de Qiskit, (b) Ejemplo de un circuito cuántico implementado dentro de Jupyter Notebook de Python.

Objetivo

- Desarrollar el algoritmo de Shor para la factorización de un número dado usando la librería Qiskit en el lenguaje python.
- Calcular la diferencia de tiempo de procesamiento para un número N entre el algoritmo de criba general de cuerpo de números usando una computadora clásica, el algoritmo de Shor usando una computadora clásica y el algortimo de Shor usando una computadora cuántica.

Marco teórico - Estados cuánticos

Definición

Un bit cuántico, qbit o qubit es la unidad básica de información cuántica, es la versión cuántica del clásico bit binario.

Estados basicos del qubit.

$$|0\rangle\,, \qquad |1\rangle$$

Por convención se llega a utilizar la siguiente expresión:

$$|1100\rangle \rightarrow |12\rangle$$

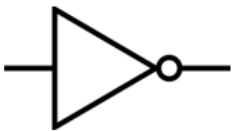


Figura 2: Representación gráfica de la compuerta NOT.

A	NOT(A)
0	1
1	0

Tabla 2: Compuerta clásica NOT.

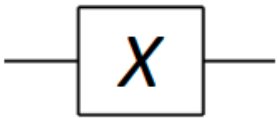


Figura 3: Representación gráfica de la compuerta cuántica NOT.

A	NOT(A)
$\|0\rangle$	$\|1\rangle$
$\|1\rangle$	$\|0\rangle$
$\|\psi\rangle$	$\hat{X}\|\psi\rangle$

Tabla 3: Compuerta cuántica NOT.

Compuertas lógicas clásicas y cuánticas - Compuerta de Hadamard

$$|0\rangle \underset{\hat{H}}{\rightarrow} \frac{|0\rangle + |1\rangle}{\sqrt{2}} \equiv |+\rangle$$

$$|1\rangle \underset{\hat{H}}{\rightarrow} \frac{|0\rangle - |1\rangle}{\sqrt{2}} \equiv |-\rangle$$

donde:

$$\hat{H} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

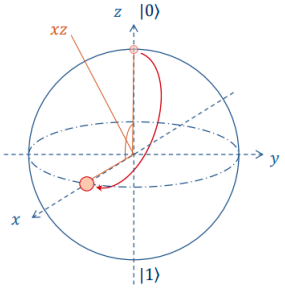


Figura 4: Representación gráfica del efecto de la transformación de Hadamard aplicada a un qubit.

Compuertas lógicas clásicas y cuánticas - Compuerta de desplazamiento de fase

$$|0\rangle \underset{\hat{R}(\phi)}{\rightarrow} |0\rangle$$

$$|1\rangle \underset{\hat{R}(\phi)}{\rightarrow} e^{i\phi}|1\rangle$$

donde:

$$\hat{R}(\phi) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{bmatrix}$$

Compuertas lógicas clásicas y cuánticas - Compuerta SWAP

$$U_{SWAP} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

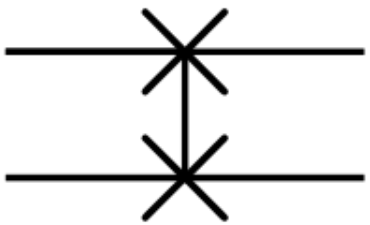


Figura 5: Representación gráfica de la compuerta SWAP.

Compuertas lógicas clásicas y cuánticas - Compuerta Controlada

$$U_{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

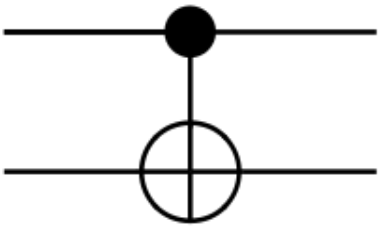


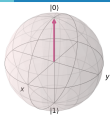
Figura 6: Representación gráfica de la compuerta NOT controlada.

Transformada de Fourier Cuántica

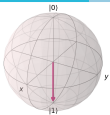
$$|\tilde{x}\rangle \equiv QFT\,|x\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{\frac{2\pi i x y}{N}}\,|y\rangle$$

$$|\tilde{x}\rangle = \frac{1}{\sqrt{N}} \bigotimes_{k=1}^{N} \left[|0\rangle + e^{2\pi i x 2^{-k}}\,|1\rangle\right].$$

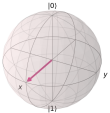
Transformada de Fourier Cuántica



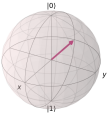
(a) Qubit con el estado cuántico $|0\rangle$ representado en la esfera de Bloch.



(b) Qubit con el estado cuántico $|1\rangle$ representado en la esfera de Bloch.



(c) Qubit con el estado cuántico $|\tilde{0}\rangle$ representado en la esfera de Bloch.



(d) Qubit con el estado cuántico $|\tilde{1}\rangle$ representado en la esfera de Bloch.

Figura 7: Representación en la esfera de Bloch de los estados bases computacionales cuánticos y las bases de Fourier.

Estimación de Fase Cuántica (QPE)

$$\mathcal{U}\,|\psi\rangle = e^{i\theta_\psi}\,|\psi\rangle\,.$$

Periodo de la función a^xmod N

$$f(x) = a^x \, mod N \tag{3}$$

$$a^r \, mod N = 1 \tag{4}$$

Usando como ejemplo a=3 y N=35, se tiene el siguiente periodo:

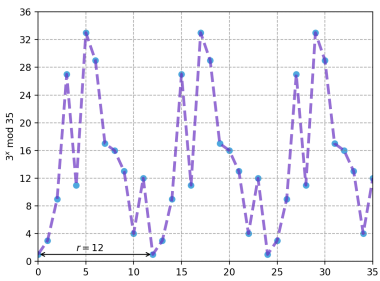


Figura 8: Periodo de la función 3 para visualizar la condición 4

Periodo de la función $a^x \bmod N$

$$
\begin{aligned}
\mathcal{U}\,|1\rangle &= |3\rangle \\
\mathcal{U}^2\,|1\rangle &= |9\rangle \\
\mathcal{U}^3\,|1\rangle &= |12\rangle \\
&\vdots \\
\mathcal{U}^{r-1}\,|1\rangle &= |12\rangle \\
\mathcal{U}^{r}\,|1\rangle &= |1\rangle
\end{aligned}
$$

$$
|u_0\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \left| a^k \, mod N \right\rangle . \qquad (5)
$$

Periodo de la función $a^x \bmod N$

$$
\begin{aligned}
|u_0\rangle &= \frac{1}{\sqrt{12}}\left(|1\rangle + |3\rangle + |9\rangle + \cdots + |4\rangle + |12\rangle\right) \\
U|u_0\rangle &= \frac{1}{\sqrt{12}}\left(U|1\rangle + U|3\rangle + U|9\rangle + \cdots + U|4\rangle + U|12\rangle\right) \\
&= \frac{1}{\sqrt{12}}\left(|3\rangle + |9\rangle + |27\rangle + \cdots + |12\rangle + |1\rangle\right) \\
&= |u_0\rangle\,.
\end{aligned}
$$

Periodo de la función $a^x \bmod N$

$$|u_1\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-\frac{2\pi i k}{r}} \left| a^k \bmod N \right\rangle$$

$$U|u_1\rangle = e^{\frac{2\pi i}{r}} |u_1\rangle .$$

$$\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |u_s\rangle = |1\rangle .$$

Periodo de la función $a^x \bmod N$

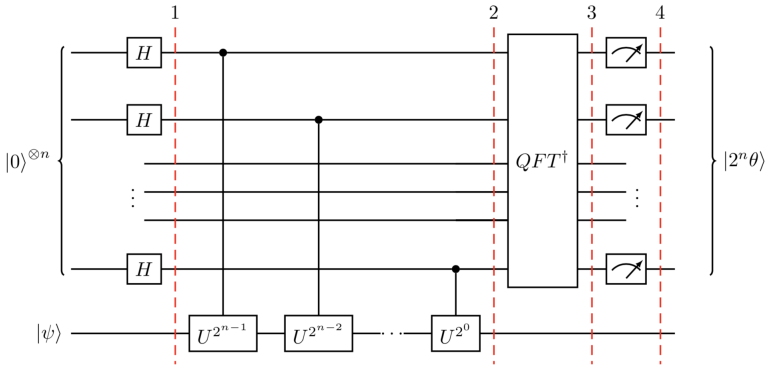


Figura 9: Circuito cuántico que realiza el algortimo QPE (Quantum Phase Estimation) sobre una serie de qubits.

Implementación en Qiskit

```
In [2]: from qiskit import QuantumCircuit

        def a_x_mod15(a, x):
            if a not in [2,7,8,11,13]:
                raise ValueError("'a' must be 2,7,8,11 or 13")
            U = QuantumCircuit(4)
            for iteration in range(x):
                if a in [2,13]:
                    U.swap(0,1)
                    U.swap(1,2)
                    U.swap(2,3)
                if a in [7,8]:
                    U.swap(2,3)
                    U.swap(1,2)
                    U.swap(0,1)
                if a == 11:
                    U.swap(1,3)
                    U.swap(0,2)
                if a in [7,11,13]:
                    for q in range(4):
                        U.x(q)
            U = U.to_gate()
            U.name = "%i^%i mod 15" % (a, x)
            c_U = U.control()
            return c_U
```

Implementación en Qiskit

In [3]:
```
def modular_exponentiation(given_circuit, n, m, a):
    for x in range(n):
        exponent = 2**x
        given_circuit.append(a_x_mod15(a, exponent),
                             [x] + list(range(n, n+m)))
```

In [37]:
```
def shor_program(n, m, a):
    # set up quantum circuit
    shor = QuantumCircuit(n+m, n)
    # initialize the qubits
    initialize_qubits(shor, n, m)
    shor.barrier()
    # apply modular exponentiation
    modular_exponentiation(shor, n, m, a)
    shor.barrier()
    # apply inverse QFT
    apply_iqft(shor, range(n))
    # measure the first n qubits
    shor.measure(range(n), range(n))
    return shor
n = 4; m = 4; a = 7
mycircuit = shor_program(n, m, a)
mycircuit.draw(output='text')
```

Implementación con Qiskit

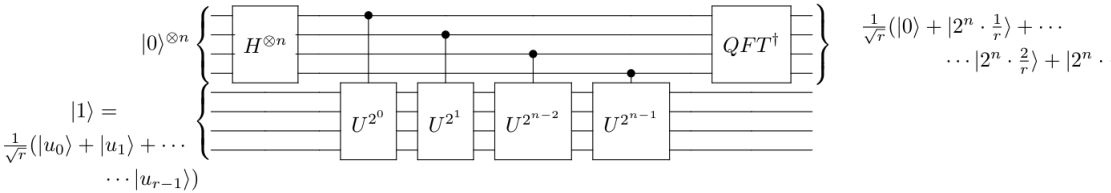


Figura 10: Algoritmo de Shor para la factorización de un número visto desde la estructura que propone Qiskit.

Resultados

Algortimo	Número a factorizar	IBM Device	Lanzamientos	Repeticiones
CCGN	21	-	-	1000
Shor (Local)		Vigo	1000	
Shor (IBM Q)				

Tabla 4: Parámetros de entrada de cada algoritmo de factorización.

Resultados

Algoritmo	Promedio (ms)	σ (ms)
Clasico	0.056	0.0096
Shor	0.036	0.0153
Shor IBM	0.024	0.0102

Tabla 5: Promedio y desviación estandar de cada algoritmo de factotización ejecutados en una computadora clásica y cuántica.

Resultados

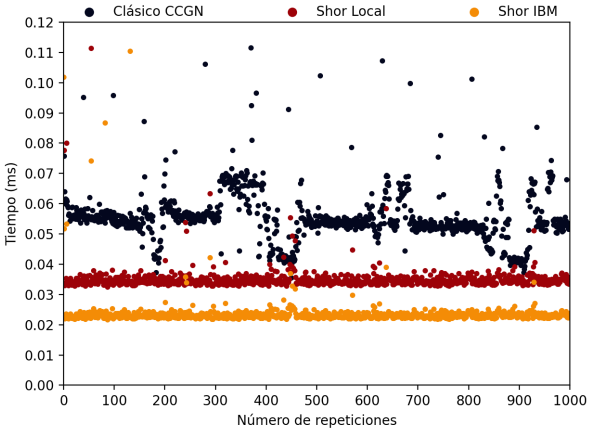


Figura 11: Comparación de los tiempos de procesamiento de los códigos ejecutados en una computadora clásica y en una computadora cuántica.

Conclusiones y discusión

Figura 12: Lista de dispositivos disponibles al público desde *IBM Q Experience*

Referencias

Agrios. *Introducción a La Introducción a La Teoría de Números.*, volume 3. 2003.

G. P. Berman, G. D. Doolen, G. V. López, and V. I. Tsifrinovich. Nonresonant effects in the implementation of the quantum Shor algorithm. *Physical Review A - Atomic, Molecular, and Optical Physics*, 61(4):7, 2000.

Vicente Moret Bonillo. Principios Fundamentales De Computación Cuántica. *univerasidad de A Coruña*, pages 1–181, 2013.

Luanne S. Cohen and Tanya Wendling. Técnicas de diseño. *Técnicas de diseño*, pages 15–18, 1998.

Edward Gerjuoy. Shor's factoring algorithm and modern cryptography. An illustration of the capabilities inherent in quantum computers. *American Journal of Physics*, 73(6):521–540, 2005.

F. Ghisi and S. V. Ulyanov. The information role of entanglement and interference operators in Shor quantum algorithm gate dynamics. *Journal of Modern Optics*, 47(12):2079–2090, 2000.

Daniel Koch, Saahil Patel, Laura Wessing, and Paul M. Alsing. Fundamentals In Quantum Algorithms: A Tutorial Series Using Qiskit Continued. 2020.

Samuel J. Lomonaco and Louis H. Kauffman. A continuous variable Shor algorithm. pages 97–108, 2005.

Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.

D E Ingenieros D E Telecomunicación. Números primos especiales y sus aplicaciones criptográficas. 2003.

Lieven M.K. Vandersypen, Matthias Breyta, Gregory Steffen, Costantino S. Yannoni, Mark H. Sherwood, and Isaac L. Chuang. Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance. *Nature*, 414(6866):883–887, 2001.

Anocha Yimsiriwattana and Samuel J. Lomonaco Jr. Distributed quantum computing: a distributed Shor algorithm. *Quantum Information and Computation II*, 5436:360, 2004.

S. S. Zhou, T. Loke, J. A. Izaac, and J. B. Wang. Quantum Fourier transform in computational basis. *Quantum Information Processing*, 16(3):1–19, 2017.