Algoritmos Cuánticos Básicos

Renato Portugal

Pesquisador Titular del
Laboratorio Nacional de Computación Científica
LNCC/MCTI

Traducción al español
Frank Acasiete

Versión original en inglés disponível em
https://arxiv.org/abs/2201.10574

Versión en español disponível em
http://qubit.lncc.br/files/ACB.pdf

1 de octubre de 2022

Resumen

La computación cuántica está evolucionando tan rápidamente que nos obliga a revisar, reescribir y actualizar la base de la teoría. Algoritmos Cuánticos Básicos revisa los primeros algoritmos cuánticos. Comenzó en 1985 con Deutsch tratando de evaluar una función en dos puntos de dominio simultáneo. Luego, Deutsch y Jozsa crearon en 1992 un algoritmo cuántico que determina si una función booleana es constante o equilibrada. Al año siguiente, Bernstein y Vazirani se dieron cuenta de que se puede usar el mismo algoritmo para encontrar una función booleana específica en el conjunto de funciones booleanas lineales. En 1994, Simon presentó un nuevo algoritmo cuántico que determina si una función es uno a uno o dos a uno exponencialmente más rápido que cualquier algoritmo clásico para el mismo problema. En el mismo año, Shor creó dos nuevos algoritmos cuánticos para factorizar números enteros y calcular logaritmos discretos, amenazando los métodos criptográficos ampliamente utilizados en la actualidad. En 1995, Kitaev describió una versión alternativa de los algoritmos de Shor que resultó útil en muchas otras aplicaciones. Al año siguiente, Grover creó un algoritmo de búsqueda cuántica cuadráticamente más rápido que su contraparte clásica. En este trabajo, todos esos notables algoritmos se describen en detalle con un enfoque en el modelo de circuito.

Índice general

Capítulo 1

Introducción

Los algoritmos cuánticos es una subárea de la computación cuántica que está evolucionando rápidamente no solo en términos de nuevos algoritmos sino también en términos de aplicaciones e implementaciones. Los algoritmos básicos son los pilares de esta nueva estructura. La construcción comenzó con un cambio en las reglas del juego. En lugar de almacenar información en bits, que toman cero o uno, podemos almacenar información en qubits, cuyo estado es una superposición de ceros y unos. Las reglas basadas en la mecánica clásica fueron reemplazadas por reglas basadas en la mecánica cuántica.

La primera idea llegó con la propuesta de Deutsch en el año 1985 de evaluar una función booleana de un bit en dos puntos simultáneamente, mediante la explotación de la superposición de ceros y unos; lo que se conoce como paralelismo cuántico. En este momento, faltaba un marco para la creación de nuevos algoritmos, que sólo se estableció en 1989 cuando Deutsch publicó un artículo sobre circuitos cuánticos, presentando las puertas cuánticas que reemplazan a las conocidas puertas clásicas, como AND, OR, NOT. En 1992, el área del algoritmo cuántico cobró impulso cuando Deutsch y Jozsa crearon un algoritmo para determinar si una función booleana está equilibrada o es constante. Este algoritmo estimuló el desarrollo de algoritmos basados en un Oráculo. Tenemos una función a nuestra disposición y necesitamos revelar una propiedad oculta de esta función. Podemos evaluar la función tantas veces como queramos, pero el objetivo es encontrar la propiedad haciendo la consulta la menor cantidad de veces posible.

Bernstein y Vazirani notaron en el año 1993 que el algoritmo de Deutsch-Jozsa podría usarse para encontrar una función booleana específica en el conjunto de funciones booleanas lineales. El algoritmo de Bernstein-Vazirani es más rápido que su contraparte clásica sin explotar el entrelazamiento. El poder del algoritmo se basa únicamente en el paralelismo cuántico.

El impulso aumentó aún más cuando Simon en el año 1994 publicó un algoritmo cuántico que distingue si una función es uno a uno o dos a uno, exponencialmente más rápido que los algoritmos clásicos para el mismo propósito. En este caso, tenemos que extender el tipo de función para tener una salida con muchos bits. Este algoritmo explota el enredo máximo. El problema de Simon tiene una aplicación interesante para encontrar un subgrupo oculto de una clase especial de grupos.

Shor alcanzó la cumbre cuando creó en el año 1994 dos célebres algoritmos cuánticos para factorizar enteros compuestos y calcular logaritmos discretos, que tienen un fuerte impacto en poder romper los métodos de criptografía utilizados en nuestra vida diaria. Los algoritmos de Shor ayudaron a poner la computación cuántica en el centro de atención. Desde entonces, el área está aumentando a un ritmo asombroso. El algoritmo de Shor también se puede formular

como un algoritmo basado en un Oráculo con una función que es periódica. El objetivo es encontrar el período evaluando la función lo menos posible. Encontrar el período es un trabajo para la transformada de Fourier, que es útil en los algoritmos clásicos a pesar de ser del orden de $O(N \log N)$, donde N es el tamaño de los datos. En el caso cuántico, la transformada de Fourier se implementa usando puertas universales del orden de $O(\log^2 N)$ y la superposición cuántica hace la magia.

Kitaev presentó en 1995 otra versión de los algoritmos de Shor tras desarrollar un algoritmo cuántico para la estimación de fase. Dado un operador unitario y uno de sus vectores propios, el algoritmo encuentra eficientemente el valor propio correspondiente; que está totalmente caracterizado por su fase. El algoritmo de Kitaev resultó útil para otras aplicaciones, como el conteo cuántico.

Grover puso el foco en las bases de datos no ordenadas y en el año 1996 creó un algoritmo cuántico que encuentra un elemento cuadráticamente más rápido que la búsqueda clásica. El algoritmo de Grover también se puede formular como un algoritmo basado en un Oráculo, con una función booleana que es constante excepto por un único punto en el dominio. El objetivo es encontrar ese punto evaluando la función lo menos posible. Cuando se escribe como un algoritmo de caja negra, está claro que el algoritmo de Grover tiene una amplia aplicabilidad.

Algoritmos cuánticos básicos describe en detalle las notables contribuciones descritas anteriormente. No hay esperanza si no usamos el lenguaje correcto: Matemáticas, más específicamente, álgebra lineal. Nociones como superposición cuántica y entrelazamiento, tienen una definición precisa cuando usamos álgebra lineal. Las medidas se describen mediante proyectores, las puertas mediante operadores unitarios y los qubits mediante vectores. Proyectores, operadores unitarios y vectores son palabras del lenguaje del álgebra lineal. Al describir algoritmos cuánticos o cualquier cosa relacionada con la cuántica, es mejor usar las matemáticas porque sin ellas, probablemente alguien diga tonterías.

Algoritmos cuánticos básicos sigue en la medida de lo posible el ordenamiento histórico, que es el orden de complejidad creciente. Nos sentimos como subiendo escalones con una altura cada vez mayor, fortaleciendo músculos y preparándonos para el desafío de comprender algoritmos cuánticos complejos. Cada capítulo es lo más independiente posible para ayudar a los lectores que están familiarizados con algunos algoritmos a saltarse algunas partes. El texto está escrito para alguien a quien le gustaría enseñar algoritmos cuánticos, no para alguien a quien le gustaría aprender por sí mismo, lo cual se desaconseja en general porque es mejor pasar la Ciencia de persona a persona directamente. Los materiales escritos son herramientas que ayudan en el proceso.

Por último, pero no menos importante, no dude en ponerse en contacto con el autor (portugal@lncc.br) si hay errores o problemas de imprecisión o falta de citas. Las sugerencias también son bienvenidas.

Agradecimientos

El autor agradece a P.H.G. Lugão por sugerir mejoras.

Capítulo 2

Circuitos cuánticos

El objetivo de este Capítulo es definir los conceptos de qubit, puerta lógica y circuito cuántico. Antes de eso, repasamos brevemente los hechos clave del álgebra lineal [2, 61] usando la notación de Dirac desde el principio. Parte de este material fue publicado en el tutorial [47]. Las referencias para esta sección son [51] y [64]. Las referencias adicionales para el álgebra lineal para la computación cuántica son el Apéndice A de [46] y la Sección 2.1 de [42].

2.1. Repaso del álgebra lineal usando la notación de Dirac

Hay varias notaciones para mostrar que una variable v es un vector, por ejemplo, $\vec{v}$, $\mathbf{v}$, etc. En computación cuántica, el más utilizado es el de Dirac: $|v\rangle$. Una secuencia de vectores se denota por $|v_0\rangle$, $|v_1\rangle$, $|v_2\rangle$ y así sucesivamente. Es muy común abusar de esta notación y denotar la misma secuencia como $|0\rangle$, $|1\rangle$, $|2\rangle$ y así sucesivamente.

La base canónica de un espacio vectorial bidimensional tiene dos vectores, que se denotan por $\{|0\rangle, |1\rangle\}$ en la notación de Dirac, donde $|0\rangle$ y $|1\rangle$ tienen la siguiente representación

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{y} \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Estos vectores tienen dos entradas o componentes, son unitarios y son ortogonales. Entonces, esta base es ortonormal. Se llama *base canónica* en álgebra lineal y *base computacional* en computación cuántica. Tenga en cuenta que $|0\rangle$ no es el vector nulo, sino el primer vector de la base canónica. Todas las entradas del vector nulo son iguales a 0. En el caso bidimensional, es

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

sin ninguna denominación especial en la notación de Dirac.

Un vector genérico en un espacio vectorial bidimensional se obtiene mediante la *combinación lineal* de los vectores base,

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle,$$

donde α y β son números complejos. Estos números son las entradas del vector $|\psi\rangle$, como se puede ver en la notación

$$|\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}.$$

El *vector dual* al vector $|\psi\rangle$ se denota por $\langle\psi|$ y se obtiene transponiendo $|\psi\rangle$ y conjugando cada entrada. Usando la ecuación anterior, obtenemos

$$\langle\psi| = \begin{pmatrix}\alpha^* & \beta^*\end{pmatrix},$$

que se puede escribir como

$$\langle\psi| = \alpha^*\langle 0| + \beta^*\langle 1|,$$

donde

$$\langle 0| = \begin{pmatrix}1 & 0\end{pmatrix} \quad \text{and} \quad \langle 1| = \begin{pmatrix}0 & 1\end{pmatrix}.$$

El vector dual $\langle\psi|$ es una matriz 1×2 y el vector $|\psi\rangle$ es una matriz 2×1. En este punto, introducimos el símbolo *dagger*, denotado por †, que es la notación para el vector transpuesto conjugado (transponer el vector y luego conjugar cada entrada o viceversa). Entonces, podemos escribir $\langle\psi| = |\psi\rangle^\dagger$ y $|\psi\rangle = \langle\psi|^\dagger$. Dos puñales, uno tras otro, funcionan como la operación de identidad.

Supongamos que $|\psi_1\rangle$ y $|\psi_2\rangle$ son vectores bidimensionales dados por

$$|\psi_1\rangle = \begin{pmatrix}\alpha\\ \beta\end{pmatrix} \quad \text{y} \quad |\psi_2\rangle = \begin{pmatrix}\gamma\\ \delta\end{pmatrix}.$$

El *producto interno* de dos vectores $|\psi_1\rangle$ y $|\psi_2\rangle$ es un número complejo denotado por $\langle\psi_1|\psi_2\rangle$ y definido como el producto matricial del vector dual $\langle\psi_1|$ por $|\psi_2\rangle$, como sigue

$$\langle\psi_1|\psi_2\rangle = \begin{pmatrix}\alpha^* & \beta^*\end{pmatrix}\begin{pmatrix}\gamma\\ \delta\end{pmatrix} = \alpha^*\gamma + \beta^*\delta.$$

En la notación de Dirac, el cálculo del producto interior se realiza distribuyendo el producto matricial sobre la suma de vectores, de la siguiente manera

$$\langle\psi_1|\psi_2\rangle = \big(\alpha^*\langle 0| + \beta^*\langle 1|\big)\cdot\big(\gamma|0\rangle + \delta|1\rangle\big) = \alpha^*\gamma\,\langle 0|0\rangle + \beta^*\delta\,\langle 1|1\rangle = \alpha^*\gamma + \beta^*\delta.$$

La *norma* del vector $|\psi_1\rangle$ se denota por $\| \, |\psi_1\rangle \, \|$ y se define como

$$\| \, |\psi_1\rangle \, \| = \sqrt{\langle\psi_1|\psi_1\rangle} = \sqrt{|\alpha|^2 + |\beta|^2},$$

donde $|\alpha|$ es el *valor absoluto* de α, es decir

$$|\alpha| = \sqrt{\alpha\cdot\alpha^*}.$$

Si $\alpha = a + b\,\mathrm{i}$, donde i es la unidad imaginaria ($\mathrm{i} = \sqrt{-1}$), a es la parte real y b es la parte imaginaria, entonces

$$|\alpha| = \sqrt{(a + b\,\mathrm{i})\cdot(a - b\,\mathrm{i})} = \sqrt{a^2 + b^2}.$$

Un número complejo α tal que $|\alpha| = 1$ se llama *número complejo unitario* y se puede escribir como $\mathrm{e}^{\mathrm{i}\theta} = \cos\theta + \mathrm{i}\sin\theta$, donde θ es un ángulo. En espacios vectoriales reales, el producto interno se llama *producto escalar* y viene dado por

$$\langle\psi_1|\psi_2\rangle = \| \, |\psi_1\rangle \| \; \| \, |\psi_2\rangle \| \cos\theta,$$

donde θ es el ángulo entre los vectores $|\psi_1\rangle$ y $|\psi_2\rangle$.

Usando estas definiciones, podemos mostrar que la base $\{|0\rangle, |1\rangle\}$ es ortonormal, es decir, los vectores $|0\rangle$ y $|1\rangle$ son ortogonales y tienen norma 1, es decir

$$\langle 0|0\rangle = 1, \quad \langle 0|1\rangle = 0, \quad \langle 1|0\rangle = 0, \quad \langle 1|1\rangle = 1.$$

Una forma algebraica de denotar la ortonormalidad y de compactar las últimas cuatro ecuaciones en una sola es

$$\langle k|\ell\rangle = \delta_{k\ell},$$

donde k y ℓ son bits y $\delta_{k\ell}$ es el *delta de Kronecker* , definido como

$$\delta_{k\ell} = \begin{cases} 1, \text{ si } k = \ell, \\ 0, \text{ si } k \neq \ell. \end{cases}$$

2.2. Qubit y superposición

La unidad de memoria básica de una computadora clásica es el *bit*, que asume el valor 0 o 1. Por lo general, el bit se implementa usando dos voltajes distintos, siguiendo la convención de que el voltaje bajo o nulo representa el bit 0 y el voltaje alto representa el bit 1 Para determinar si la salida es el bit 0 o 1 al final del cálculo, es necesario medir el voltaje.

La unidad de memoria básica de una computadora cuántica es el *qubit*, que también asume, al final del cálculo, un valor 0 o 1. El qubit se implementa utilizando una corriente eléctrica en un pequeño superconductor, siguiendo la convención de que la corriente en sentido horario representa 0 y la corriente en sentido antihorario representa 1, o al revés. La diferencia con el dispositivo clásico ocurre durante el cómputo ya que el qubit admite la coexistencia simultánea de los valores 0 y 1. Durante el cómputo, es decir, antes de la medición, el *estado* de un qubit está representado por una norma de valor 1 dos -vector dimensional y los estados de un qubit correspondientes a 0 y 1 son $|0\rangle$ y $|1\rangle$. La definición de *estado* es un vector de norma 1 en un espacio vectorial complejo dotado del producto interno presentado en la Sección anterior.[1] El estado se puede considerar como el "valor" del qubit antes de la medición. La coexistencia cuántica se representa matemáticamente mediante una combinación lineal de vectores ortonormales de la siguiente manera

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle,$$

donde α y β son números complejos que obedecen la restricción

$$|\alpha|^2 + |\beta|^2 = 1.$$

El estado del qubit es el vector $|\psi\rangle$ de norma 1 con las entradas α y β. Los números complejos α y β son las *amplitudes* del estado $|\psi\rangle$.

La coexistencia de los bits 0 y 1 no se puede implementar en un dispositivo clásico, ya que no es posible tener bajo y alto voltaje al mismo tiempo, como todos saben. En mecánica cuántica, es difícil de creer, es posible tener un sistema cuántico (generalmente microscópico) con bajo y alto voltaje al mismo tiempo. Esta coexistencia sólo puede mantenerse completamente, si el sistema cuántico está completamente aislado del entorno macroscópico circundante. Cuando medimos el sistema cuántico para determinar el valor del voltaje, el dispositivo de medición inevitablemente afecta el voltaje, generando un resultado estocástico, que es un voltaje bajo o

[1] Un espacio vectorial de dimensión finita con un producto interno es un *espacio de Hilbert* .

alto, similar al bit clásico. En otras palabras, la coexistencia solo se mantiene cuando nadie (ni nada) está tratando de determinar si el voltaje es alto o bajo. Tenga en cuenta que la mecánica cuántica es una *teoría científica*, es decir, sus leyes y resultados pueden probarse objetivamente en laboratorios. Además, las leyes y declaraciones innecesarias se descartan sumariamente. Por lo tanto, la declaración de "la coexistencia sólo se mantiene cuando el sistema está aislado"tiene consecuencias prácticas y es una declaración que se ha probado y se ha vuelto a probar durante más de 100 años, en miles de laboratorios de mecánica cuántica en todo el mundo. Por otro lado, las pruebas experimentales han descartado teorías alternativas que son más aceptables clásicamente, que podrían ayudar a comprender o visualizar la superposición cuántica.

Desde un punto de vista computacional, tenemos un qubit en superposición y usamos esta característica en un circuito. Por ejemplo, el circuito

$$|\psi\rangle \longrightarrow \boxed{\;\;} \Longrightarrow 0 \text{ o } 1$$

nos dice que el "valorïnicial del qubit es $|\psi\rangle$ y esta información se transmite sin cambios de izquierda a derecha hasta que se realiza una medición, como lo muestra el *medidor* (la pantalla de un voltímetro). Las salidas de la medición son 0 o 1. La información clásica se transmite por un cable doble. Si el estado del qubit es $|\psi\rangle = |0\rangle$, una medición necesariamente generará 0 y si el estado es $|1\rangle$, una medición necesariamente generará 1. En el caso general, si el estado es $\alpha|0\rangle + \beta|1\rangle$, una medición arrojará 0 con probabilidad $|\alpha|^2$ o 1 con probabilidad $|\beta|^2$, como se muestra en el circuito

$$\alpha|0\rangle + \beta|1\rangle \longrightarrow \boxed{\;\;} \Longrightarrow \begin{cases} 0, \text{ con probabilidad } |\alpha|^2, \\ 1, \text{ con probabilidad } |\beta|^2. \end{cases}$$

La salida se puede representar mediante un histograma de la distribución de probabilidades. Es importante repetir que α y β se llaman *amplitudes* del estado $\alpha|0\rangle + \beta|1\rangle$ y son números que pueden ser negativos y tener parte imaginaria. Por otro lado, $|\alpha|^2$ y $|\beta|^2$ son números reales positivos en el intervalo $[0, 1]$ y se denominan probabilidades. Un intercambio descuidado entre amplitudes y probabilidades crea errores imperdonables.

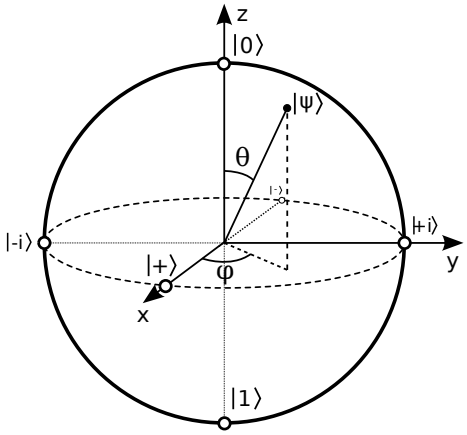


Figura 2.1: Esfera de Bloch y la ubicación de los estados $|0\rangle$, $|1\rangle$, $|\pm\rangle$, y $|\pm\mathrm{i}\rangle$. Un estado arbitrario $|\psi\rangle$ es mostrado con los ángulos esféricos θ y φ.

El estado de un qubit se puede caracterizar por dos ángulos θ y φ de la siguiente manera

$$|\psi\rangle = \cos\frac{\theta}{2}\,|0\rangle + \mathrm{e}^{\mathrm{i}\varphi}\sin\frac{\theta}{2}\,|1\rangle,$$

donde $0 \leq \theta \leq \pi$ y $0 \leq \varphi < 2\pi$. Esta notación muestra que existe una correspondencia uno a uno entre el conjunto de estados de un qubit y los puntos en la superficie de una esfera de radio 1, llamada *esfera de Bloch*. Los ángulos θ y φ son ángulos esféricos que describen la ubicación del estado $|\psi\rangle$, como se muestra en la Fig. 2.1. Un punto en la esfera de Bloch se describe mediante un vector tridimensional con entradas reales

$$\begin{pmatrix} \sin\theta\cos\varphi \\ \sin\theta\sin\varphi \\ \cos\theta \end{pmatrix}.$$

Las ubicaciones en la esfera de Bloch de los estados $|0\rangle$ y $|1\rangle$, cuyos ángulos esféricos son $(\theta, \varphi) = (0, 0)$ y $(\theta, \varphi) = (\pi, 0)$, se señalan en la Fig. 2.1. Las ubicaciones de los estados

$$\begin{aligned} |\pm\rangle &= \frac{|0\rangle \pm |1\rangle}{\sqrt{2}}, \\ |\pm \mathrm{i}\rangle &= \frac{|0\rangle \pm \mathrm{i}|1\rangle}{\sqrt{2}} \end{aligned}$$

se determinan después de averiguar los ángulos esféricos, que son $(\theta, \varphi) = (\pi/2, \pi/2 \mp \pi/2)$ y $(\theta, \varphi) = (\pi/2, \pm\pi/2)$, respectivamente.[2] Luego, sus ubicaciones son las intersecciones del eje x con la esfera de Bloch y el eje y con la esfera de Bloch.

Si tenemos un estado arbitrario de 1 qubit $\alpha|0\rangle + \beta|1\rangle$ y queremos encontrar los ángulos esféricos θ y φ, lo primero que debemos hacer es escribir α y β como $r_1 \mathrm{e}^{\mathrm{i}\varphi_1}$ y $r_2 \mathrm{e}^{\mathrm{i}\varphi_2}$, respectivamente; donde $r_1 = |\alpha|$ y $r_2 = |\beta|$. Ahora multiplicamos el estado por $\mathrm{e}^{-\mathrm{i}\varphi_1}$ para obtener $r_1|0\rangle + \mathrm{e}^{\mathrm{i}(\varphi_2 - \varphi_1)} r_2|1\rangle$. Luego, tomamos $\varphi = \varphi_2 - \varphi_1$ y $\theta = 2 \arccos r_1$. Tenga en cuenta que $r_2 = \sin(\theta/2)$ porque $r_1^2 + r_2^2 = 1$. No hay problema en multiplicar el estado por un número complejo unitario como $\mathrm{e}^{-\mathrm{i}\varphi_1}$ porque en mecánica cuántica dos estados cuánticos que difieren en un factor global se consideran iguales. El factor global debe ser un número complejo unitario y suele llamarse fase global.

2.3. Puertas de un solo qubit

Una puerta de un solo qubit es una matriz cuadrada unitaria bidimensional. Una matriz U es unitaria si al multiplicar U por un vector de norma arbitrario da como resultado un vector de norma 1. Formalmente, supongamos que $|\psi'\rangle = U|\psi\rangle$, donde $|\psi\rangle$ es un vector bidimensional de norma 1. Si U es una matriz unitaria, entonces $|\psi'\rangle$ tendrá norma 1. Por ejemplo, la matriz de Hadamard

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

es unitaria. Por lo tanto, la multiplicación de H por los vectores base tiene que dar como resultado vectores de norma 1. En realidad,

$$H|0\rangle = H \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle.$$

[2] No intente entender $\pm$ simultáneamente porque nuestra mente entra en un estado no deseado, superpuesto, enredado. Por favor, se debe abordar primero el signo superior de todas las expresiones. Luego, se dirige al signo inferior.

Tenga en cuenta que el vector resultante tiene norma de valor 1. Denotamos este vector por $|+\rangle$, es decir

$$|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle.$$

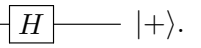
Multiplicando H por $|1\rangle$ se obtiene el vector $|-\rangle$ definido como

$$|-\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle,$$

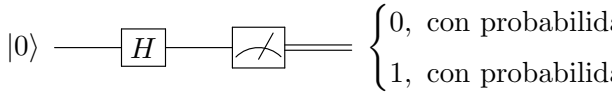
que también tiene norma de vaor 1. Estos cálculos son importantes porque necesitamos conocer la salida de la puerta. Si la entrada es $|0\rangle$, la salida será $|+\rangle$. Si la entrada es $|1\rangle$, la salida será $|-\rangle$. Si la entrada es una superposición $\alpha|0\rangle + \beta|1\rangle$, la salida será la superposición de $|+\rangle$ y $|-\rangle$ con las mismas amplitudes, $\alpha|+\rangle + \beta|-\rangle$, porque usamos la propiedad de linealidad de la puerta, es decir, en lugar de pensar que H es una matriz, usamos que H es un operador lineal y si se aplica H a una combinación lineal de los vectores $|0\rangle$ y $|1\rangle$ con amplitudes α y β, el resultado es una combinación lineal de $H|0\rangle$ y $H|1\rangle$ con las mismas amplitudes α y β. Podemos evitar este punto de vista abstracto, pero cuando multiplicamos una matriz por una suma de vectores, tenemos que distribuir la multiplicación entre los vectores.

Comprobar que la multiplicación de H por los vectores de la base ortonormal da como resultado vectores de norma 1 no es suficiente para demostrar que H es una matriz unitaria. También es necesario demostrar que los vectores resultantes son ortogonales, es decir, demostrar que $\langle -|+\rangle = 0$. En este punto, es más fácil comprobar que H es unitario calculando $HH^\dagger$, donde $H^\dagger$ se obtiene transponiendo H y conjugando cada entrada. Si el resultado es la matriz identidad, H será unitario. Para que conste, $H^\dagger$ se llama la *transposición hermítica* de H.

Un *circuito cuántico* es una representación gráfica de un algoritmo cuántico. El qubit de entrada está a la izquierda y la información (el estado del qubit) se transmite sin cambios de izquierda a derecha hasta que se encuentre con una puerta lógica. La puerta recibe la entrada de la izquierda, luego procesa la información y el resultado sale por la derecha y continua su curso. El procesamiento de la puerta se realiza multiplicando la matriz unitaria que representa la puerta por el vector que representa el estado del qubit. Por ejemplo, la expresión $|+\rangle = H|0\rangle$ está representado por el siguiente circuito:



La entrada es el vector $|0\rangle$, que el cable transmite sin cambios a H, que actúa sobre la entrada y la transforma en $|+\rangle$, que luego se transporta hacia la derecha. La acción de la puerta se calcula multiplicando H por $|0\rangle$. Por lo tanto, el resultado del cálculo es $|+\rangle$. Si al final del cálculo realizamos una medición, el circuito es



El circuito muestra que la salida de la medida del qubit, cuyo estado era $|+\rangle$, es 0 con probabilidad 1/2 o 1 con la misma probabilidad. La Fig. 2.2 muestra el histograma de la distribución de probabilidades generada en Qiskit.[3]. Un ejemplo más sencillo que el anterior es la puerta X,

[3]Qiskit es un software de código abierto para ejecutar programas en computadoras cuánticas de la IBM.

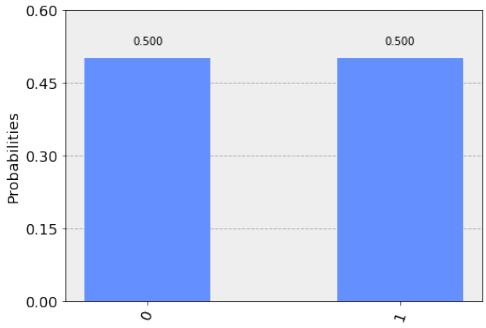
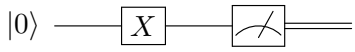


Figura 2.2: Histograma de la distribución de probabilidades generada al medir un qubit cuyo estado es $|+\rangle$.

definida como

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

X es la puerta NOT cuántica porque $|1\rangle = X|0\rangle$ y $|0\rangle = X|1\rangle$. Podemos verificar estas ecuaciones mediante la multiplicación matricial de X con $|0\rangle$ y $|1\rangle$. De forma más compacta, podemos escribir $|j \oplus 1\rangle = X|j\rangle$, donde $\oplus$ es la operación XOR o suma módulo 2. Debido a esto, la puerta X también se representa como $\oplus$. Un circuito que usa la puerta X es



1 con probabilidad 1.

Ahora podemos aumentar la complejidad. ¿Cómo generar una superposición tal que las amplitudes de $\alpha|0\rangle + \beta|1\rangle$ sean diferentes y no nulas? Por ejemplo, ¿cómo generar un estado $\alpha|0\rangle + \beta|1\rangle$ tal que $|\alpha|^2 = 25\,\%$ y $|\beta|^2 = 75\,\%$ antes de la medición? La respuesta es usar la puerta de 1 qubit más general, cuya expresión algebraica es

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

Después de aplicar $U(\theta, 0, 0)$ sobre $|0\rangle$, obtenemos

$$U(\theta, 0, 0)|0\rangle = \cos\frac{\theta}{2}|0\rangle + \sin\frac{\theta}{2}|1\rangle.$$

Debemos elegir $\theta = 2\pi/3$, porque $\cos^2(\pi/3) = 1/4$. El ejemplo anterior que usa la puerta de Hadamard se puede reproducir tomando $\theta = \pi/2$ y $\lambda = \pi$ porque $H = U(\pi/2, 0, \pi)$.

$U(\theta, \phi, \lambda)$ es una puerta comodín porque reemplaza a todas las demás puertas de 1 qubit. Por ejemplo, tres puertas útiles obtenidas de U son

$$U\left(\theta, -\frac{\pi}{2}, \frac{\pi}{2}\right) = R_x(\theta) = \begin{bmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix},$$

$$U(\theta, 0, 0) = R_y(\theta) = \begin{bmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix},$$

$$U(0, 0, \lambda) = e^{i\lambda/2}R_z(\lambda) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\lambda} \end{bmatrix}.$$

donde R_x, R_y y R_z son los operadores que giran la esfera de Bloch sobre los ejes x, y y z, respectivamente. Desafortunadamente, U es demasiado perfecta para ser verdad, ya que sería posible elegir θ tan pequeño como queramos al costo de una sola puerta. Los errores eventualmente impedirían obtener buenos resultados después de elegir ángulos muy pequeños.

Para determinar la complejidad computacional de un algoritmo, tenemos que restringirnos a un conjunto finito de puertas de un solo qubit. Las puertas de 1 qubit más importantes son

$$I_2 \;=\; \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}, \quad X \;=\; \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}, \quad Y \;=\; \begin{bmatrix}0 & -\mathrm{i}\\ \mathrm{i} & 0\end{bmatrix}, \quad Z \;=\; \begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}$$

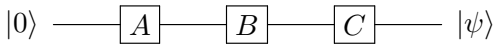
conocidas como *matrices de Pauli*,

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}, \quad S = \begin{bmatrix}1 & 0\\ 0 & \mathrm{i}\end{bmatrix}, \quad S^\dagger = \begin{bmatrix}1 & 0\\ 0 & -\mathrm{i}\end{bmatrix}, \quad T = \begin{bmatrix}1 & 0\\ 0 & \mathrm{e}^{\frac{\mathrm{i}\pi}{4}}\end{bmatrix}, \quad T^\dagger = \begin{bmatrix}1 & 0\\ 0 & \mathrm{e}^{-\frac{\mathrm{i}\pi}{4}}\end{bmatrix}$$

conocida como puerta de Hadamard, puerta de fase, su puerta conjugada, puerta $\pi/8$ o puerta T y su conjugada.[4] los números complejos $\mathrm{e}^{\pm\frac{\mathrm{i}\pi}{4}}$ son iguales a

$$\mathrm{e}^{\pm\frac{\mathrm{i}\pi}{4}} = \frac{1 \pm \mathrm{i}}{\sqrt{2}}.$$

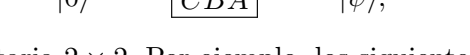
Todo circuito cuántico sin medición, tiene una expresión algebraica equivalente. Por ejemplo, si A, B, C son puertas de un solo qubit, el circuito



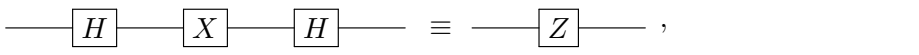
es equivalente a la expresión algebraica

$$|\psi\rangle \;=\; C \cdot B \cdot A \cdot |0\rangle,$$

donde $\cdot$ es el producto de matrices, que generalmente se omite. Tenga en cuenta que la expresión algebraica equivalente al circuito tiene el orden inverso. Por lo tanto, el último circuito también se puede escribir como



donde CBA es una matriz unitaria 2×2. Por ejemplo, los siguientes circuitos son equivalentes:



porque $Z = HXH$. La expresión algebraica equivalente se puede utilizar para simplificar el circuito y predecir su salida.

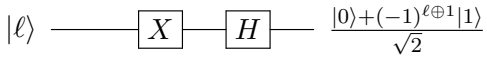
En computación cuántica, tenemos que aprovechar la notación computacional clásica. Por ejemplo, piensa que ℓ en $|\ell\rangle$ es un bit clásico. Entonces, $|\ell\rangle$ representa tanto $|0\rangle$ como $|1\rangle$. Esta notación se usa en $|\ell \oplus 1\rangle = X|\ell\rangle$. Del mismo modo, tenemos $(-1)^\ell|\ell\rangle = Z|\ell\rangle$ y $(-1)^\ell \mathrm{i}\,|\ell \oplus 1\rangle = Y|\ell\rangle$. Esas expresiones se utilizan para determinar la salida de las puertas X, Y y Z, cuando la entrada es un vector de la base computacional. Para la puerta $U(0, 0, \lambda)$, tenemos $\mathrm{e}^{\mathrm{i}\lambda\ell}|\ell\rangle =$

[4] La puerta $T^\dagger$ es de hecho la puerta transpuesta-conjugada, pero como T es diagonal, $T^\dagger$ es simplemente la puerta conjugada.

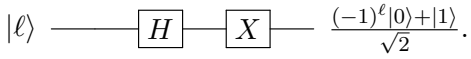
$U(0,0,\lambda)|\ell\rangle$, que incluye S, T y sus puertas conjugadas como subcasos. Todas esas puertas no crean superposición. La única que lo hace es la puerta de Hadamard, cuya salida es

$$\frac{|0\rangle + (-1)^{\ell}|1\rangle}{\sqrt{2}} = H|\ell\rangle.$$

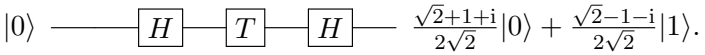
Ahora estamos listos para demostrar que



y



Con un poco más de esfuerzo, demostramos que



Implementación en las computadoras cuánticas de la IBM.

En este punto, es una buena idea utilizar el *composer* de la IBM. Después de iniciar sesión en el sitio web de la IBM[5] (es necesario registrarse), debemos iniciar el composer haciendo clic en Launch Composer. El composer de la IBM es amigable porque podemos arrastrar las puertas disponibles al circuito. Mantengámoslo en un uso básico en este momento. La fig. 2.3 muestra un circuito con la puerta de Hadamard seguida de su medición. Simplemente arrastramos H y lo soltamos en el primer cable del circuito, luego arrastramos el medidor y lo soltamos después de H. La flecha del medidor muestra que la salida se redirige a un registrador clásico auxiliar en la parte inferior del circuito.

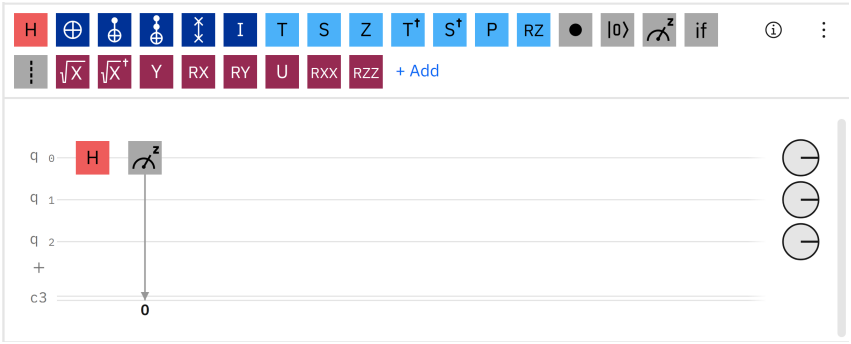


Figura 2.3: Ejemplo de un circuito con una puerta Hadamard y un medidor (Reimpresión cortesía de IBM Corporation ©)

Después de que el circuito esté listo, hacemos clic en Setup and run y luego tenemos dos opciones: (1) Ejecutar el circuito en una computadora cuántica seleccionando uno de los sistemas cuánticos disponibles, o (2) simular el circuito seleccionando un simulador. Por lo general, es mejor comenzar con la segunda opción. Seleccionamos *ibm_qasm_simulator* como proveedor, luego seleccionamos el número de *shots* y luego hacemos clic en Run en la parte inferior. La Fig. 2.4 muestra el resultado de una ejecución. El resultado 000 se obtuvo 503 veces y el 001

[5] https://quantum-computing.ibm.com/

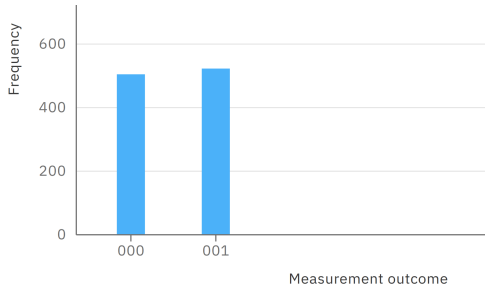


Figura 2.4: Salida del circuito con una puerta Hadamard y un medidor (Reimpresión cortesía de IBM Corporation ©)

se obtuvo 521 veces de 1024 shots. Los dos primeros bits deben descartarse porque se refieren a qubits que no se han utilizado (qubits q_2 y q_1). La salida usa el orden $q_2q_1q_0$, diferente del orden adoptado en la mayoría de los libros de texto sobre computación cuántica. Como podemos ver, el composer ejecuta el experimento varias veces (hasta 8192) y muestra el histograma de la distribución de frecuencia acumulada. En el caso de la puerta de Hadamard, el resultado más probable es del 50 % cada uno, pero los resultados cercanos al 50 % tienen probabilidades no despreciables y ocurren con frecuencia. Para obtener resultados más cercanos al 50 %, tenemos que aumentar el número de shots.

Si elegimos ejecutar en una computadora cuántica, el circuito se pondrá en fila y puede llevar mucho tiempo. Podemos comprobar el tamaño de la fila cuando estamos seleccionando el sistema. La salida de la computadora cuántica suele ser diferente a la del simulador porque los errores degradan el resultado. Aumentar el número de shots no garantiza que la distribución de probabilidades tienda a la distribución correcta, y es importante verificar la corrección del circuito a través del simulador antes de ejecutar el sistema cuántico.

2.4. Estados cuánticos y entrelazamiento

El estado de dos qubits se describe mediante un vector de norma 1 que pertenece a un espacio vectorial de cuatro dimensiones porque hay cuatro posibles resultados después de medir los qubits: 00, 01, 10, 11. El primer bit se refiere al primer qubit y el segundo bit al segundo qubit, como es habitual en los libros de texto sobre computación cuántica, el bit menos significativo está a la derecha.

Siguiendo las leyes de la mecánica cuántica, existe una correspondencia uno a uno entre la base canónica y los posibles resultados de la medición de la siguiente manera:

$$|00\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad |01\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \quad |10\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \quad |11\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

En el caso clásico, el estado de dos bits es 00 o 01 o 10 u 11, exclusivamente. En el caso cuántico, el estado de dos qubits es la combinación lineal

$$|\psi\rangle = a_0|00\rangle + a_1|01\rangle + a_2|10\rangle + a_3|11\rangle,$$

donde a_0, a_1, a_2 y a_3 son números complejos. Cuando el estado de dos qubits es $|\psi\rangle$, el resultado de una medición en la base computacional es 00 o 01 o 10 u 11, exclusiva y estocásticamente;

no hay forma de predecir el resultado de la medición incluso sabiendo $|\psi\rangle$. Sin embargo, si conocemos $|\psi\rangle$, entonces conocemos las probabilidades de los resultados, que son

$$\begin{aligned}
\text{prob}(00) &= \left|\langle 00|\psi\rangle\right|^2 = |a_0|^2, \\
\text{prob}(01) &= \left|\langle 01|\psi\rangle\right|^2 = |a_1|^2, \\
\text{prob}(10) &= \left|\langle 10|\psi\rangle\right|^2 = |a_2|^2, \\
\text{prob}(11) &= \left|\langle 11|\psi\rangle\right|^2 = |a_3|^2.
\end{aligned}$$

La suma de estas probabilidades es 1. Si no conocemos el estado $|\psi\rangle$, una sola medición no permite determinar $|\psi\rangle$, es decir, no podemos encontrar las amplitudes a_0, a_1, a_2 y a_3. Hay un teorema importante en la mecánica cuántica conocido como el *teorema de no clonación* [20, 44, 63].

Teorema 2.1. (No clonación) Usando operadores unitarios, es imposible hacer una copia idéntica de un estado cuántico arbitrario desconocido que está disponible para nosotros.

Este teorema restringe severamente cualquier posibilidad de determinar $|\psi\rangle$ a través de mediciones. Sin embargo, si podemos generar $|\psi\rangle$ una y otra vez, por ejemplo, a través de un circuito; podemos repetir todo el proceso varias veces y obtener una aproximación para prob(00), prob(01), prob(10) y prob(11). Por ejemplo, repitiendo 1000 veces, podemos determinar estas probabilidades con dos dígitos. Desafortunadamente, aún no podemos determinar $|\psi\rangle$ exactamente porque conocer $|a_0|^2$ no nos permite determinar a_0 exactamente. Puede parecer que esto no es importante; falso. Consideremos un ejemplo crítico. Suponemos que

$$\text{prob}(00) = \frac{1}{2}, \qquad \text{prob}(01) = 0, \qquad \text{prob}(10) = 0, \qquad \text{prob}(11) = \frac{1}{2}.$$

Tenemos al menos dos posibilidades para $|\psi\rangle$:

$$|\psi_1\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}} \qquad \text{y} \qquad |\psi_2\rangle = \frac{|00\rangle - |11\rangle}{\sqrt{2}}.$$

Tenga en cuenta que $|\psi_1\rangle$ y $|\psi_2\rangle$ son ortogonales. Esto demuestra que podemos cometer un grave error. No sabemos si dos circuitos son equivalentes cuando sus distribuciones de probabilidades son las mismas.

En este punto, la siguiente pregunta es relevante: Supongamos que conocemos $|\psi\rangle$, ¿es posible determinar el estado de cada qubit? La respuesta es "depende de $|\psi\rangle$". Si $|\psi\rangle$ es uno de los estados de la base computacional, entonces conocemos el estado de cada qubit. Por ejemplo, suponga $|\psi\rangle = |10\rangle$. Tenemos que factorizar $|\psi\rangle$ como

$$|10\rangle \;=\; |1\rangle|0\rangle \;=\; |1\rangle \otimes |0\rangle,$$

donde $\otimes$ se llama *producto de Kronecker*. Cuando la factorización es exitosa, conocemos el estado de cada qubit. En este caso, el estado del primer qubit es $|1\rangle$ y el estado del segundo es $|0\rangle$. Cuando escribimos $|1\rangle|0\rangle$, se asume implícitamente el producto de Kronecker.

El producto de Kronecker[6] de dos vectores o dos matrices se define de la siguiente manera. Sea A una matriz $m \times n$ y B una matriz $p \times q$. Tenemos el siguiente resultado,

$$A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ & \ddots & \\ a_{m1}B & \cdots & a_{mn}B \end{bmatrix}.$$

[6]Hay una formulación más abstracta del producto de Kronecker llamada *producto tensorial*. En este trabajo, usamos estos términos indistintamente. La notación $A \otimes B$ se dice "A tensor B"; sin embargo, los términos *tensor* y *producto de tensores* no tienen relación con tensores (una generalización de matrices) o producto de tensores.

El resultado es una matriz $mp \times nq$. El producto de Kronecker de los vectores $|1\rangle$ y $|0\rangle$ se calcula viendo estos vectores como matrices 2×1 y viene dado por

$$|1\rangle \otimes |0\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ 1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = |10\rangle.$$

Tenga en cuenta que el producto de Kronecker no es conmutativo. Por ejemplo, $|1\rangle \otimes |0\rangle \neq |0\rangle \otimes |1\rangle$. Un consejo importante es nunca cambiar el orden del producto de Kronecker.

De las leyes de la mecánica cuántica, si el estado de un qubit es $|\psi_1\rangle$ y el estado de un segundo es $|\psi_2\rangle$, entonces el estado $|\psi\rangle$ del sistema compuesto de dos qubits que interactúan, será inicialmente

$$|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle.$$

Siempre podemos obtener el estado del sistema compuesto cuando conocemos los estados de las partes. Sin embargo, el proceso inverso no es posible en general. Por ejemplo, suponemos que el estado de dos qubits es

$$|\psi\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}.$$

Queremos encontrar estados de 1 qubit $|\psi_1\rangle = \alpha|0\rangle + \beta|1\rangle$ y $|\psi_2\rangle = \gamma|0\rangle + \delta|1\rangle$ tales que

$$(\alpha|0\rangle + \beta|1\rangle) \otimes (\gamma|0\rangle + \delta|1\rangle) = \frac{|00\rangle + |11\rangle}{\sqrt{2}}.$$

Desarrollando el lado izquierdo, obtenemos el siguiente sistema de ecuaciones:

$$\alpha\gamma = \frac{1}{\sqrt{2}}, \quad \alpha\delta = 0, \quad \beta\gamma = 0, \quad \beta\delta = \frac{1}{\sqrt{2}}.$$

Dado que este sistema no tiene solución, el estado de 2 qubits $|\psi\rangle$ no se puede escribir como el producto de Kronecker de estados de 1 qubit. En mecánica cuántica, un sistema cuántico compuesto puede tener un estado definido $|\psi\rangle$ mientras que partes del sistema no tienen un estado definido.

Un estado cuántico de un sistema compuesto que no se puede factorizar en términos del producto de Kronecker se llama *estado entrelazado*. Los estados entrelazados son muy importantes en la computación cuántica porque sin ellos la potencia computacional de la computadora cuántica se vería gravemente afectada. Sin embargo, tenga en cuenta que la presencia de entrelazamiento en un algoritmo cuántico no garantiza que este algoritmo sea más eficiente que un algoritmo clásico.

El término "estado definido" $|\psi\rangle$ en mecánica cuántica significa *estado puro*. Un estado de un sistema cuántico se llama estado puro si estamos 100 % seguros de que el sistema está descrito por un vector $|\psi\rangle$ de norma 1. Por otro lado, si no estamos 100 % seguros, es decir, si sabemos que el estado del sistema es $|\psi_1\rangle$ con probabilidad $0 < p < 1$ o $|\psi_2\rangle$ con probabilidad $1 - p$, entonces el estado es *mixto* y se representa por un *conjunto* o una matriz ρ tal que $\mathrm{Tr}(\rho) = 1$. Cuando el estado de un sistema cuántico compuesto está entrelazado, el estado de cualquier subsistema es siempre un estado mixto.

Podemos generalizar la discusión de esta Sección a n qubits, donde $n \geq 1$. La base computacional tiene 2^n vectores, cada vector con 2^n entradas,

$$|0\cdots00\rangle = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad |0\cdots01\rangle = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix}, \quad \cdots, \quad |1\cdots11\rangle = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix}.$$

Tenga en cuenta que el número binario dentro del ket, por ejemplo, $0\cdots0$ en $|0\cdots0\rangle$, tiene n bits y el estado en sí es el producto de Kronecker de n estados de 1 qubit. El número binario $0\cdots0$ se puede escribir en notación decimal como $|0\cdots0\rangle \to |0\rangle$. Cada número binario dentro de los *kets* se puede escribir en notación decimal como

$$|0\cdots00\rangle \to |0\rangle, \quad |0\cdots01\rangle \to |1\rangle, \quad |0\cdots10\rangle \to |2\rangle, \quad ..., \quad |1\cdots11\rangle \to |2^n - 1\rangle.$$

Un estado genérico $|\psi\rangle$ pertenece a un espacio vectorial 2^n-dimensional. Después tenemos,

$$|\psi\rangle = a_0\,|0\rangle + a_1\,|1\rangle + a_2\,|2\rangle + \cdots + a_{2^n-1}\,|2^n - 1\rangle,$$

donde

$$|a_0|^2 + |a_1|^2 + |a_2|^2 + \cdots + |a_{2^n-1}|^2 = 1.$$

Después de una medición de todos los qubits, obtenemos una cadena de n-bits estocásticos. El resultado es la cadena de n-bits $0\cdots00$ con probabilidad $|a_0|^2$, o la cadena de n-bits $0\cdots01$ con probabilidad $|a_1|^2$, y así sucesivamente. Tenemos un *espacio de muestra* que comprende esas cadenas de n-bits y una distribución de probabilidades dada por $\text{prob}(\ell) = |a_\ell|^2$, donde ℓ es una cadena de n-bits. El resultado de la medición es una variable aleatoria que toma un valor ℓ en este espacio de muestra con probabilidad $\text{prob}(\ell)$.

Como hemos dicho antes, un vector de la base computacional de n qubits se puede escribir como el producto de Kronecker de vectores n de 1 qubit. Por ejemplo, para $n = 3$ qubits, podemos obtener el segundo vector de la base computacional usando el producto de Kronecker como

$$|0\rangle \otimes |0\rangle \otimes |1\rangle \;=\; \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} \;=\; \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \;=\; |001\rangle.$$

En la notación decimal, $|0\rangle$ se puede confundir con el estado de 1 qubit. Para evitar confusiones, tenemos que saber cuál es el número de qubits. Por ejemplo, si $|0\rangle$ se refiere al estado de 3 qubits en notación decimal, entonces en binario tenemos $|000\rangle$.

En esta Sección, hemos definido estados entrelazados. No es una buena idea tratar de entender este concepto sin usar las matemáticas desde el principio. Aprender la definición básica de entrelazamiento es similar a aprender qué es un número primo en la aritmética básica. Supongamos que no sabemos nada y vamos a aprender las bases de la aritmética empezando por la suma. Después de asistir a clases, hacer muchos ejercicios, memorizar tablas, dominamos el concepto

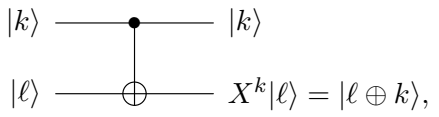
de suma. El siguiente paso es la multiplicación. Tenemos que pasar por el mismo calvario de clases, ejercicios y memorización de tablas, pero eventualmente, dominamos el concepto de multiplicación. Entonces estamos listos para aprender el proceso inverso, es decir, la factorización. Aprendemos que los factores de 15 son 3 y 5. Ahora probamos con 17. Tenemos que encontrar dos números positivos estrictamente menores que 17 para que cuando multipliquemos esos números obtengamos 17; comprobamos que no existen tales números. Entonces estamos listos para comprender el concepto de números primos, que es muy importante en matemáticas. Si comprendemos los números primos, entonces estamos preparados para comprender los estados entrelazados. Pero esta vez tenemos que aprender la suma directa de vectores, luego el producto de vectores de Kronecker, y luego tratamos de factorizar un vector. Es decir, tenemos que encontrar dos vectores $|\psi_1\rangle$ y $|\psi_2\rangle$ que tengan menos entradas que $|\psi\rangle$ para que $|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$. Los estados entrelazados son los vectores irreducibles de los sistemas cuánticos compuestos en términos del producto de Kronecker. Un estado de un solo qubit no está entrelazado porque un solo qubit no es un sistema compuesto. Necesitamos al menos dos qubits. Hay un largo camino por delante, por supuesto, pero dos qubits es un buen punto de partida. En la siguiente Sección, veremos cómo producir un estado entrelazado en una computadora cuántica.

2.5. Puertas cuánticas de dos qubits

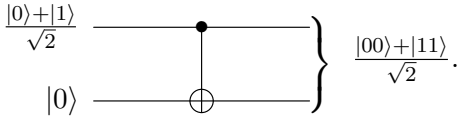
La puerta de 2 qubits más importante es la CNOT o puerta NOT controlada, también representada por $C(X)$ o CX. se define como

$$\text{CNOT}\,|k\rangle|\ell\rangle = |k\rangle X^k|\ell\rangle,$$

y está representado por el circuito



donde k y ℓ son bits. El estado del primer qubit (*control*) no cambia después de aplicar CNOT. El estado del segundo qubit (*objetivo*) cambia sólo si el bit k es 1. En este caso, la salida es $X|\ell\rangle = |\ell \oplus 1\rangle$. Si $k = 0$ entonces $X^0 = I_2$ y $I_2|\ell\rangle = |\ell\rangle$, donde I_2 es la matriz identidad 2×2. Hemos definido CNOT mostrando su acción sobre los vectores de la base computacional. En álgebra lineal, esta definición es completa, porque para conocer la acción de CNOT sobre un vector arbitrario, que es una combinación lineal de vectores de la base computacional, usamos la linealidad de esta puerta. Por ejemplo, en el siguiente circuito, la primera entrada está en superposición:



¿Cuál es la salida? La mejor manera de determinar la salida es a través de cálculos algebraicos. Después de usar la propiedad distributiva del producto de Kronecker sobre la suma de vectores, la entrada al circuito es

$$\frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes |0\rangle \;=\; \frac{1}{\sqrt{2}}\,|0\rangle \otimes |0\rangle + \frac{1}{\sqrt{2}}\,|1\rangle \otimes |0\rangle \;=\; \frac{|00\rangle + |10\rangle}{\sqrt{2}}.$$

Para calcular la acción de CNOT sobre una suma de vectores, usamos la linealidad del producto de matrices, es decir,

$$\text{CNOT} \cdot \left(\frac{|00\rangle + |10\rangle}{\sqrt{2}} \right) \;=\; \frac{1}{\sqrt{2}} \,\text{CNOT} \cdot |00\rangle + \frac{1}{\sqrt{2}} \,\text{CNOT} \cdot |10\rangle,$$

donde $\text{CNOT} \cdot |00\rangle$ denota la multiplicación de la matriz CNOT por el vector $|00\rangle$. Usando la definición dada al comienzo de la Sección, obtenemos $\text{CNOT}|00\rangle = |00\rangle$ y $\text{CNOT}|10\rangle = |11\rangle$, y confirmamos que la salida es

$$\frac{|00\rangle + |11\rangle}{\sqrt{2}}.$$

Dado que este resultado es un estado entrelazado, no podemos factorizarlo y por lo tanto, no podemos escribir la salida para cada qubit.

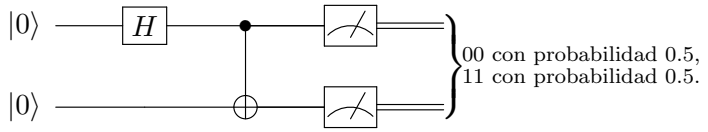
El mismo resultado se obtiene si usamos la representación matricial, que es

$$\text{CNOT} = \begin{bmatrix} I_2 & \\ & X \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix},$$

y la representación de $|00\rangle$ y $|10\rangle$ como vectores de 4 dimensiones, es decir,

$$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

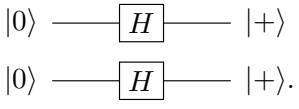
El circuito completo que implementa el estado entrelazado anterior, cuando el estado inicial de la computadora cuántica es $|00\rangle$ es



Dado que el estado del primer qubit es inicialmente $|0\rangle$, tenemos que usar H para generar $(|0\rangle + |1\rangle)/2$. De hecho, tenemos

$$\text{CNOT} \cdot (H \otimes I)\,|00\rangle = \text{CNOT} \cdot \big(H|0\rangle \otimes |0\rangle\big) = \frac{|00\rangle + |11\rangle}{\sqrt{2}}.$$

El siguiente ejemplo es más sencillo que el anterior. Considere el circuito sin mediciones.



¿Cuál es la salida? Por lo general, para calcular la salida, convertimos el circuito a su expresión algebraica equivalente. Para este circuito tenemos $(H \otimes H)|00\rangle$. El cálculo se realiza de la siguiente manera:

$$(H \otimes H)|00\rangle \;=\; (H \otimes H) \cdot (|0\rangle \otimes |0\rangle) \;=\; (H|0\rangle) \otimes (H|0\rangle) \;=\; |+\rangle \otimes |+\rangle.$$

En la segunda igualdad, usamos la siguiente propiedad del producto de Kronecker:

$$(A \otimes B) \cdot (C \otimes D) \;=\; (A \cdot C) \otimes (B \cdot D),$$

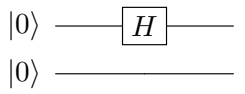
para matrices A, B, C, D, o

$$(A \otimes B) \cdot (|\psi_1\rangle \otimes |\psi_2\rangle) \;=\; (A|\psi_1\rangle) \otimes (B|\psi_2\rangle),$$

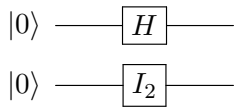
lo cual es válido para cualquier matriz A y B y vectores $|\psi_1\rangle$ y $|\psi_2\rangle$ siempre que el número de entradas de los vectores sea igual al número correspondiente de columnas de las matrices. Entonces la salida del circuito es

$$|+\rangle \otimes |+\rangle \;=\; \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} \;=\; \frac{|0\rangle + |1\rangle + |2\rangle + |3\rangle}{2}.$$

Consideremos otro ejemplo. Tome el siguiente circuito sin mediciones:



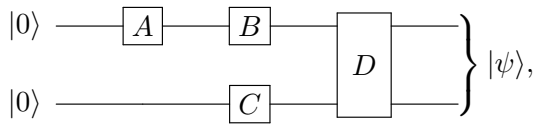
¿Cómo calcular la salida usando la expresión algebraica equivalente? La sugerencia es usar el siguiente circuito equivalente:



donde I_2 es la matriz identidad bidimensional. El cálculo algebraico se realiza de la siguiente manera:

$$(H \otimes I_2)|00\rangle \;=\; (H|0\rangle) \otimes (I_2|0\rangle) \;=\; |+\rangle \otimes |0\rangle \;=\; \frac{|00\rangle + |10\rangle}{\sqrt{2}}.$$

Cuando convertimos un circuito cuántico a su expresión algebraica equivalente, debemos usar el producto de Kronecker para puertas en la misma columna y el producto matricial para puertas en el mismo cable o en secuencia; sin embargo, debemos invertir el orden de las puertas en el segundo caso. Por ejemplo, la expresión algebraica equivalente al circuito



donde A, B y C son puertas de 1 qubit y D es una puerta de 2 qubits, es

$$|\psi\rangle \;=\; D \cdot (B \otimes C) \cdot (A \otimes I_2) \cdot |00\rangle.$$

Podemos simplificar un poco esta expresión y escribir

$$|\psi\rangle \;=\; D\,(BA \otimes C)|00\rangle.$$

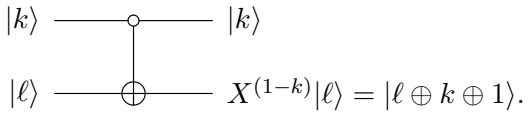
Sólo D puede crear o destruir entrelazamientos. Los operadores A, B y C no pueden crear ni destruir entrelazamientos. La prueba de que A no puede crear ni destruir el entrelazamientos es la siguiente. Suponemos que el estado de entrada es $|\psi_1\rangle \otimes |\psi_2\rangle$ (desentrelazado). La acción de A genera $(A|\psi_1\rangle) \otimes |\psi_2\rangle$, que se desentrelaza. Ahora suponga que la entrada es un estado

entrelazado $|\psi\rangle$. La acción de A genera $(A \otimes I)|\psi\rangle$. Si este estado es desentrelazado, es decir, existen $|\psi_1\rangle$ y $|\psi_2\rangle$ tales que $(A \otimes I)|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$, llegamos a una contradicción porque la última ecuación es equivalente a $|\psi\rangle = (A^\dagger|\psi_1\rangle) \otimes |\psi_2\rangle$.

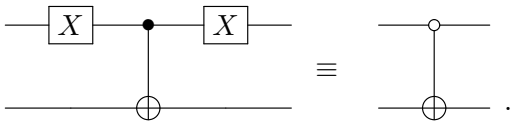
CNOT es tan importante que describimos una variante que es CNOT activado por 0. Se define por la expresión

$$|k\rangle|\ell\rangle \longrightarrow |k\rangle X^{(1-k)}|\ell\rangle,$$

y está representado por el circuito



Tenga en cuenta que el qubit de control se indica mediante un círculo vacío que indica que el control de CNOT no está activado si el estado del qubit de control es $|1\rangle$. Esta puerta se obtiene del CNOT habitual multiplicando $(X \otimes I_2)$ en ambos lados, como se muestra en la siguiente equivalencia de circuito:



Esta puerta está representada por una matriz de bloques de la forma

$$(X \otimes I_2) \cdot CNOT \cdot (X \otimes I_2) = \begin{bmatrix} & I_2 \\ I_2 & \end{bmatrix} \begin{bmatrix} I_2 & \\ & X \end{bmatrix} \begin{bmatrix} & I_2 \\ I_2 & \end{bmatrix} = \begin{bmatrix} X & \\ & I_2 \end{bmatrix}.$$

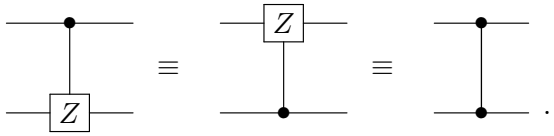
Una forma alternativa de obtener la representación matricial es enumerar secuencialmente la salida de cada vector de la base computacional

$$\begin{aligned} |00\rangle &\xrightarrow{\circ\!-\!\oplus} |01\rangle, \\ |01\rangle &\xrightarrow{\circ\!-\!\oplus} |00\rangle, \\ |10\rangle &\xrightarrow{\circ\!-\!\oplus} |10\rangle, \\ |11\rangle &\xrightarrow{\circ\!-\!\oplus} |11\rangle. \end{aligned}$$

Luego, se convierte cada salida a la notación vectorial y se une una al lado de la otra para formar una matriz:

$$\begin{pmatrix}0\\1\\0\\0\end{pmatrix}\begin{pmatrix}1\\0\\0\\0\end{pmatrix}\begin{pmatrix}0\\0\\1\\0\end{pmatrix}\begin{pmatrix}0\\0\\0\\1\end{pmatrix} \longrightarrow \begin{bmatrix}0&1&0&0\\1&0&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix} = \begin{bmatrix} X & \\ & I_2 \end{bmatrix}.$$

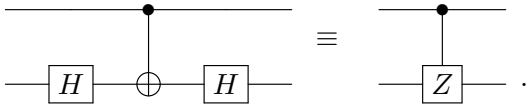
La segunda puerta de 2 qubits más importante es la puerta Z controlada, denotada por $C(Z)$ o CZ. Sus representaciones de circuito son



Tenga en cuenta que no importa qué qubit sea el control o el objetivo, es decir, Z puede estar controlado por el primer qubit y el objetivo por el segundo, o al revés. La tercera representación es interesante porque los qubits están en pie de igualdad. La representación matricial es única y está dada por

$$C(Z) = \begin{bmatrix} I_2 & \\ & Z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}.$$

Las puertas CNOT y $C(Z)$ están conectadas como se muestra en la siguiente equivalencia del circuito:



La equivalencia se sigue de

$$(I \otimes H)\,\mathrm{CNOT}\,(I \otimes H) = C(Z),$$

que se puede mostrar usando la representación matricial. Hay una forma alternativa de mostrar la equivalencia usando el hecho de que $H^2 = I$ y luego $I \otimes H$ pueden ser reemplazados por $C(H)$ porque dos H actúan trivialmente si el control del CNOT no está activado. Entonces tenemos

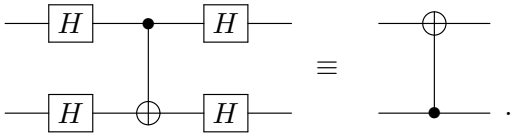
$$(I \otimes H)\,\mathrm{CNOT}\,(I \otimes H) = C(H)C(X)C(H) = C(HXH) = C(Z),$$

porque $HXH = Z$ y $C(A)C(B) = C(AB)$ para cualquier puerta de 1 qubit A y B.

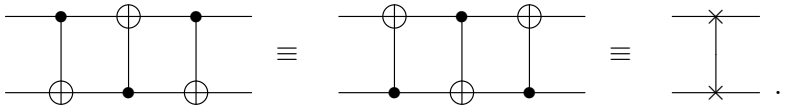
Ahora estamos listos para mostrar que el control y el objetivo de un CNOT se invierten cuando multiplicamos CNOT por $H \otimes H$ en ambos lados. Por lo tanto tenemos,

$$(H \otimes H)\,\mathrm{CNOT}\,(H \otimes H) = (H \otimes I)(I \otimes H)\,\mathrm{CNOT}\,(I \otimes H)(H \otimes I) = (H \otimes I)C(Z)(H \otimes I).$$

Considere el primer qubit como el objetivo de $C(Z)$. Luego tenemos $HZH = X$ controlado por el segundo qubit, y hemos terminado. La representación del circuito es



La tercera puerta de 2 qubits más importante es la puerta SWAP. Sus representaciones en un circuito son



La representación matricial es

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

Implementación en las computadoras cuánticas de la IBM.

La implementación del circuito con puertas H y CNOT en el composer de la IBM se muestra en la Fig. 2.5. Tenemos que arrastrar y soltar una puerta H y una puerta CNOT, de modo que el control de CNOT sea el qubit q_0 y el objetivo sea el qubit q_1. También tenemos que arrastrar y soltar dos medidores. Si queremos ver la simulación exacta de la distribución de probabilidad en el composer, tenemos que quitar los medidores. Tenga en cuenta que en el composer, el qubit q_0 representa el qubit más a la derecha, es decir, el orden de los qubits es $|q_2q_1q_0\rangle$ y la salida se refiere a $q_2q_1q_0$. En este trabajo y en la mayoría de los artículos y libros, se asume el orden opuesto. El resultado de una simulación con 1024 repeticiones se muestra en la Fig. 2.6. La figura muestra que el resultado 000 se obtuvo 507 veces, 001 21 veces, 010 51 veces y 011 445 veces. Debemos descartar el qubit no utilizado q_2 (qubit más a la izquierda). Por lo tanto, el resultado muestra que el algoritmo devuelve 00 con una probabilidad del 49,5 % y 11 con una probabilidad del 43,5 % aproximadamente, lo cual es inesperado según el cálculo analítico. Se obtuvieron los resultados erróneos 01 y 10 porque las computadoras cuánticas son propensas a errores.

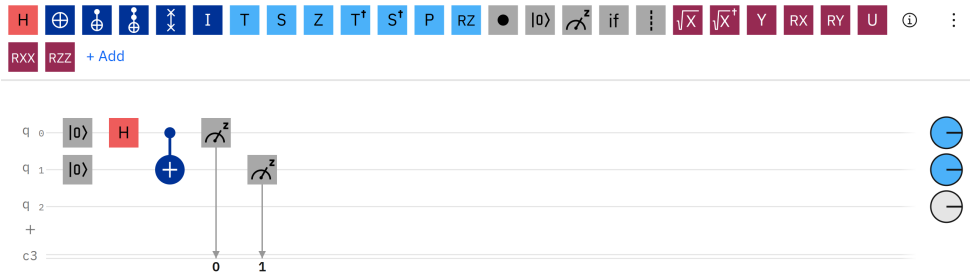


Figura 2.5: Circuito con puertas H y CNOT (Reimpresión cortesía de IBM Corporation ©)

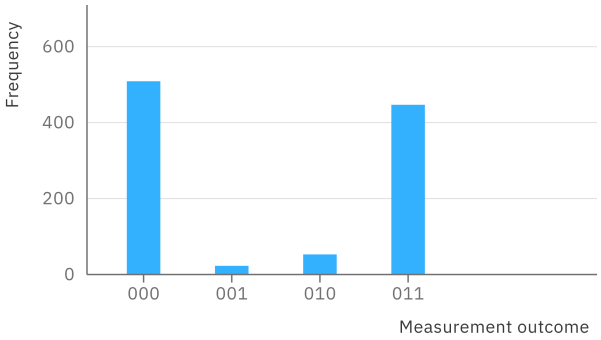


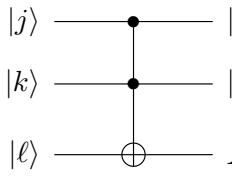
Figura 2.6: Salida del circuito en la Fig. 2.5 usando la computadora cuántica *ibmq_manila* (Reimpresión cortesía de IBM Corporation ©)

2.6. Puertas multiqubit

La puerta de 3 qubits más importante es la puerta Toffoli, denotada por CCNOT o $C^2(X)$, definida por la expresión

$$C^2(X)\,|j\rangle|k\rangle|\ell\rangle = |j\rangle|k\rangle X^{jk}|\ell\rangle,$$

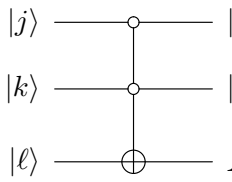
y representada por el circuito



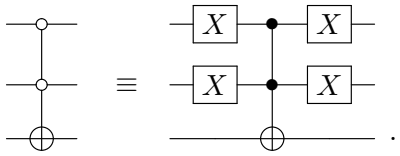
Esta puerta tiene dos controles y un objetivo. La puerta X actúa sobre el qubit de destino sí y solo sí ambos qubits de control tienen valor uno. Si un control tiene valor cero, el objetivo no cambia. Esta puerta puede verse como la versión cuántica de la puerta AND clásica porque si $\ell = 0$, la salida del tercer qubit es (j AND k). La representación matricial de la puerta de Toffoli es una matriz de bloques diagonales dada por

$$C^2(X) = \begin{bmatrix} I_2 & & & \\ & I_2 & & \\ & & I_2 & \\ & & & X \end{bmatrix}.$$

Hay variantes de la puerta Toffoli que se activan cuando los qubits de control se establecen en cero. Por ejemplo, el circuito



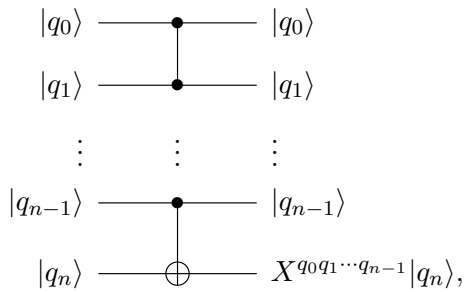
implementa una puerta que aplica X en el tercer qubit sí y solo sí los dos primeros qubits de control se establecen en cero. Se puede implementar usando una puerta Toffoli estándar y puertas X, como se muestra en la siguiente equivalencia de circuito:



La puerta multiqubit Toffoli $C^n(X)$ es una puerta de $(n+1)$-qubits con n qubits de control y un objetivo. Se define por la expresión

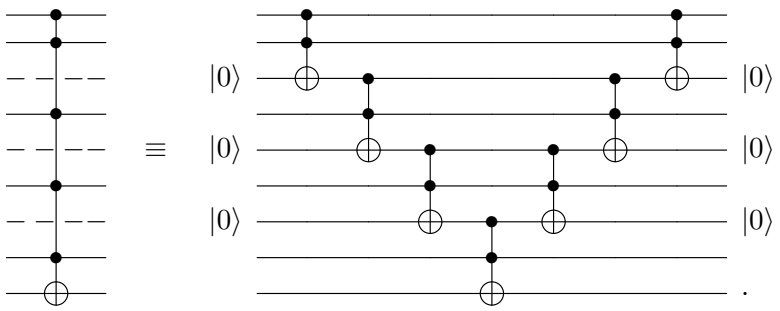
$$C^n(X)|q_0\rangle|q_1\rangle\cdots|q_{n-1}\rangle|q_n\rangle = |q_0\rangle|q_1\rangle\cdots|q_{n-1}\rangle X^{q_0 q_1 \cdots q_{n-1}}|q_n\rangle,$$

y está representado por el circuito



donde $q_0 q_1 \cdots q_{n-1}$ es el producto de los bits q_0, q_1, ..., q_{n-1}. Por lo tanto, el estado del qubit objetivo cambia sí y solo sí todos los qubits de control tienen valor 1. El estado de cada qubit de control no cambia cuando describimos esta puerta actuando sobre la base computacional. La acción sobre un vector genérico se obtiene escribiendo el vector como una combinación lineal de vectores de la base computacional y luego usando la linealidad.

La forma más sencilla de descomponer la puerta Toffoli multiqubit en términos de la puerta de Toffoli habitual es usando $(n-2)$ borradores qubits llamados *ancillas*. Los qubits auxiliares se entrelazan con los qubits de control, insertando el primer qubit auxiliar entre el segundo y el tercer qubit. La mejor manera de explicar la descomposición es mostrar un ejemplo. Considere la puerta $C^5(X)$, cuya descomposición requiere tres *ancillas*, como se muestra en el siguiente circuito de equivalencia:



La puerta Toffoli multiqubit también se puede activar por cero. En este caso, el qubit de control se muestra como un círculo vacío. Para n qubits, tenemos puertas Toffoli de 2^n multiqubit, que pueden implementar cualquier función booleana de n bits, como se describe en la siguiente sección.

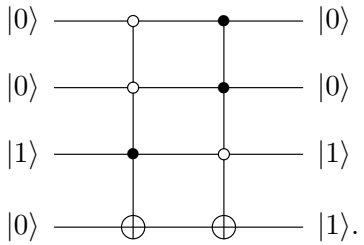
2.7. Circuito de una función booleana

Vamos a mostrar como obtener el circuito cuántico de una tabla de verdad. Solo necesitamos puertas Toffoli multiqubit. Para mostrar que las puertas Toffoli multiqubit pueden implementar cualquier función booleana en una computadora cuántica, tomemos como ejemplo la función booleana de 3 bits $f(a, b, c)$ definida por la siguiente tabla de verdad:

a	b	c	$f(a,b,c)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Después de este ejemplo, es evidente como se obtiene el caso general. Como f tiene tres bits de entrada, usamos puertas Toffoli multiqubit con tres controles. El cuarto qubit es el objetivo. La salida de f es la salida de la medición del qubit objetivo. Dado que f tiene dos cláusulas en la

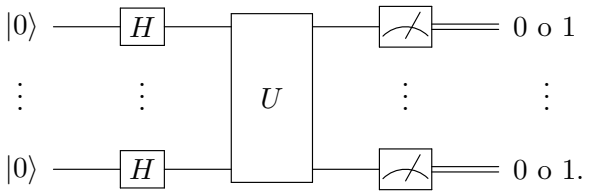
forma normal disyuntiva, usamos dos puertas Toffoli multiqubit.[7] La primera puerta debe ser activada por la entrada $|001\rangle$ y la segunda por $|110\rangle$, que corresponden a las filas de la tabla de verdad cuya salida es 1. El siguiente circuito implementa f:



Tenga en cuenta que si la entrada es $|a, b, c\rangle|0\rangle$, la salida será $|a, b, c\rangle|f(a, b, c)\rangle$. Esto muestra que la computadora cuántica puede calcular cualquier función booleana de n-bits usando una puerta Toffoli multiqubit con $(n+1)$ qubits para cada salida 1 de la tabla de verdad. El objetivo aquí no es implementar algoritmos clásicos en computadoras cuánticas porque no tiene sentido construir una máquina mucho más costosa para ejecutar solo algoritmos clásicos. Sin embargo, la implementación que acabamos de describir puede usarse para entradas en superposición, lo que no está permitido en una computadora clásica. Desafortunadamente, esta técnica de construcción de circuitos cuánticos para calcular tablas de verdad no es eficiente, ya que el número de puertas Toffoli multiqubit aumenta exponencialmente en función del número de qubits en el peor de los casos.

2.8. Paralelismo cuántico

El modelo más estándar de la computación cuántica se describe mediante el circuito



El estado inicial de cada qubit es $|0\rangle$. Luego, la puerta de Hadamard se aplica a cada qubit, preparándose para el *paralelismo cuántico*. Luego, se aplica la matriz unitaria U a todos los qubits. Finalmente, hay un medidor para cada qubit, devolviendo una cadena de bits.

El paralelismo cuántico es la ejecución simultánea de un algoritmo con más de una entrada a un solo procesador cuántico. Para ejecutar la misma tarea en una computadora clásica se requeriría un número exponencial de procesadores clásicos. Para entender este concepto, nos enfocamos sólo en U sin considerar las puertas H. U y los medidores se pueden interpretar como un algoritmo aleatorio en el sentido clásico. Si x es una cadena de bits n y U es una matriz $2^n \times 2^n$, entonces $U|x\rangle$ es una combinación lineal de 2^n vectores de la base computacional. Después de las mediciones, se devuelve una cadena n-bits y, donde $|y\rangle$ es uno de los términos

[7] En la forma normal disyuntiva, solo consideramos las filas de la tabla de verdad cuya salida es 1. Para cada fila con salida 1 , escribimos una conjunción de tres literales, usando NOT para cada entrada 0. Luego escribimos una disyunción de las conjunciones, así, $f(a, b, c) = (\bar{a} \wedge \bar{b} \wedge c) \vee (a \wedge b \wedge \bar{c})$, donde $(\bar{a} \wedge \bar{b} \wedge c)$ está asociado con 001 y $(a \wedge b \wedge \bar{c})$ con 110.

de la combinación lineal. Es decir, si la entrada a U es x, la salida después de la medición es y. U es un "algoritmo", que se ejecuta para una sola entrada x cuando se realiza el cálculo $U|x\rangle$. Ahora traemos de vuelta las puertas H.

El primer paso del circuito es realizar el siguiente cálculo: $(H|0\rangle) \otimes \cdots \otimes (H|0\rangle)$, es decir

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

Después de expandir este producto, obtenemos

$$\frac{1}{\sqrt{2^n}}\big(|0\cdots0\rangle + |0\cdots01\rangle + |0\cdots10\rangle + \cdots + |1\cdots1\rangle\big).$$

Cada cadena de n-bits está dentro de un ket de la suma anterior. Es decir, todas las entradas posibles de un algoritmo clásico están representadas por kets en la suma anterior. El siguiente paso en el modelo estándar es aplicar U. Debido a la linealidad del álgebra lineal, U se aplica a todos los kets de la suma anterior *simultáneamente*. Por lo tanto, es posible realizar 2^n cálculos simultáneos en una computadora cuántica de n-qubits. Tenga en cuenta, sin embargo, que el resultado de estos cálculos es un estado de superposición y, después de una medición, el resultado final es una sola cadena de n-bits.

Podemos refinar el modelo estándar agregando un registrador adicional para borradores de los cálculos. Dado que las puertas unitarias son invertibles, todo el proceso de cálculo sin medición es reversible. Esto significa que no se borra ninguna información. El número de qubits de salida debe ser igual al número de qubits de entrada. Posponemos la discusión sobre este refinamiento y lo presentamos después de describir y analizar los algoritmos cuánticos básicos.

Capítulo 3

Algoritmo de Deutsch

El algoritmo de Deutsch es el primer algoritmo que explora el paralelismo cuántico. Utiliza dos qubits (sólo uno en la versión económica) y tiene una ganancia muy modesta, pero ha inspirado la construcción de varios algoritmos cuánticos nuevos que son más eficientes que su contraparte clásica. El problema de Deutsch se planteó en el año 1985 [17] sin utilizar aún el modelo de circuito cuántico. Los conceptos de puertas universales y circuitos cuánticos se presentaron por primera vez en [18], aproximadamente cuatro años después. Una generalización del algoritmo de Deutsch se describió en [19] y se conoce como el algoritmo de Deutsch-Jozsa (ver el próximo Capítulo). La versión descrita aquí sigue la visión moderna del algoritmo de Deutsch [14, 42], que difiere ligeramente del original. En la Sección final, describimos una implementación del algoritmo de Deutsch con solo un qubit.

3.1. Formulación del problema

Supongamos que tenemos una función booleana de 1 bit $f : \{0,1\} \longrightarrow \{0,1\}$ sin conocer los detalles de la implementación de f. Queremos determinar si esta función es *balanceada* o *constante*. Una función booleana de 1 bit está balanceada si $f(0) \neq f(1)$; en caso contrario, la función es constante; en este caso $f(0) = f(1)$. Hay cuatro funciones booleanas f, cuyas tablas de verdad se describen en la Fig. 3.1 con los nombres f_0 a f_3. Las *formas normales disyuntivas*

x	$f_0(x)$	$f_1(x)$	$f_2(x)$	$f_3(x)$
0	0	0	1	1
1	0	1	0	1

Figura 3.1: Tablas de verdad de todas las funciones de 1 bit.

son

$$
\begin{aligned}
f_0(x) &= \text{Falso}, \\
f_1(x) &= x, \\
f_2(x) &= \bar{x}, \\
f_3(x) &= \bar{x} \vee x,
\end{aligned}
$$

respectivamente, donde $\bar{x} = \text{NOT}\, x$. La expresión booleana de f_3 se puede simplificar dado que $\bar{x} \vee x = \text{Verdadero}$.

Un algoritmo clásico que encuentre la solución a este problema necesita evaluar f dos veces, es decir, realmente es necesario evaluar $f(0)$ y $f(1)$. Sin embargo, el algoritmo de Deutsch usa un operador unitario U_f que implementa f y llama a este operador sólo una vez. En este caso, también se evalúan $f(0)$ y $f(1)$, pero la diferencia es que las evaluaciones se realizan simultáneamente. Esta idea se utiliza de forma recurrente en los algoritmos cuánticos.

En el caso cuántico, f se implementa a través de un operador unitario de 2 qubits U_f definido como

$$U_f|x\rangle|j\rangle = |x\rangle|j \oplus f(x)\rangle,$$

donde $\oplus$ es la operación XOR o suma módulo 2. Esta es una receta que se puede usar para implementar una función booleana de n-bits arbitraria. Tenemos que tener cuidado de que x sea la entrada del primer registrador, y para obtener $f(x)$ establecemos $j = 0$ como la entrada del segundo registrador y luego observamos la salida del segundo registrador. Ahora usamos la técnica descrita en el Capítulo 2 para obtener el circuito cuántico de funciones f_0 a f_3. Usamos sus formas normales disyuntivas y tenemos que usar una puerta Toffoli multiqubit para cada salida 1 en la tabla de verdad. En el caso de 2 qubits, una puerta Toffoli multiqubit es un CNOT activado por 0 o 1. No hay salida 1 en la tabla de verdad de f_0. Después,

$$U_{f_0} = I \otimes I.$$

Hay una salida 1 en la tabla de verdad de f_1, que tiene entrada 1. Usamos el CNOT estándar, que se activa cuando el control se activa con 1. Luego,

$$U_{f_1} = \text{CNOT}.$$

Hay una salida 1 en la tabla de verdad de f_2, que tiene entrada 0. Usamos el CNOT que se activa cuando el control es activado en 0. Luego,

$$U_{f_2} = (X \otimes I) \cdot \text{CNOT} \cdot (X \otimes I).$$

Finalmente, hay dos salidas 1 en la tabla de verdad de f_3 con entrada 0 y 1. Usamos el CNOT estándar y el CNOT que se activa cuando el control se activa en 0. Luego,

$$U_{f_3} = (X \otimes I) \cdot \text{CNOT} \cdot (X \otimes I) \cdot \text{CNOT}.$$

Está claro que la receta general para implementar funciones booleanas usando sus tablas de verdad produce circuitos grandes innecesarios. En este caso, necesitamos simplificar la expresión booleana antes de construir el circuito. En este punto, no hay una receta que nos guíe, excepto nuestra habilidad para manejar funciones booleanas y hacer circuitos. Para f_3, sabemos que $f_3(x) =$ Verdadero y la salida debe ser 1 independientemente de la entrada x al primer qubit. Dado que la entrada al segundo qubit es 0, la salida 1 se obtiene mediante un X. Entonces, la versión simplificada de U_{f_3} es

$$U_{f_3} = I \otimes X.$$

Tenga en cuenta que U_f es unitario en todos los casos.

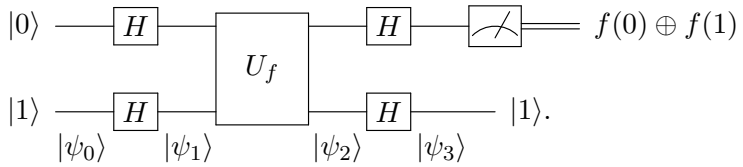
Algoritmo 3.1: Algoritmo de Deutsch

Entrada: Una función booleana $f : \{0,1\} \longrightarrow \{0,1\}$.
Salida: 0 si f es constante, 1 si f está balanceada.

1 Preparar el estado inicial $|0\rangle|1\rangle$;
2 Aplicar $H \otimes H$;
3 Aplicar U_f;
4 Aplicar $H \otimes H$;
5 Mida el primer qubit en la base computacional.

3.2. El algoritmo

El algoritmo de Deutsch se describe en el Algoritmo 3.1 y el circuito (no económico) es



Comprobamos fácilmente que el circuito corresponde exactamente a los pasos del Algoritmo 3.1. La salida $f(0) \oplus f(1)$ es 0 si f es constante, y 1 si f está balanceado. Tenga en cuenta que la última puerta Hadamard aplicada al segundo qubit se puede eliminar sin afectar el algoritmo. Esta puerta está aquí porque la parte central del circuito es simétrica y el análisis del algoritmo es más claro con ella que sin ella. Los estados en la parte inferior del circuito se utilizan en el análisis del algoritmo.

3.3. Análisis del algoritmo

Después del primer paso, el estado de la computadora cuántica es

$$|\psi_0\rangle = |0\rangle \otimes |1\rangle.$$

Después del segundo paso, el estado de la computadora cuántica es

$$\begin{aligned} |\psi_1\rangle &= (H|0\rangle) \otimes (H|1\rangle) \\ &= \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes |-\rangle, \end{aligned}$$

donde $|-\rangle = (|0\rangle - |1\rangle)/\sqrt{2}$. Después del tercer paso, el estado de la computadora cuántica es

$$\begin{aligned} |\psi_2\rangle &= U_f|\psi_1\rangle \\ &= \frac{U_f|0\rangle|-\rangle + U_f|1\rangle|-\rangle}{\sqrt{2}}. \end{aligned}$$

Para simplificar $|\psi_2\rangle$, mostremos la siguiente proposición:

Proposición 3.1. Sea $f : \{0,1\}^n \longrightarrow \{0,1\}$ una función booleana de n bits y defina U_f como

$$U_f = \sum_{x\in\{0,1\}^n} \sum_{j=0}^{1} |x, j \oplus f(x)\rangle\langle x, j|.$$

Después

$$U_f\big(|x\rangle \otimes |-\rangle\big) \;=\; (-1)^{f(x)}|x\rangle \otimes |-\rangle.$$

Demostración. Usando la definición de $|-\rangle$, obtenemos

$$U_f\big(|x\rangle \otimes |-\rangle\big) \;=\; \frac{U_f|x\rangle|0\rangle - U_f|x\rangle|1\rangle}{\sqrt{2}}.$$

Usando la definición de U_f, obtenemos

$$U_f\big(|x\rangle \otimes |-\rangle\big) \;=\; \frac{|x\rangle|f(x)\rangle - |x\rangle|1 \oplus f(x)\rangle}{\sqrt{2}}.$$

Si $f(x) = 0$, el lado derecho es $|x\rangle|-\rangle$ y si $f(x) = 1$, el lado derecho es $-|x\rangle|-\rangle$. Entonces, juntando estos resultados, tenemos $(-1)^{f(x)}|x\rangle \otimes |-\rangle$. □

Usando la proposición anterior, simplificamos $|\psi_2\rangle$ a

$$\begin{aligned} |\psi_2\rangle &= \frac{(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle}{\sqrt{2}} \otimes |-\rangle \\ &= \begin{cases} \pm\,|+\rangle \otimes |-\rangle, & \text{si } f(0) = f(1), \\ \pm\,|-\rangle \otimes |-\rangle, & \text{si } f(0) \neq f(1). \end{cases} \end{aligned}$$

En el último paso, hemos simplificado el estado del primer qubit sin especificar exactamente cuál es el signo de la amplitud. Este signo no tiene efecto en la salida del algoritmo. Después del cuarto paso, el estado de la computadora cuántica es

$$\begin{aligned} |\psi_3\rangle &= (H \otimes H)|\psi_2\rangle \\ &= \begin{cases} \pm\,|0\rangle \otimes |1\rangle, & \text{si } f(0) = f(1), \\ \pm\,|1\rangle \otimes |1\rangle, & \text{si } f(0) \neq f(1), \end{cases} \end{aligned}$$

donde hemos usado el hecho que $H|+\rangle = |0\rangle$ y $H|-\rangle = |1\rangle$, que se puede deducir usando $|+\rangle = H|0\rangle$, $|-\rangle = H|1\rangle$ y $H^2 = I$. El estado $|\psi_3\rangle$ se puede simplificar aún más para

$$|\psi_3\rangle \;=\; \pm\,|f(0) \oplus f(1)\rangle \otimes |1\rangle.$$

Luego del quinto paso, la medición del primer qubit en la base computacional retorna $f(0)\oplus f(1)$, que es 0 si f es constante y 1 si f está balanceada, concluyendo el análisis del algoritmo.

Nótese que el signo de $|\psi_3\rangle$ no influye en el resultado de la medición porque la probabilidad de obtener $f(0) \oplus f(1)$ es $| \pm 1|^2 = 1$. Es fácil comprobar, si es relevante, que este signo es $(-1)^{f(0)}$.

3.4. Análisis del entrelazamiento

Resumiendo el conjunto de los estados de los qubits después de cada paso, tenemos

$$
\begin{aligned}
|\psi_0\rangle &= |0\rangle \otimes |1\rangle, \\
|\psi_1\rangle &= |+\rangle \otimes |-\rangle, \\
|\psi_2\rangle &= \begin{cases} \pm\,|+\rangle \otimes |-\rangle, & \text{si } f(0) = f(1), \\ \pm\,|-\rangle \otimes |-\rangle, & \text{si } f(0) \neq f(1), \end{cases} \\
|\psi_3\rangle &= \pm\,|f(0) \oplus f(1)\rangle \otimes |1\rangle.
\end{aligned}
$$

Independientemente de f, la computadora cuántica no pasa por un estado entrelazado durante la ejecución del algoritmo de Deutsch porque cada estado $|\psi_i\rangle$ para i de 1 a 3 es un producto tensorial de estados puros de 1 qubit. Los qubits están desentrelazados todo el tiempo. Esto significa que el algoritmo de Deutsch es más rápido que los algoritmos clásicos que utilizan únicamente el paralelismo cuántico.

Hay una ruta alternativa para analizar el entrelazamiento. Ningún operador de 2 qubits $A \otimes B$ crea o destruye el entrelazamiento. En el algoritmo de Deutsch, solo el CNOT puede crear entrelazamiento y se usa como máximo una vez. Entonces, para cada función f podemos simplificar todo el algoritmo para obtener solo un operador final, cuya entrada es $|01\rangle$. Si simplificamos el algoritmo para cada f, obtenemos

$$
\begin{aligned}
(H \otimes H)\, U_{f_0} (H \otimes H) &= (H \otimes H) \cdot (H \otimes H) = I \otimes I, \\
(H \otimes H)\, U_{f_1} (H \otimes H) &= (H \otimes H) \cdot \mathrm{CNOT} \cdot (H \otimes H) = \mathrm{CNOT}_{10}, \\
(H \otimes H)\, U_{f_2} (H \otimes H) &= (Z \otimes I) \cdot \mathrm{CNOT}_{10} \cdot (Z \otimes I), \\
(H \otimes H)\, U_{f_3} (H \otimes H) &= I \otimes Z,
\end{aligned}
$$

donde el control de CNOT_{10} es el segundo qubit y el objetivo es el primer qubit. Concluimos de inmediato que no se crea ningún entrelazamiento cuando f es f_0 o f_3. Para f_1 y f_2, el qubit de control del CNOT se ubica en 1 porque la entrada es $|01\rangle$, en estos casos, tampoco se crea entrelazamiento porque el CNOT crea entrelazamiento sólo si el estado del qubit de control es un estado de superposición.

Podemos aprovechar la versión simplificada del algoritmo para verificar la salida nuevamente y volver a analizar el algoritmo. Los estados de los qubits justo antes de la medición son

$$
\begin{aligned}
|\psi_3\rangle\big|_{f_0} &= (I \otimes I)|01\rangle = |0,1\rangle, \\
|\psi_3\rangle\big|_{f_1} &= \mathrm{CNOT}_{10}|01\rangle = |1,1\rangle, \\
|\psi_3\rangle\big|_{f_2} &= (Z \otimes I) \cdot \mathrm{CNOT}_{10} \cdot (Z \otimes I)|01\rangle = -|1,1\rangle, \\
|\psi_3\rangle\big|_{f_3} &= (I \otimes Z)|01\rangle = -|0,1\rangle.
\end{aligned}
$$

La salida de una medición del primer qubit es 0 para f_0 y f_3, que son las funciones constantes, y 1 para f_1 y f_2, que son las funciones balanceadas. También podemos comprobar que el signo es $(-1)^{f(0)}$ porque $f_0(0) = f_1(0) = +1$ y $f_2(0) = f_3(0) = -1$.

3.5. ¿Quién implementa el oráculo?

Se dice que el algoritmo de Deutsch es más eficiente que su contraparte clásica en un contexto limitado llamado "complejidad de consulta" [30]. En este contexto, debemos comparar el número

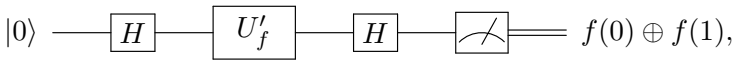
de evaluaciones de f en el caso clásico con el número de veces que se usa U_f en el caso cuántico. Esta es una regla del juego. El algoritmo de Deutsch aplica U_f solo una vez y el algoritmo clásico necesita evaluar f dos veces. Entonces, Deutsch es más rápido.

Tenga en cuenta que el análisis del algoritmo de Deutsch muestra que f se evalúa en dos puntos distintos del dominio simultáneamente. Esto no es posible de llevar a cabo utilizando un algoritmo clásico secuencial. Sin embargo, se puede llevar a cabo en una computadora clásica con procesadores paralelos si el número de subprocesos simultáneos no aumenta. Dado que el algoritmo de Deutsch utiliza un número fijo de qubits, no podemos ampliarlo y no es posible realizar un análisis asintótico. En términos de complejidad temporal, la versión cuántica no tiene ganancia. La importancia del algoritmo de Deutsch radica en que estimuló la búsqueda de generalizaciones, como los algoritmos de Deutsch-Jozsa, Bernstein-Vazirani y Simon, que inspiraron a Shor en la elaboración de un algoritmo cuántico para la factorización de enteros compuestos y un algoritmo cuántico para calcular logaritmos discretos.

Por último, pero no menos importante, destacamos que no depende de nosotros implementar el oráculo. Es el trabajo de otra persona. Sin esta comprensión, nos enfrentamos a una contradicción: tenemos que saber la respuesta incluso antes de comenzar a implementar el algoritmo que encontrará la respuesta. Se permite consultar la función f implementada por alguien sin mirar los detalles de su implementación. Recibimos lo que se llama una *caja negra* de una computadora cuántica con U_f ya implementado, que podemos usar más de una vez, y podemos agregar nuevas puertas, pero no podemos ver las entrañas de la caja negra que implementa U_f.

3.6. Circuito económico del algoritmo de Deutsch

El algoritmo de Deutsch se puede implementar con sólo un qubit de la siguiente manera:



donde

$$U'_f = \sum_{x=0}^{1} (-1)^{f(x)} |x\rangle\langle x|.$$

De la Proposición 3.1, vemos que el segundo qubit no es necesario para el algoritmo, aunque es necesario si queremos obtener $f(x)$ en el sentido clásico, como se muestra a continuación. Expandiendo la suma, obtenemos

$$U'_f = \begin{bmatrix} (-1)^{f(0)} & 0 \\ 0 & (-1)^{f(1)} \end{bmatrix}.$$

Después,

$$U'_f = \begin{cases} \pm I, \text{ si } f(0) = f(1), \\ \pm Z, \text{ si } f(0) \neq f(1). \end{cases}$$

El análisis del algoritmo se reduce a calcular

$$HU'_f H|0\rangle = \begin{cases} \pm\, I|0\rangle, & \text{si } f(0) = f(1), \\ \pm\, X|0\rangle, & \text{si } f(0) \neq f(1). \end{cases}$$

Usando el hecho que $f(0) \oplus f(1) = 0$ si $f(0) = f(1)$, y $f(0) \oplus f(1) = 1$ si $f(0) \neq f(1)$, obtenemos

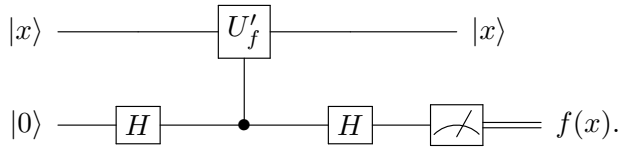
$$HU'_f H|0\rangle = \pm|f(0) \oplus f(1)\rangle.$$

Después de una medición, la salida es $f(0) \oplus f(1)$ con probabilidad $|\pm 1|^2 = 1$.

Ahora es sencillo verificar que no haya entrelazamiento en el algoritmo de Deutsch porque un qubit no puede entrelazarse consigo mismo. El entrelazamiento requiere al menos dos qubits.

Consultando al oráculo

En el circuito no económico, si la entrada al primer qubit de U_f es $|x\rangle$, U_f retorna $f(x)$ cuando realizamos una medición del segundo qubit. En el circuito económico, obtenemos $f(x)$ en el sentido clásico usando el circuito



De hecho, los pasos de este circuito son

$$|x\rangle|0\rangle \xrightarrow{I\otimes H} |x\rangle|+\rangle \xrightarrow{U'_f} \frac{|x\rangle|0\rangle + (-1)^{f(x)}|x\rangle|1\rangle}{\sqrt{2}} \xrightarrow{I\otimes H} |x\rangle|f(x)\rangle.$$

Para calcular el último paso, usamos que $f(x)$ es 0 o 1 para un x fijo. En primer lugar, suponemos que $f(x) = 0$, luego $I \otimes H$ se aplica a $|x\rangle|+\rangle$ dando como resultado $|x\rangle|f(x)\rangle$, y en segundo lugar suponemos que $f(x) = 1$, luego $I \otimes H$ se aplica a $|x\rangle|-\rangle$ dando como resultado $|x\rangle|f(x)\rangle$. Después de medir el segundo qubit en la base computacional, obtenemos $f(x)$ con probabilidad 1.

Capítulo 4

Algoritmo Deutsch-Jozsa

El algoritmo de Deutsch-Jozsa es un algoritmo cuántico determinístico, es una generalización del algoritmo de Deutsch y el primer ejemplo que es exponencialmente más rápido que su algoritmo determinístico clásico equivalente. Fue publicado en el año 1992 [19] y revisado en el año 1998 [14]. Muchos libros [30, 33, 35, 37, 42, 62] y artículos [40, 49, 50] han revisado y generalizado este algoritmo.

4.1. Formulación del problema

Sea $f : \{0,1\}^n \longrightarrow \{0,1\}$ una función booleana de n-bits, $n \geq 2$, con la siguiente propiedad: O bien f está balanceada o es constante. Una función booleana está balanceada si la imagen inversa del punto 0 tiene cardinalidad 2^{n-1}, y es constante si la imagen inversa del punto 0 tiene cardinalidad 0 o 2^n. Supongamos que podemos evaluar esta función en cualquier punto del dominio; sin embargo, no tenemos acceso a los detalles de la implementación de f, es decir, se nos da una computadora cuántica de *caja negra* con f ya implementado. El algoritmo de Deutsch-Jozsa resuelve el siguiente problema: Determina si f está balanceada o constante usando esta computadora cuántica de caja negra.

Solo hay dos funciones constantes, que son $f(x) =$ Falso y $f(x) =$ Verdadero para toda la cadena de de n-bits x, pero hay muchas funciones equilibradas. El mejor algoritmo determinístico clásico que resuelve este problema, dada una función de caja negra f que es balanceada o constante e implementada en una computadora clásica de n-bits, es el siguiente: Evaluar f en $2^{n-1}+1$ puntos distintos en su dominio y verificar si la salida es siempre la misma (f es constante) o no (f es balanceada).

El algoritmo de Deutsch-Jozsa, por otro lado, es un algoritmo cuántico determinístico que usa un operador unitario de caja negra U_f sólo una vez. La acción de U_f sobre la base computacional es

$$U_f|x\rangle|j\rangle = |x\rangle|j \oplus f(x)\rangle,$$

donde $x \in \{0,1\}^n$ y $j \in \{0,1\}$. Los qubits se dividen en dos registradores cuánticos con tamaños n y 1.[1] Después de hacer una superposición de vectores $|x\rangle$ para todos los x, el algoritmo Deutsch-Jozsa puede calcular $f(x)$ para todos los x con una sola aplicación de U_f y después de un rápido procesamiento posterior puede determinar si f es constante o equilibrada.

[1] Un registrador cuántico es un conjunto de qubits.

En la siguiente proposición mostramos que U_f es unitaria. Entonces, dado que cada entrada de U_f es 0 o 1, U_f es una matriz de permutación 2^{n+1}-dimensional.[2]

Proposición 4.1. U_f es unitaria para cualquier función booleana de n-bits.

Demostración. Mostramos que $U_f^\dagger U_f = I$. Usando la definición de U_f, tenemos

$$\begin{aligned}\left\langle x'\right|\left\langle j'\right|U_f^\dagger U_f\left|x\right\rangle\left|j\right\rangle &= \left\langle x'\middle|x\right\rangle \left\langle j'\oplus f(x')\middle|j\oplus f(x)\right\rangle \\ &= \delta_{xx'}\left\langle j'\oplus f(x')\middle|j\oplus f(x)\right\rangle.\end{aligned}$$

Usando ese $\delta_{xx'} \neq 0$ sólo si $x = x'$, tenemos $\delta_{xx'}\left\langle j'\oplus f(x')\middle|j\oplus f(x)\right\rangle = \delta_{xx'}\left\langle j'\middle|j\right\rangle$. Entonces

$$\left\langle x'\right|\left\langle j'\right|U_f^\dagger U_f\left|x\right\rangle\left|j\right\rangle = \delta_{xx'}\delta_{jj'}.$$

Dado que j, j' son bits arbitrarios y x, x' cadenas de n bits arbitrarias, la prueba está completa. □

4.2. El algoritmo cuántico

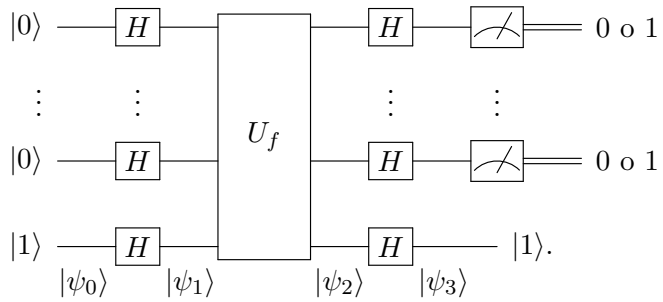
Algoritmo 4.1: Algoritmo Deutsch-Jozsa

Entrada: Una caja negra U_f que implementa una función booleana de n-bits $f : \{0,1\}^n \longrightarrow \{0,1\}$, que es balanceada o constante.

Salida: 0 si f es constante; de lo contrario, f está equilibrada.

1 Preparar el estado inicial $|0\rangle^{\otimes n}|1\rangle$;
2 Aplicar $H^{\otimes(n+1)}$;
3 Aplicar U_f;
4 Aplicar $H^{\otimes(n+1)}$;
5 Medir el primer registrador en la base computacional.

El algoritmo Deutsch-Jozsa se describe en el Algoritmo 4.1 y el circuito de $(n+1)$-qubits es



Comprobamos fácilmente que el circuito corresponde exactamente a los pasos del Algoritmo 4.1. La salida es la cadena de n-bits, 0 si f es constante, y diferente de 0 si f está balanceada. Tenga en cuenta que la última puerta de Hadamard aplicada al último qubit se puede eliminar sin afectar el algoritmo. Esta puerta está aquí porque la parte central del circuito es simétrica y el análisis del algoritmo es más claro con ella que sin ella. Los estados en la parte inferior del circuito se utilizan en el análisis del algoritmo.

[2] Una *matriz de permutación* es una matriz binaria cuadrada tal que cada fila y cada columna tiene exactamente una entrada igual a 1 y ceros en otros lugares.

4.3. Análisis del algoritmo

Después del primer paso, el estado de la computadora cuántica es

$$|\psi_0\rangle = |0\rangle^{\otimes n} \otimes |1\rangle.$$

Después del segundo paso, el estado de la computadora cuántica es

$$\begin{aligned} |\psi_1\rangle &= (H|0\rangle)^{\otimes n} \otimes (H|1\rangle) \\ &= \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle|-\rangle, \end{aligned}$$

donde $|-\rangle = (|0\rangle - |1\rangle)/\sqrt{2}$. Después del tercer paso, el estado de la computadora cuántica es

$$\begin{aligned} |\psi_2\rangle &= U_f|\psi_1\rangle \\ &= \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} U_f\big(|x\rangle|-\rangle\big). \end{aligned}$$

Para simplificar $|\psi_2\rangle$, usamos la Proposición 3.1 de la Página 29, que establece que

$$U_f\big(|x\rangle|-\rangle\big) = (-1)^{f(x)}|x\rangle|-\rangle.$$

Obtenemos

$$|\psi_2\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} (-1)^{f(x)}|x\rangle|-\rangle.$$

Después del cuarto paso, el estado de la computadora cuántica es

$$\begin{aligned} |\psi_3\rangle &= H^{\otimes(n+1)}|\psi_2\rangle \\ &= \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} (-1)^{f(x)}\big(H^{\otimes n}|x\rangle\big) \otimes \big(H|-\rangle\big). \end{aligned}$$

Para simplificar $|\psi_3\rangle$, mostremos la siguiente proposición:

Proposición 4.2. Sea $x \in \{0,1\}^n$ una cadena de n-bits $x_0 \cdots x_{n-1}$. Entonces

$$H^{\otimes n}|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} (-1)^{x\cdot y}|y\rangle,$$

donde $x \cdot y = x_0y_0 + \cdots + x_{n-1}y_{n-1} \mod 2$.

Demostración. Usando $x = (x_0 \cdots x_{n-1})_2$, obtenemos

$$\begin{aligned} H^{\otimes n}|x\rangle &= \big(H|x_0\rangle\big) \otimes \cdots \otimes \big(H|x_{n-1}\rangle\big) \\ &= \Big(\frac{1}{\sqrt{2}} \sum_{y_0=0}^{1} (-1)^{x_0y_0}|y_0\rangle\Big) \otimes \cdots \otimes \Big(\frac{1}{\sqrt{2}} \sum_{y_{n-1}=0}^{1} (-1)^{x_{n-1}y_{n-1}}|y_{n-1}\rangle\Big). \end{aligned}$$

Colocando todas las sumas al principio, obtenemos

$$H^{\otimes n}|x\rangle \;=\; \frac{1}{\sqrt{2^n}} \sum_{y_0,\ldots,y_{n-1}=0}^{1} (-1)^{(x_0y_0+\cdots+x_{n-1}y_{n-1})} |y_0\rangle \otimes \cdots \otimes |y_{n-1}\rangle.$$

Usando la definición de $x \cdot y$ y convirtiéndola a la notación decimal, completamos la demostración. □

Usando la proposición anterior, simplificamos $|\psi_3\rangle$ a

$$|\psi_3\rangle \;=\; \frac{1}{2^n} \sum_{y=0}^{2^n-1} \left(\sum_{x=0}^{2^n-1} (-1)^{x\cdot y+f(x)} \right) |y\rangle \otimes |1\rangle.$$

La amplitud del estado $|0\rangle|1\rangle$ (0 en decimal) es

$$\frac{1}{2^n} \sum_{x=0}^{2^n-1} (-1)^{f(x)}.$$

La probabilidad de que una medición del primer registrador devuelva $y = 0$ (en decimal) es

$$p(0) \;=\; \left| \frac{1}{2^n} \sum_{x=0}^{2^n-1} (-1)^{f(x)} \right|^2.$$

Si f es constante, $p(0) = 1$ y sabemos con certeza que la salida es $y = 0$. Si f está balanceada, $p(0) = 0$ y sabemos con certeza que la salida es $y \neq 0$.

4.4. Análisis del entrelazamiento

No hay entrelazamiento entre los registradores, porque $|\psi_0\rangle$ a $|\psi_3\rangle$ se pueden escribir como $|\psi\rangle \otimes |1\rangle$ o $|\psi\rangle \otimes |-\rangle$: para algún estado $|\psi\rangle$. Solo tenemos que comprobar el primer registrador. Del circuito del algoritmo, nos damos cuenta de que el único operador que crea o destruye entrelazamiento es U_f. Entonces, basta con analizar

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

que es el estado del primer registrador después de aplicar U_f. Nos preguntamos si hay a_i y b_i para que

$$|\psi\rangle \;=\; \frac{a_0|0\rangle + b_0|1\rangle}{\sqrt{2}} \otimes \cdots \otimes \frac{a_{n-1}|0\rangle + b_{n-1}|1\rangle}{\sqrt{2}},$$

donde cada par (a_i, b_i) debe obedecer a que $|a_i|^2 + |b_i|^2 = 2$. Esta es la única manera de no tener entrelazamientos en absoluto. De manera equivalente, nos preguntamos si el sistema de

ecuaciones

$$\begin{aligned} a_0...a_{n-2}a_{n-1} &= (-1)^{f(0...00)} \\ a_0...a_{n-2}b_{n-1} &= (-1)^{f(0...01)} \\ a_0...b_{n-2}a_{n-1} &= (-1)^{f(0...10)} \\ a_0...b_{n-2}b_{n-1} &= (-1)^{f(0...11)} \\ &\vdots \\ b_0...b_{n-2}b_{n-1} &= (-1)^{f(1...11)} \end{aligned}$$

admite solución o no. Es sencillo comprobar que $a_i \neq 0$ y $b_i \neq 0$ para todos los i. Además, no debemos preocuparnos por las fases de a_i. De hecho, sin pérdida de generalidad, consideramos a_i real y positivo porque se puede descartar un factor global. Entonces, seleccionando la ecuación

$$b_0 a_1 a_2 ... a_{n-2} a_{n-1} = (-1)^{f(1...00)},$$

concluimos que b_0 también es real. Lo mismo se aplica al otro b_i. Ahora, mostremos que $a_i = 1$ y $b_i = \pm 1$. Comencemos con a_0 y b_0 seleccionando las siguientes ecuaciones:

$$\begin{aligned} a_0 a_1 a_2 ... a_{n-2} a_{n-1} &= (-1)^{f(0...00)}, \\ b_0 a_1 a_2 ... a_{n-2} a_{n-1} &= (-1)^{f(1...00)}. \end{aligned}$$

Dividiéndolos, obtenemos $a_0 \pm b_0 = 0$. Este resultado junto con la restricción de $a_0^2 + b_0^2 = 2$ implica que $a_0 = 1$ y $b_0 = \pm 1$. Lo mismo se aplica a los otros a_i y b_i. Ahora, contando el número de estados no entrelazados, obtenemos como máximo 2^n, hasta un signo global. O, como mucho, 2^{n+1}.

Solo hay dos funciones constantes: $f(x) = 0$ y $f(x) = 1$ para todo x, que corresponden a $|\psi\rangle = \big(H|0\rangle\big)^{\otimes n}$ y $|\psi\rangle = -\big(H|0\rangle\big)^{\otimes n}$, respectivamente. En ambos casos, no hay entrelazamiento. Entonces, contemos el número de funciones balanceadas. El número exacto es el número de subconjuntos con cardinalidad 2^{n-1}, porque tan pronto como encontramos dicho subconjunto S, definimos una función balanceada tomando $f(x) = 0$ si $x \in S$ y $f(x) = 1$ en caso contrario. Cuando cubrimos todos esos subconjuntos, hemos obtenido todas las funciones balanceadas. Dado que el dominio tiene cardinalidad 2^n, la cantidad de funciones balanceadas es $\binom{2^n}{2^{n-1}}$.[3]

El número de funciones balanceadas crece más rápido que el número de estados no entrelazados. De hecho, usando la aproximación asintótica[4]

$$\binom{2p}{p} \approx \frac{2^{2p}}{\sqrt{\pi p}}$$

obtenemos

$$\binom{2^n}{2^{n-1}} \approx \frac{\sqrt{2}}{\sqrt{\pi}} \frac{2^{2^n}}{\sqrt{2^n}}.$$

Concluimos que cuando n aumenta, hay cada vez más funciones f equilibradas con U_f creando entrelazamiento.

[3] Una forma alternativa de contar la cantidad de funciones balanceadas es considerar la cantidad de permutaciones de una lista con 2^{n-1} ceros y 2^{n-1} unos.

[4] https://en.wikipedia.org/wiki/Binomial_coficient

La discusión anterior muestra que hay $n_0 \geq 2$, de modo que para todos los $n \geq n_0$, U_f crea un entrelazamiento. Dado que $\binom{2^n}{2^{n-1}} > 2^{n+1}$ por $n \geq 3$, podemos tomar $n_0 = 3$. El único caso restante que puede no tener entrelazamiento es $n = 2$. Para $n = 2$, hay 6 funciones balanceadas, cuyas tablas de verdad se describen en la Fig. 4.1 con los nombres de función f_0 a f_5.

x_0	x_1	$f_0(x)$	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	$f_5(x)$
0	0	0	0	0	1	1	1
0	1	0	1	1	0	0	1
1	0	1	0	1	0	1	0
1	1	1	1	0	1	0	0

Figura 4.1: Tablas de verdad de todas las funciones balanceadas de 2-bits.

Basta analizar f_0 a f_2 porque las demás funciones son el complemento de aquellas. Por ejemplo, f_5 es el complemento de f_0, lo que significa que el estado $|\psi\rangle$ creado por U_{f_5} es igual al estado creado por U_{f_0} hasta una fase global. Los estados $|\psi\rangle$ que corresponden a f_0 a f_2 (sin normalización) son

$$|00\rangle + |01\rangle - |10\rangle - |11\rangle = \big(|0\rangle - |1\rangle\big) \otimes \big(|0\rangle + |1\rangle\big),$$
$$|00\rangle - |01\rangle + |10\rangle - |11\rangle = \big(|0\rangle + |1\rangle\big) \otimes \big(|0\rangle - |1\rangle\big),$$
$$|00\rangle - |01\rangle - |10\rangle + |11\rangle = \big(|0\rangle - |1\rangle\big) \otimes \big(|0\rangle - |1\rangle\big),$$

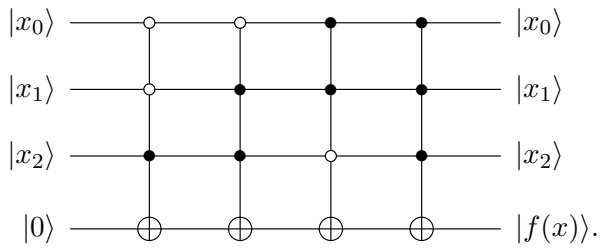
respectivamente. Todos esos estados están desentrelazados. Concluimos que no hay entrelazamiento en el algoritmo de Deutsch-Jozsa cuando $n = 2$.

4.5. Implementando el oráculo

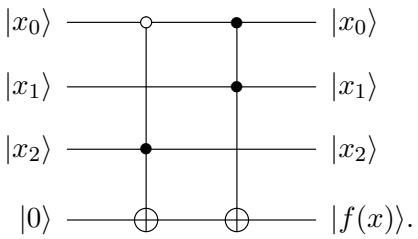
Repetimos aquí lo que ya se ha escrito antes sobre el oráculo. No depende de nosotros implementar el oráculo. Es el trabajo de otra persona. Sin esta comprensión, nos enfrentamos a una contradicción: tenemos que saber la respuesta incluso antes de comenzar a implementar el algoritmo que encontrará la respuesta. Se nos permite evaluar la función f implementada por alguien sin mirar los detalles de su implementación. Nos dan lo que se llama una computadora cuántica de *caja negra* con U_f ya implementado, que podemos usar, agregar nuevas puertas; pero no podemos ver el interior. En el caso clásico, tenemos que contar el número de evaluaciones de f. En el caso cuántico, tenemos que contar el número de aplicaciones de U_f. Así es como calculamos la complejidad de la consulta de los algoritmos basados en el oráculo. Tenga en cuenta que no importa si la evaluación de f es eficiente o no.

Con este entendimiento, mostremos como implementar el oráculo U_f para la siguiente función balanceada entrelazada: $f(x) = 0$ si $x \in \{000, 010, 100, 101\}$ y $f(x) = 1$ si $x \in \{001, 011, 110, 111\}$. Usando la forma normal disyuntiva, agregamos al circuito una puerta de Toffoli multiqubit para

cada punto en $\{001, 011, 110, 111\}$, de la siguiente manera



Dado que la primera y la segunda puerta tienen controles opuestos en el segundo qubit y controles idénticos en el primer y tercer qubit (la tercera y la cuarta puerta tienen una característica similar), este circuito se puede simplificar a



Este ejemplo ayuda a mostrar como implementar cualquier oráculo balanceado usando $n+1$-qubits. Es posible implementar el algoritmo de Deutsch-Jozsa con solo n-qubits, pero esta discusión se pospone y se aborda en detalle en el Capítulo 9.

4.6. Observaciones finales

Existen algoritmos clásicos aleatorios eficientes que resuelven el problema de Deutsch-Jozsa. Es posible encontrar la solución correcta con alta probabilidad consultando el oráculo clásico varias veces. Más formalmente, considere el siguiente algoritmo aleatorio: (1) Seleccione uniformemente al azar $k \geq 2$ puntos x_0, ..., x_{k-1} en el dominio, se permiten repeticiones, (2) si $f(x_0) = \cdots = f(x_{k-1})$, devuelva "$f$ es constante"; de lo contrario, devuelva "f está equilibrada". Este algoritmo devuelve la salida "f está balanceada" con certeza porque tan pronto como obtenemos dos valores diferentes después de evaluar f, reclamamos la promesa de que f está balanceada o constante; debe estar balanceada. La probabilidad de que la salida "f sea constante" sea correcta es $1 - 1/2^{k-1}$.[5] La probabilidad de éxito tiende rápidamente a 1, por ejemplo, tome $k = 10$, la probabilidad de éxito es al menos el 99.8 %.

[5]Para calcular la probabilidad de que "f sea constante" sea correcta, comenzamos calculando la probabilidad de que "f sea constante" es equivocada. La salida "f es constante" es equivocada cuando f está balanceada y $f(x_0) = \cdots = f(x_{k-1})$. La probabilidad de que $f(x_0) = \cdots = f(x_{k-1}) = 0$ sea $1/2^k$ porque cada evaluación es independiente. Asimismo, la probabilidad de que $f(x_0) = \cdots = f(x_{k-1}) = 1$ sea $1/2^k$. Entonces, la probabilidad de que "f sea constante" sea incorrecta es $2/2^k$ porque todos los resultados iguales a 0 y todos los resultados iguales a 1 son mutuamente excluyentes. Entonces, la probabilidad de que "f sea constante" sea correcta es $1 - 1/2^{k-1}$.

Capítulo 5

Algoritmo de Bernstein-Vazirani

El algoritmo de Bernstein-Vazirani se presentó en una conferencia en el año 1993 [8], y el artículo completo se publicó en el año 1997 [9], siendo el primer algoritmo cuántico determinístico con ganancia lineal sobre el mejor algoritmo clásico aleatorio o determinístico. El algoritmo de Bernstein-Vazirani explota el paralelismo cuántico pero no tiene ningún entrelazamiento. Este algoritmo se describe en algunos libros [37, 41, 51].

5.1. Formulación del problema

Sea $s = s_0...s_{n-1}$ una cadena de n-bits, desconocida. Aunque no conocemos s, tenemos a nuestra disposición una función booleana $f : \{0,1\}^n \longrightarrow \{0,1\}$ definida como

$$f(x) = s \cdot x = s_0x_0 + ... + s_{n-1}x_{n-1} \quad \text{mód } 2,$$

donde $x_0, ..., x_{n-1}$ son los bits de x. Tenga en cuenta que f es una función lineal y cada función booleana lineal se caracteriza por un s específico. Nuestro objetivo es encontrar s evaluando f sin conocer los detalles de su implementación. En computación cuántica, el operador que implementa esta función se llama *oráculo*; como si un oráculo revelara $f(x)$ sin mostrar s explícitamente, y $f(x)$ pudiera usarse para determinar s. Cuando construimos el circuito del algoritmo, f es implementado por otra persona, porque no conocemos s. Es importante entender esto; de lo contrario, tenemos la impresión de que necesitamos saber la respuesta para encontrar la respuesta, lo cual es absurdo.

En la versión clásica de este problema, tenemos que consultar el *oráculo clásico* al menos n veces. De hecho, elegimos $x = 10..,0$ y le preguntamos al oráculo qué es $f(10..,0)$. La respuesta es s_0. A continuación, elegimos $x = 010..,0$ y le preguntamos al oráculo qué es $f(010..,0)$. La respuesta es s_1. La última consulta es $f(0..,01)$, cuya respuesta es s_{n-1}. Esto muestra que necesitamos consultar el oráculo n veces y no hay forma de reducir este número sin introducir un error en el algoritmo. En el caso cuántico, consultamos el *oráculo cuántico* solo una vez, lo que nos permite encontrar todos los bits de s, como se describe a continuación.

En el caso cuántico, la función $f(x) = s \cdot x$ se implementa utilizando el operador unitario U_f de $n + 1$ qubits, definido como

$$U_f|x\rangle|j\rangle = |x\rangle|j \oplus f(x)\rangle,$$

donde $x \in \{0,1\}^n$, j es un bit, y $\oplus$ es la operación XOR o suma módulo 2. Este operador utiliza dos registradores, con tamaños n y 1, respectivamente. Podemos usar U_f tantas veces como queramos. Sin embargo, se usa sólo una vez en el algoritmo de Bernstein-Vazirani.

En resumen, el algoritmo clásico consulta el oráculo clásico n veces usando una computadora clásica de n-bits. El algoritmo cuántico consulta el oráculo cuántico solo una vez utilizando una computadora cuántica de $(n+1)$-qubits. En la última Sección de este Capítulo, mostramos que el algoritmo se puede implementar en una computadora cuántica de n-qubits.

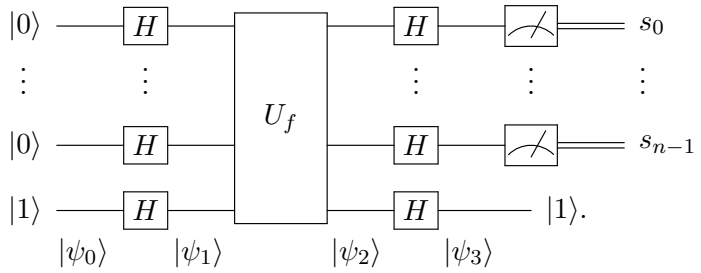
5.2. El algoritmo

Algoritmo 5.1: Algoritmo de Bernstein-Vazirani

Entrada: Una función booleana $f : \{0,1\}^n \longrightarrow \{0,1\}$ tal que $f(x) = s \cdot x$.
Salida: s con probabilidad igual a 1.

1 Preparar el estado inicial $|0\rangle^{\otimes n}|1\rangle$;
2 Aplicar $H^{\otimes(n+1)}$;
3 Aplicar U_f;
4 Aplicar $H^{\otimes(n+1)}$;
5 Medir el primer registrador en la base computacional.

El algoritmo de Bernstein-Vazirani se describe en el Algoritmo 5.1 y el circuito es



Tenga en cuenta que este circuito es igual al circuito del algoritmo de Deutsch-Jozsa, la única diferencia es la función utilizada en el algoritmo de Bernstein-Vazirani, que no necesita ser constante ni balanceada. Además, en el algoritmo de Deutsch-Jozsa comprobamos si todas las salidas son 0 para concluir que la función es constante; por otro lado, equilibrado. En el algoritmo de Bernstein-Vazirani, cada salida es valiosa para determinar la incógnita s. Los estados en la parte inferior del circuito se utilizan en el análisis del algoritmo.

5.3. Análisis del algoritmo

Después del primer paso, el estado de la computadora cuántica es

$$|\psi_0\rangle = |0\rangle^{\otimes n}|1\rangle.$$

Después del segundo paso, el estado de la computadora cuántica es

$$\begin{aligned} |\psi_1\rangle &= (H^{\otimes n}|0\rangle^{\otimes n}) \otimes (H|1\rangle) \\ &= \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \otimes |-\rangle, \end{aligned}$$

donde $|-\rangle = (|0\rangle - |1\rangle)/\sqrt{2}$ y x se escriben en notación decimal. Después del tercer paso, el estado de la computadora cuántica es

$$\begin{aligned} |\psi_2\rangle &= U_f|\psi_1\rangle \\ &= \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} U_f|x\rangle \otimes |-\rangle. \end{aligned}$$

Para simplificar $|\psi_2\rangle$, usamos la Proposición 3.1 de la Página 29, que establece que

$$U_f\big(|x\rangle|-\rangle\big) = (-1)^{f(x)}|x\rangle|-\rangle.$$

Obtenemos

$$|\psi_2\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} (-1)^{s\cdot x}|x\rangle \otimes |-\rangle.$$

Después del cuarto paso, el estado de la computadora cuántica es

$$\begin{aligned} |\psi_3\rangle &= H^{\otimes(n+1)}|\psi_2\rangle \\ &= H^{\otimes n}\left(\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} (-1)^{s\cdot x}|x\rangle\right) \otimes (H|-\rangle). \end{aligned} \tag{5.1}$$

Para simplificar $|\psi_3\rangle$, usamos la Proposición 4.2 de la Página 35, que establece que para cualquier $s = (s_0...s_{n-1})_2$

$$H^{\otimes n}|s\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} (-1)^{s\cdot x}|x\rangle,$$

donde $s \cdot x = s_0x_0 + \cdots + s_{n-1}x_{n-1}$ mód 2. Luego usamos $H^2 = I$ para obtener

$$|s\rangle = H^{\otimes n}\left(\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} (-1)^{s\cdot x}|x\rangle\right).$$

Reemplazamos este resultado en la Ec. (5.1) para obtener

$$|\psi_3\rangle = |s\rangle \otimes |1\rangle.$$

La salida de la medición del primer registrador es $s_0, ..., s_{n-1}$ con probabilidad 1 porque $|s\rangle = |s_0\rangle \otimes \cdots \otimes |s_{n-1}\rangle$.

U_f se aplica una sola vez, por lo que decimos que se realizó una única consulta al oráculo cuántico. Tenga en cuenta que U_f se aplica a una superposición de todos los vectores de la base computacional, por lo tanto, f se evalúa simultáneamente en todos los puntos del dominio; todas las cadenas de n-bits . El resultado de esta evaluación masiva es un estado de superposición, que es inútil en muchos casos. No en el algoritmo de Bernstein-Vazirani porque la aplicación de $H^{\otimes n}$ al primer registrador al final revela s. Antes del último paso, s era una fase. Después, s entró en el núcleo de un "ket". Esto hace toda la diferencia porque una fase está asociada con la probabilidad de obtener un determinado resultado, mientras que el núcleo del "ket" está asociado con el resultado de la medición.

5.4. El algoritmo de Bernstein-Vazirani no tiene entrelazamiento

El análisis del entrelazamiento sigue al principio la misma pista utilizada en el algoritmo de Deutsch-Jozsa. Del circuito del algoritmo, nos damos cuenta de que el único operador que crea o destruye entrelazamiento es U_f. Entonces, basta con analizar $|\psi_2\rangle$. El algoritmo de Bernstein-Vazirani tiene entrelazamiento sí y solo sí $|\psi_2\rangle$ está total o parcialmente entrelazado. El estado $|\psi_2\rangle$ viene dado por

$$|\psi_2\rangle \;=\; \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} (-1)^{s\cdot x} |x\rangle \otimes |-\rangle.$$

La Proposición 4.2 de la Página 35 establece que para cualquier $s = (s_0 ... s_{n-1})_2$

$$H^{\otimes n}|s\rangle \;=\; \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} (-1)^{s\cdot x} |x\rangle.$$

Entonces $|\psi_2\rangle$ se puede escribir como

$$|\psi_2\rangle \;=\; \big(H^{\otimes n}|s\rangle\big) \otimes |-\rangle,$$

que se puede factorizar completamente como

$$|\psi_2\rangle \;=\; \big(H|s_0\rangle\big) \otimes \cdots \otimes \big(H|s_{n-1}\rangle\big) \otimes |-\rangle.$$

Tenga en cuenta que $|\psi_2\rangle$ no tiene ningún entrelazamiento porque es el producto de Kronecker de estados puros de 1 qubit [38, 21].

5.5. Circuito del oráculo

En los algoritmos basados en un oráculo, el oráculo no lo implementamos nosotros, lo implementa otra persona. Sin embargo, es importante saber cómo se hace para entender todo el proceso. Pongamos un ejemplo con $n = 4$ y $s = 1011$ que será suficiente para entender el caso general. En este caso particular, f es

$$f(x) = s \cdot x = x_0 + x_2 + x_3 \mod 2$$

y la acción de U_f sobre un vector arbitrario de la base computacional $|x\rangle|j\rangle$ es

$$U_f|x_0x_1x_2x_3\rangle|j\rangle = |x_0x_1x_2x_3\rangle|j \oplus f(x)\rangle = |x_0x_1x_2x_3\rangle|j \oplus x_0 \oplus x_2 \oplus x_3\rangle.$$

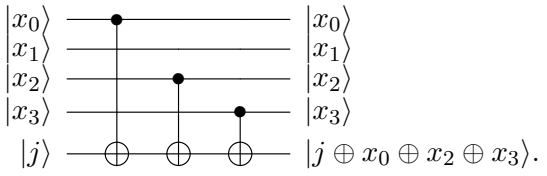
Sea CNOT_{04} la puerta CNOT que actúa sobre los qubits 0 y 4. Sin mostrar los qubits 1, 2 y 3, tenemos

$$\mathrm{CNOT}_{04}|x_0\rangle|j\rangle = |x_0\rangle X^{x_0}|j\rangle = |x_0\rangle|j \oplus x_0\rangle.$$

Tenga en cuenta que si usamos CNOT_{04}, CNOT_{24} y CNOT_{34}, generamos el resultado esperado $j \oplus x_0 \oplus x_2 \oplus x_3$ en el segundo registrador. Después,

$$U_f \;=\; \mathrm{CNOT}_{34} \cdot \mathrm{CNOT}_{24} \cdot \mathrm{CNOT}_{04},$$

cuyo circuito es

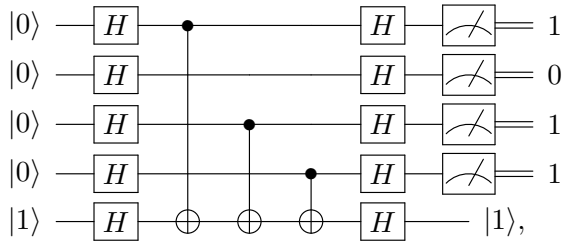


A partir de este ejemplo, podemos generalizar la implementación a un $s = s_0 \cdots s_{n-1}$ arbitrario. sólo se considera

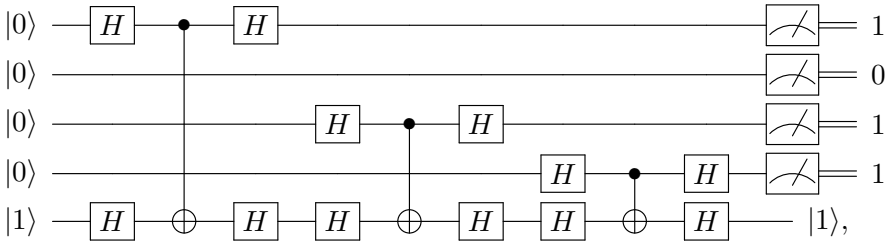
$$U_f \;=\; (\mathrm{CNOT}_{0n})^{s_0}\,(\mathrm{CNOT}_{1n})^{s_1}\,\cdots\,(\mathrm{CNOT}_{n-1,n})^{s_{n-1}}\,,$$

donde CNOT_{ij} está controlado por el qubit i y el objetivo es el qubit j, y $(\mathrm{CNOT}_{ij})^{s_k}$ es el operador de identidad si $s_k = 0$, y el estándar CNOT_{ij} si $s_k = 1$. Tenga en cuenta que el orden de las puertas CNOT es irrelevante. Hay un CNOT para cada bit 1 de s.

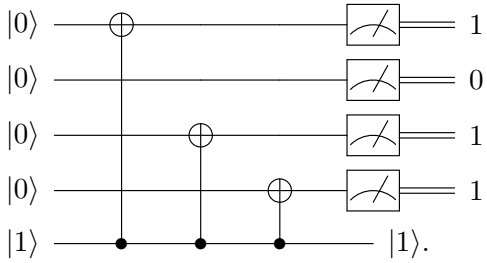
Terminemos el circuito del algoritmo de Bernstein-Vazirani para nuestro ejemplo. Todo el circuito es



que es equivalente a



porque $H^2 = I$. Usando $(H \otimes H) \cdot \mathrm{CNOT}_{ij} \cdot (H \otimes H) = \mathrm{CNOT}_{ji}$, el último circuito se simplifica a



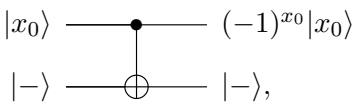
De este ejemplo, podemos derivar fácilmente el caso genérico, que siempre se puede expresar como CNOTs controlados por el último qubit y los objetivos son los qubits correspondientes a los bits de s que son iguales a 1. Esta simplificación ayuda a entender que el algoritmo de Bernstein-Vazirani no tiene entrelazamiento porque U_f se multiplicó por $H^{\otimes(n+1)}$, que no puede crear ni destruir el entrelazamiento. La representación de U_f en el circuito del algoritmo nos da la impresión de que U_f crea un entrelazamiento, pero eso es engañoso.

5.6. Circuito económico del algoritmo de Bernstein-Vazirani

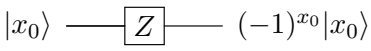
El algoritmo de Bernstein-Vazirani se puede implementar con n-qubits en lugar de $n+1$. De hecho, hemos aprendido que U_f es un producto de CNOTs

$$U_f \;=\; (\mathrm{CNOT}_{0n})^{s_0} \cdots (\mathrm{CNOT}_{n-1,n})^{s_{m-1}}\,,$$

donde el CNOT_{ij} está controlado por el qubit i y el objetivo es el qubit j, y el $(\mathrm{CNOT}_{ij})^{s_k}$ es la identidad si $s_k = 0$, y el CNOT_{ij} si $s_k = 1$. En el algoritmo, justo antes de la acción de U_f, el estado del n-ésimo qubit es $|-\rangle$. El circuito que describe la acción del CNOT_{0n} es



donde hemos representado sólo el primer y el último qubit y hemos colocado $(-1)^{x_0}$ en el primer qubit porque esto está matemáticamente permitido. Este circuito es equivalente a



Podemos convertir todos los CNOT de U_f en Z y luego

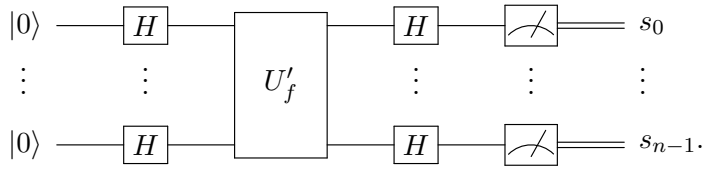
$$U'_f \;=\; Z^{s_0} \otimes \cdots \otimes Z^{s_{n-1}}.$$

Tenga en cuenta que si un bit s_i de s es 0, $Z^{s_i} = I$. Entonces

$$U'_f|x\rangle \;=\; (-1)^{s\cdot x}|x\rangle.$$

La función f es la misma que antes. Lo que cambia es la forma en que implementamos f como un operador unitario.

El circuito de la versión económica del algoritmo de Bernstein-Vazirani es



Como antes, este circuito se simplifica a

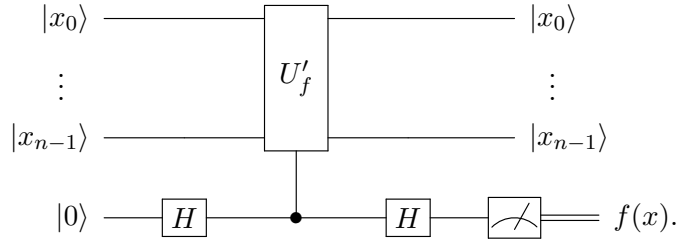
$$H^{\otimes n} U'_f H^{\otimes n} \;=\; X^{s_0} \otimes \cdots \otimes X^{s_{n-1}}$$

porque $HZH = X$. Ahora, es sencillo verificar que no haya entrelazamientos en el algoritmo de Bernstein-Vazirani. U'_f no es una puerta de n-qubits genuina, sino que es el producto tensorial de n puertas de 1 qubit. La representación del circuito es engañosa.

Consultando al oráculo

En el circuito no económico, si la entrada al primer registro de U_f es $|x\rangle$, U_f retorna $f(x) = (-1)^{s\cdot x}$ cuando realizamos una medición del último qubit. En el circuito económico, obtenemos

$f(x)$ usando el circuito



De hecho, los pasos de este circuito son

$$|x\rangle|0\rangle \xrightarrow{I\otimes H} |x\rangle|+\rangle \xrightarrow{U'_f} \frac{|x\rangle|0\rangle + (-1)^{f(x)}|x\rangle|1\rangle}{\sqrt{2}} \xrightarrow{I\otimes H} |x\rangle|f(x)\rangle,$$

donde I es el operador 2^n-dimensional identidad . Para calcular el último paso, sólo hay casos en los que arreglamos x: $f(x) = 0$ o $f(x) = 1$. En primer lugar, suponemos que $f(x) = 0$, luego $I \otimes H$ se aplica a $|x\rangle|+\rangle$ dando como resultado $|x\rangle|f(x)\rangle$, y en segundo lugar suponemos que $f(x) = 1$, luego $I \otimes H$ se aplica a $|x\rangle|-\rangle$ dando como resultado $|x\rangle|f(x)\rangle$. Después de realizar una medición del último qubit en la base computacional, obtenemos $f(x)$ con probabilidad 1.

Capítulo 6

El problema de Simon

El problema de Simon se presentó en una conferencia en el año 1994 [57], junto con los algoritmos de Shor [54]; el artículo completo se publicó en 1997 [58]. Es un algoritmo cuántico exponencialmente más rápido que el mejor algoritmo clásico equivalente determinístico o aleatorio. Es una contribución científica notable, pero subestimada por la computación cuántica. El algoritmo de Simon explota no sólo el paralelismo cuántico, sino también el entrelazamiento máximo. Este algoritmo y su generalización se describen en algunos libros [30, 37, 41, 51, 64] y artículos [13, 39, 65].

6.1. Formulación del problema

Sea $f : \{0,1\}^n \longrightarrow \{0,1\}^n$ una entrada de n-bits y la función de salida de n-bits con la siguiente propiedad: Hay una cadena de bits distinta de cero $s \in \{0,1\}^n$ tal que

$$f(x) = f(y) \iff x \oplus y \in \{0, s\}$$

para todos los $x, y \in \{0,1\}^n$. Esto quiere decir que cada punto de la imagen es imagen de exactamente dos puntos del dominio, es decir, el punto $f(x)$ de la imagen tiene asociado un x y un $x \oplus s$ del dominio porque $f(x) = f(x \oplus s)$ para todo $x \in \{0,1\}^n$. Por eso, la función f está en el conjunto de funciones dos a uno. Considere el siguiente problema computacional: determine s consultando f tan pocas veces sea posible.

Nuestro objetivo es encontrar s evaluando f sin conocer los detalles de su implementación. En computación cuántica, el operador que implementa esta función se llama *oráculo*, como si un oráculo revelara $f(x)$ sin mostrar s explícitamente, y $f(x)$ se puede usar para determinar s, pero evaluar f sólo una vez no es suficiente. Cuando construimos el circuito del algoritmo, la parte asociada con f la implementa otra persona, porque no conocemos s.

En la versión clásica de este problema, tenemos que consultar un *oráculo clásico*, y para determinar s con alta probabilidad, el número de consultas a la función f crece como una función exponencial sobre el número de bits n, si usamos una computadora clásica. De hecho, un algoritmo determinístico ingenuo sería calcular $f(x')$ para un x' fijo, luego buscar sistemáticamente x en el dominio tal que $f(x) = f(x')$. Necesitamos 2^n evaluaciones en el peor de los casos. Es mejor elegir x y x' uniformemente al azar y luego verificar si $f(x) = f(x')$. Este método requiere $\Omega(\sqrt{2^n})$ evaluaciones para lograr la probabilidad de éxito $\Omega(1)$. La demostración es la misma que en el problema de colisión de dos a uno [12] o en la *paradoja del cumpleaños* [16].

En el caso cuántico, f se implementa utilizando el operador unitario U_f de $2n$ qubits, definido como

$$U_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle,$$

donde x, y y $f(x)$ son cadenas de n-bits, y $\oplus$ es la operación bit a bit XOR o bit a bit o suma módulo 2. Este operador utiliza dos registradores cada uno con n-qubits. U_f es una matriz de permutación 2^{2n}-dimensional. La prueba de que U_f es unitaria es una extensión de la prueba presentada en la Proposición 3.1 en la Página 29.

El algoritmo de Simon tiene dos partes. La parte cuántica devuelve una cadena de n-bits x, que obedece a $x \cdot s = 0$, donde

$$x \cdot s = x_0 s_0 + \cdots + x_{n-1} s_{n-1} \quad \text{mód } 2.$$

Conocer tal x no es suficiente para determinar s. Es necesario ejecutar la parte cuántica muchas veces, recopilar muchas cadenas de bits x que obedecen a $x \cdot s = 0$ y luego ejecutar la parte clásica del algoritmo, que revela s con una probabilidad mayor que $1/2$.

6.2. El algoritmo

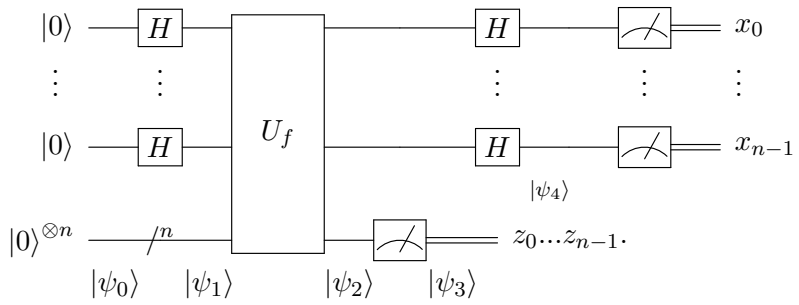
Algoritmo 6.1: Algoritmo de Simon

Entrada: Función $f : \{0,1\}^n \longrightarrow \{0,1\}^n$ con la promesa de que
$f(x) = f(y) \iff x \oplus y \in \{0, s\}$.

Salida: s con probabilidad mayor que $1/2$.

1. Ejecute la parte cuántica $n-1$ veces (Algoritmo 6.2, suponga que se obtiene $x^{(1)}, ..., x^{(n-1)}$);
2. Resolver el sistema de ecuaciones lineales $\{x^{(1)} \cdot s \equiv 0, ..., x^{(n-1)} \cdot s \equiv 0\}$ mód 2 (asumir solución para $s_0, ..., s_{n-2}$);
3. Tome $s_{n-1} = 0$ y verifique si $f(s) = f(0)$;
4. Si es verdadero, devuelve $s_0 ... s_{n-2}0$; de lo contrario, devuelve $s_0 ... s_{n-2}1$.

El algoritmo de Simon se describe en el Algoritmo 6.1 y la parte cuántica se describe en el Algoritmo 6.2. El circuito es de la parte cuántica es



Los estados en la parte inferior del circuito se utilizan en el análisis del algoritmo. Describen el estado de los qubits después de cada paso. Tenga en cuenta que el estado $|\psi_4\rangle$ se refiere solo al primer registrador. La notación "$/^n$" sobre un cable indica que es un registrador de n-qubits.

Algoritmo 6.2: Parte cuántica del Algoritmo de Simon

Entrada: Una caja negra U_f que implementa la función $f : \{0,1\}^n \longrightarrow \{0,1\}^n$ con la promesa de que $f(x) = f(y) \iff x \oplus y \in \{0, s\}$.

Salida: Punto $x \in \{0,1\}^n$ tal que $x \cdot s = 0$.

1 Preparar el estado inicial $|0\rangle^{\otimes n}|0\rangle^{\otimes n}$;
2 Aplicar $H^{\otimes n}$ al primer registrador;
3 Aplicar U_f;
4 Mida el segundo registrador en la base computacional (suponga salida $z_0...z_{n-1}$);
5 Aplicar $H^{\otimes n}$ al primer registrador;
6 Medir el primer registrador en la base computacional.

6.3. Análisis de la parte cuántica

Después del primer paso, el estado de la computadora cuántica es

$$|\psi_0\rangle = |0\rangle^{\otimes n}|0\rangle^{\otimes n}.$$

Después del segundo paso, el estado de la computadora cuántica es

$$\begin{aligned} |\psi_1\rangle &= \big(H|0\rangle\big)^{\otimes n} \otimes |0\rangle^{\otimes n} \\ &= \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \otimes |0\rangle^{\otimes n}, \end{aligned}$$

donde x se escribe en notación decimal. Después del tercer paso, el estado de la computadora cuántica es

$$\begin{aligned} |\psi_2\rangle &= U_f|\psi_1\rangle \\ &= \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} U_f(|x\rangle \otimes |0\cdots 0\rangle). \end{aligned}$$

Para simplificar $|\psi_2\rangle$, usamos la definición de U_f para obtener

$$|\psi_2\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \otimes |f(x)\rangle,$$

porque $(0\ldots0) \oplus f(x)$ (XOR bit a bit) es $f(x)$. El cuarto paso es una medición de cada qubit del segundo registrador, que suponemos que ha devuelto $z_0...z_{n-1}$. El estado $|\psi_2\rangle$ colapsa en una superposición de sólo dos términos porque sólo hay dos puntos en el dominio tales que $f(x) = z_0...z_{n-1}$. Sea x' uno de eses puntos. Entonces, $f(x') = f(x' \oplus s) = z_0...z_{n-1}$ y el estado $|\psi_3\rangle$ es

$$|\psi_3\rangle = \left(\frac{|x'\rangle + |x' \oplus s\rangle}{\sqrt{2}}\right) \otimes |z_0...z_{n-1}\rangle.$$

Tenga en cuenta que hemos vuelto a normalizar el estado $|\psi_3\rangle$ como exige el postulado de medición. El punto x' es desconocido. Se selecciona uniformemente al azar entre los puntos del

dominio. La información que deseamos adquirir, s, está oculta porque $x' \oplus s$ es un punto aleatorio en el dominio.

En el quinto paso, sólo consideramos el primer registrador. Después de aplicar $H^{\otimes n}$ al estado del primer registrador, obtenemos

$$|\psi_4\rangle \;=\; \frac{1}{\sqrt{2}}\left(H^{\otimes n}\big|x'\big\rangle + H^{\otimes n}\big|x' \oplus s\big\rangle\right).$$

Para simplificar $|\psi_4\rangle$, usamos la Proposición 4.2 de la Página 35, que establece que para cualquier $x' = (x'_0 ... x'_{n-1})_2$

$$H^{\otimes n}\big|x'\big\rangle \;=\; \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} (-1)^{x' \cdot x} |x\rangle,$$

donde $x' \cdot x = x'_0 x_0 + \cdots + x'_{n-1} x_{n-1} \mod 2$. Entonces

$$|\psi_4\rangle \;=\; \frac{1}{\sqrt{2^{n+1}}} \sum_{x=0}^{2^n-1} \left((-1)^{x' \cdot x} + (-1)^{(x' \oplus s) \cdot x}\right) |x\rangle.$$

Ahora usamos el hecho de que

$$\begin{aligned}(x' \oplus s) \cdot x &= (x'_0 + s_0)x_0 + \cdots + (x'_{n-1} + s_{n-1})x_{n-1} \mod 2 \\ &= (x'_0 x_0 + \cdots + x'_{n-1} x_{n-1}) + (s_0 x_0 + \cdots + s_{n-1} x_{n-1}) \mod 2 \\ &= (x' \cdot x) + (s \cdot x) \mod 2.\end{aligned}$$

Hemos usado $x'_0 \oplus s_0 = x'_0 + s_0 \mod 2$. El estado $|\psi_4\rangle$ se simplifica a

$$|\psi_4\rangle \;=\; \frac{1}{\sqrt{2^{n+1}}} \sum_{x=0}^{2^n-1} (-1)^{x' \cdot x} \Big(1 + (-1)^{s \cdot x}\Big) |x\rangle.$$

ahora usamos

$$1 + (-1)^{s \cdot x} = \begin{cases} 2, & \text{si } s \cdot x = 0, \\ 0, & \text{otherwise,} \end{cases}$$

para obtener

$$|\psi_4\rangle \;=\; \frac{1}{\sqrt{2^{n-1}}} \sum_{\substack{x=0 \\ s \cdot x = 0}}^{2^n-1} (-1)^{x' \cdot x} |x\rangle.$$

La suma es superior a x tal que $x \cdot s = 0$. Entonces, la salida de la medición del primer registrador es x tal que $x \cdot s = 0$ con probabilidad $|(-1)^{x' \cdot x}|^2 = 1$. Cuando ejecutamos la parte cuántica del algoritmo de Simon, obtenemos información parcial sobre s. Esto significa que tenemos que ejecutar la parte cuántica muchas veces para recopilar suficiente información para determinar s.

Tenga en cuenta que la probabilidad de obtener un x específico tal que $x \cdot s = 0$ es $1/2^{n-1}$, que es el cuadrado del valor absoluto de la amplitud del estado $|x\rangle$ en $|\psi_4\rangle$. La distribución es uniforme en todas las cadenas de n-bits x que satisfacen $x \cdot s = 0$. Por otro lado, la probabilidad de obtener un x arbitrario tal que $x \cdot s = 0$ es 1, o equivalentemente la probabilidad de obtener un x tal que $x \cdot s \neq 0$ es 0. Es decir, estamos 100 % seguros de que la salida x satisface $x \cdot s = 0$. Obtenemos información parcial sobre s a menos que $x = (0 \ldots 0)_2$.

El punto x', que obedece a $f(x') = z$, no interviene en el resultado final, ni en el cálculo final de la probabilidad de éxito. Esta es una buena noticia porque x' estaba ocultando s en algún momento. Después del quinto paso, x' se vuelve inofensivo.

6.4. Análisis de la parte clásica

Cada vez que ejecutamos la parte cuántica del algoritmo de Simon, la salida es una cadena de n-bits x, tal que $x \cdot s = 0$. Supongamos que hemos corrido dos veces el algoritmo y las salidas son x y x'. Esto significa que hemos obtenido un sistema homogéneo de ecuaciones lineales

$$\begin{aligned} x_0 s_0 + \cdots + x_{n-1} s_{n-1} &\equiv 0 \mod 2, \\ x'_0 s_0 + \cdots + x'_{n-1} s_{n-1} &\equiv 0 \mod 2, \end{aligned}$$

donde $s_0, ..., s_{n-1}$ son las variables (incógnitas) y x, x' son coeficientes binarios conocidos. Calculemos la probabilidad $p(2)$ de que el sistema sea independiente. La probabilidad de que la primera ecuación no sea trivial es

$$p_1 \;=\; 1 - \frac{1}{2^n}$$

porque hay 2^n cadenas x y sólo una es 0. Para calcular la probabilidad de que la segunda ecuación sea independiente, pensamos que x es un vector n-dimensional, en un espacio vectorial binario con 2^n vectores . El subespacio abarcado por x tiene dos vectores, el propio x y el vector nulo. Hay $2^n - 2$ vectores que son linealmente independientes de x. Entonces, la probabilidad p_2 de que la segunda ecuación sea independiente es

$$p_2 \;=\; 1 - \frac{2}{2^n}.$$

Entonces, la probabilidad $p(2)$ de que las dos ecuaciones sean independientes es

$$p(2) \;=\; p_1 p_2 \;=\; \left(1 - \frac{1}{2^n}\right)\left(1 - \frac{2}{2^n}\right).$$

La probabilidad de que la siguiente ecuación agregada al sistema sea independiente se calcula de la siguiente manera. Los vectores x y x' abarcan un subespacio con cuatro vectores: x, x', $x \oplus x'$ y el vector nulo. Hay vectores $2^n - 4$ que son linealmente independientes de x y x'. La probabilidad p_3 de que la tercera ecuación sea independiente es

$$p_3 \;=\; 1 - \frac{2^2}{2^n}.$$

Entonces, la probabilidad $p(3)$ de que las tres ecuaciones sean independientes es

$$p(3) \;=\; p_1 p_2 p_3 \;=\; \left(1 - \frac{1}{2^n}\right)\left(1 - \frac{2}{2^n}\right)\left(1 - \frac{2^2}{2^n}\right).$$

Procedemos de esta manera hasta obtener $n - 1$ ecuaciones independientes con probabilidad

$$p(n-1) \;=\; \prod_{i=1}^{n-1} p_i \;=\; \prod_{i=0}^{n-2} \left(1 - \frac{2^i}{2^n}\right).$$

Este producto es difícil de calcular. El objetivo ahora es encontrar un límite inferior no trivial. Si expandimos el producto obtenemos

$$\left(1 - \frac{1}{2^n}\right) \cdots \left(1 - \frac{2^{n-2}}{2^n}\right) = 1 - \sum_{i=0}^{n-2} \frac{2^i}{2^n} + \cdots .$$

La suma se calcula utilizando la serie geométrica que da como resultado $(1/2-1/2^n)$. La parte de $(\cdots)$ tiene términos de orden superior $(1/(2^n)^2, 1/(2^n)^3, ...)$, y la siguiente Proposición muestra que es positiva. Entonces,

$$p(n-1) \geq \frac{1}{2} + \frac{1}{2^n}.$$

Proposición 6.1. Sea $n \geq 2$ un número entero. Entonces

$$\prod_{i=0}^{n-2} \left(1 - \frac{2^i}{2^n}\right) \geq \frac{1}{2} + \frac{1}{2^n}.$$

Demostración. Por inducción sobre n. El caso base sigue después de reemplazar n con 2. Probemos el paso de inducción. La expresión de la izquierda para $n+1$ es

$$\prod_{i=0}^{n-1} \left(1 - \frac{2^i}{2^{n+1}}\right) = \left(1 - \frac{2^0}{2^{n+1}}\right) \prod_{i=1}^{n-1} \left(1 - \frac{2^i}{2^{n+1}}\right).$$

Al manipular el índice ficticio i, obtenemos

$$\prod_{i=0}^{n-1} \left(1 - \frac{2^i}{2^{n+1}}\right) = \left(1 - \frac{1}{2^{n+1}}\right) \prod_{i=0}^{n-2} \left(1 - \frac{2^i}{2^n}\right).$$

Supongamos que la desigualdad es verdadera para n. Después

$$\prod_{i=0}^{n-1} \left(1 - \frac{2^i}{2^{n+1}}\right) \geq \left(1 - \frac{1}{2^{n+1}}\right) \left(\frac{1}{2} + \frac{1}{2^n}\right).$$

Desarrollando el lado derecho, obtenemos

$$\prod_{i=0}^{n-1} \left(1 - \frac{2^i}{2^{n+1}}\right) \geq \frac{1}{2} + \frac{1}{2^{n+1}} + \frac{1}{2^{n+1}} \left(\frac{1}{2} - \frac{1}{2^n}\right) \geq \frac{1}{2} + \frac{1}{2^{n+1}}.$$

Esto completa la demostración. □

Aún no hemos terminado porque con $n-1$ ecuaciones independientes determinamos $n-1$ bits de s. El bit faltante s_i se determina adivinando, por ejemplo, suponiendo que $s_i = 0$ y luego usamos el oráculo clásico y preguntamos si $f(s) = f(0)$. Si es cierto, ya hemos encontrado s; de lo contrario, establecemos $s_i = 1$. El costo de ejecutar la parte clásica es básicamente el costo de resolver un sistema de $n-1$ ecuaciones lineales con n variables, que es el costo del orden de $O(n^2)$ de invertir una matriz $n \times n$.

El costo total del algoritmo es $n-1$ llamadas de U_f y una llamada de f más $O(n^2)$ pasos para resolver el sistema de ecuaciones lineales. La probabilidad de éxito es mayor que $1/2$.

6.5. Análisis del entrelazamiento

Del circuito del algoritmo, nos damos cuenta de que el único operador que crea o destruye entrelazamiento es U_f. Entonces, basta con analizar $|\psi_2\rangle$ o $|\psi_3\rangle$. Es más sencillo analizar $|\psi_3\rangle$. El

algoritmo de Simon tiene entrelazamiento sí y solo sí $|\psi_3\rangle$ está total o parcialmente entrelazado. El estado del primer registrador de $|\psi_3\rangle$ es

$$|\psi\rangle \;=\; \frac{|x\rangle + |x \oplus s\rangle}{\sqrt{2}},$$

donde x es una cadena aleatoria de n-bits y s es una cadena fija de n-bits distinta de cero. Si $s = 1..,1$ y $x = 1..,1$, $|\psi\rangle$ es el conocido estado Greenberger-Horne-Zeilinger de n qubits, definido como

$$|\text{GHZ}\rangle \;=\; \frac{|0\cdots 0\rangle + |1\cdots 1\rangle}{\sqrt{2}}.$$

Se sabe que el estado GHZ está entrelazado al máximo.[1] Si $s = 1..,1$, el estado $|\psi\rangle$ no es biseparable[2] para cualquier x porque

$$|\psi\rangle \;=\; X^{x_0} \otimes \cdots \otimes X^{x_{n-1}} |\text{GHZ}\rangle,$$

y $X^{x_0} \otimes \cdots \otimes X^{x_{n-1}}$ no crea ni destruye el entrelazamiento.

Por otro lado, podemos factorizar el estado $|\psi\rangle$ para cada bit 0 de s. Supongamos que $s = 01..,1$, entonces

$$|\psi\rangle \;=\; |x_0\rangle \otimes \frac{|x_1 ... x_{n-1}\rangle + |\bar{x}_1 ... \bar{x}_{n-1}\rangle}{\sqrt{2}},$$

donde $\bar{x}_i = x_i \oplus 1$. Este estado no está entrelazado al máximo, pero aún tiene entrelazamiento si $n > 2$. Si $s = 0..,01$, entonces

$$|\psi\rangle \;=\; |x_0\rangle \otimes \cdots |x_{n-2}\rangle \otimes \frac{|x_{n-1}\rangle + |\bar{x}_{n-1}\rangle}{\sqrt{2}},$$

no tiene ningún entrelazamiento en absoluto.

Resumiendo, si el peso de Hamming de s es mayor que 1, el algoritmo de Simon tiene entrelazamiento. La cantidad de entrelazamiento aumenta con el peso de Hamming de s y el estado del primer registrador antes de la medición se entrelaza al máximo, si el peso de Hamming de s es n.

6.6. Circuito del oráculo

En los algoritmos basados en oráculo, el oráculo no lo implementamos nosotros, lo implementa otra persona. Sin embargo, es importante saber como se hace para entender todo el proceso.

Pongamos un ejemplo con $n = 3$ y $s = 110$ que será suficiente para entender el caso general.

[1] https://en.wikipedia.org/wiki/Greenberger-Horne-Zeilinger_state

[2] Un estado puro $|\psi\rangle$ de n qubits se llama biseparable, si uno puede encontrar una partición de los qubits en dos registradores A y B tal que $|\psi\rangle = |\psi_A\rangle \otimes |\psi_B\rangle$.

Tomemos f como la siguiente función dos a uno

$x_0x_1x_2$	$f(x)$
0 0 0 1 1 0	000
0 0 1 1 1 1	001
0 1 0 1 0 0	010
0 1 1 1 0 1	100

Para construir el circuito, necesitamos que escribir la tabla de verdad explícita de 3 salidas, que es

$x_0x_1x_2$	$f_0f_1f_2$
0 0 0	0 0 0
0 0 1	0 0 1
0 1 0	0 1 0
0 1 1	1 0 0
1 0 0	0 1 0
1 0 1	1 0 0
1 1 0	0 0 0
1 1 1	0 0 1

Tenga en cuenta que $f(x) = f_0(x)f_1(x)f_2(x)$, donde f_0 a f_2 son funciones booleanas. La tabla de verdad de f_0 se obtiene considerando sólo la primera columna de la salida, y las tablas de verdad de f_1 y f_2 considerando la segunda y tercera columna, respectivamente. Ahora nos enfocamos en los bits 1 en la primera columna de la salida indicada por $f_0(x)$. Son dos, correspondientes a las entradas 011 y 101. Añadimos al circuito dos puertas Toffoli multiqubit, la primera con controles activados por 011 y la segunda activada por 101, con objetivo en el cuarto qubit, como se muestra en la Fig. 6.1. Luego, nos enfocamos en los bits 1 en la segunda columna de la salida indicada por $f_1(x)$. Hay dos de ellos, correspondientes a las entradas 010 y 100. Las puertas Toffoli multiqubit se activan con 010 y 100, respectivamente, con el objetivo en el quinto qubit, como se muestra en la Fig. 6.1. La última columna, indicada por $f_2(x)$, requiere puertas Toffoli multiqubit activadas por 001 y 111 con objetivo en el sexto qubit, como se muestra en la Fig. 6.1.

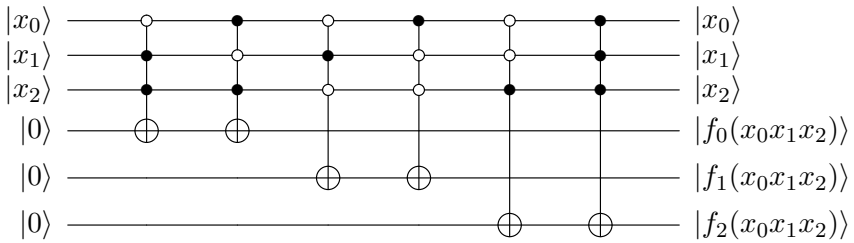


Figura 6.1: Algoritmo del oráculo de Simon.

La única simplificación trivial que se puede ver de inmediato es que las dos primeras puertas de Toffoli multiqubit se pueden simplificar en una sola puerta de Toffoli con control vacío en el qubit 2, control total en el qubit 3 y objetivo en el qubit 4.

6.7. Observaciones finales

La formulación del problema de Simon en el artículo original es ligeramente diferente de la que se presenta aquí. Simon planteó el problema de determinar si f es uno a uno (inyectivo) o un tipo especial de dos a uno caracterizado por una cadena de n-bits s tal que $f(x') = f(x)$, sí y solo sí $x' = x \otimes s$. En este último caso, tenemos que encontrar s. Esta formulación sigue la línea del algoritmo de Deutsch-Jozsa, en el que tenemos la promesa de que el oráculo es equilibrado o constante. Tenemos que determinar cuál es el caso. Tenga en cuenta que si ejecutamos la parte cuántica del algoritmo de Simon con una función uno a uno, la salida es una cadena de n-bits aleatoria. Simon usó este hecho para probar que existe un algoritmo para una máquina cuántica de Turing que resuelve el problema de Simon con probabilidad de error cero en el tiempo esperado $O(nT_f(n) + G(n))$, donde $T_f(n)$ es el tiempo requerido para calcular $f(x)$, y $G(n)$ es el tiempo requerido para resolver un sistema lineal $n \times n$ de ecuaciones sobre $\mathbb{Z}_2$ [58].

Capítulo 7

Algoritmo de Shor para factorizar enteros

Los algoritmos de Shor se presentaron en una conferencia en el año 1994 [54]. El artículo completo se publicó en el año 1997 [55] y se revisó en el año 1999 [56]. Describe dos algoritmos cuánticos para factorización de enteros y logaritmos discretos que se ejecutan en tiempo polinomial. Los algoritmos clásicos más conocidos se ejecutan en tiempo subexponencial. Los algoritmos de Shor explotan no sólo el paralelismo cuántico, sino también el entrelazamiento; siendo una contribución científica notable y célebre a la computación cuántica. El algoritmo para factorizar enteros es el tema central de este capítulo y se describe en muchos libros [27, 30, 37, 41, 51, 53, 60, 64].

7.1. Formulación del problema

Sea N un número natural compuesto. El problema computacional consiste en encontrar un factor no trivial de N. Si N es compuesto, existen los números naturales p_1 y p_2 tales que $N = p_1 p_2$, donde $1 < p_1, p_2 < N$. El objetivo es encontrar p_1 y luego p_2 que se obtiene calculando N/p_1.

Tenga en cuenta que si tenemos un algoritmo que encuentra un factor no trivial de N de manera eficiente, entonces podemos encontrar todos los factores primos de N de manera eficiente porque la cantidad de factores primos de N es como máximo $\log_2 N$.

7.2. Preliminares de la teoría de números

Aunque la factorización de números enteros es el principal interés porque tiene un gran impacto en la ruptura de métodos criptográficos como el RSA y el intercambio de claves Diffie-Hellman, podemos centrar nuestra atención en el problema de encontrar el orden multiplicativo de un número natural a módulo N, porque si se resuelve el último problema de manera eficiente, entonces también resolvemos el problema de factorización de manera eficiente.

En teoría de números, el problema de encontrar el orden multiplicativo de un número natural a módulo N tiene como objetivo determinar el entero positivo más pequeño r tal que

$$a^r \equiv 1 \mod N.$$

Por ejemplo, sea $N = 21$, que es el número compuesto más grande factorizado en una computadora cuántica hasta ahora usando el algoritmo de Shor [59]. Ahora elige al azar un número a tal

que $1 < a < N$. Digamos $a = 2$. Luego continúa multiplicando cada línea por a y simplificando usando aritmética modular hasta obtener 1:

$$\begin{array}{llll} a \equiv 2 & \text{mód } 21 & a^7 \equiv 2 & \text{mód } 21 \\ a^2 \equiv 4 & \text{mód } 21 & a^8 \equiv 4 & \text{mód } 21 \\ a^3 \equiv 8 & \text{mód } 21 & a^9 \equiv 8 & \text{mód } 21 \\ a^4 \equiv 16 & \text{mód } 21 & a^{10} \equiv 16 & \text{mód } 21 \\ a^5 \equiv 11 & \text{mód } 21 & a^{11} \equiv 11 & \text{mód } 21 \\ a^6 \equiv 1 & \text{mód } 21 & a^{12} \equiv 1 & \text{mód } 21 \end{array} \quad \ldots$$

El orden multiplicativo de 2 módulo 21 es 6 porque 6 es el entero positivo más pequeño tal que $2^6 \equiv 1 \mod 21$. Si continuamos con esta secuencia, los resultados 2, 4 y así sucesivamente se repetirán una y otra vez porque $a^{r+1} \equiv a^1 \equiv 2$, $a^{r+2} \equiv a^2 \equiv 4$, etc. Entonces, tenemos una secuencia periódica r.

Si r es par, entonces

$$\begin{aligned} \left(a^{\frac{r}{2}} + 1\right)\left(a^{\frac{r}{2}} - 1\right) &\equiv a^r - 1 \quad \text{mód } N \\ &\equiv 0 \quad \text{mód } N. \end{aligned}$$

Encontramos dos números $a^{r/2} + 1$ y $a^{r/2} - 1$ cuyo producto es múltiplo de N. Si esos números no son cero ni múltiplos de N, entonces $a^{r/2} + 1$ y $a^{r/2} - 1$ deben tener factores cuyo producto sea N. Concluimos que en este caso el $\text{mcd}\left(a^{r/2} + 1, N\right) > 1$ y el $\text{mcd}\left(a^{r/2} - 1, N\right) > 1$, donde mcd es el *máximo común divisor*.

Por ejemplo, cuando $a = 2$ y $r = 6$, tenemos $a^{r/2} + 1 = 9$ y $a^{r/2} - 1 = 7$, y en ambos casos el mcd devuelve un factor no trivial de 21. El método falla cuando $a = 5$ porque el orden multiplicativo de 5 módulo 21 es $r = 6$ y $a^{r/2} + 1 = 126$ y $a^{r/2} - 1 = 124$. En el primer caso, 126 es múltiplo de 21, y en el segundo, el $\text{mcd}(124, 21) = 1$.

En el algoritmo de Shor, comenzamos con el número N, luego elegimos al azar un número a tal que $1 < a < N$. Antes de calcular el orden de a, verificamos si el $\text{mcd}(a, N) > 1$ porque (1) si el $\text{mcd}(a, N) > 1$ entonces no hay r tal que $a^r \equiv 1 \mod N$ y (2) si el $\text{mcd}(a, N) > 1$ entonces a es un factor no trivial de N, y luego terminamos. Para comprender mejor lo que sucede aquí, dividimos el conjunto de números $\{1, ..., N\}$ en dos subconjuntos:

$$\begin{aligned} S_1 &= \{a : 1 < a < N \text{ y } \text{mcd}(a, N) > 1\}, \\ S_2 &= \{a : 1 \leq a < N \text{ y } \text{mcd}(a, N) = 1\}. \end{aligned}$$

En nuestro ejemplo con $N = 21$, tenemos

$$\begin{aligned} S_1 &= \{3, 6, 7, 9, 12, 14, 15, 18\}, \\ S_2 &= \{1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20\}. \end{aligned}$$

Cuando elegimos un número a al azar, si $a \in S_1$, encontramos rápidamente un factor de N calculando el $\text{mcd}(a, N)$ usando el algoritmo euclidiano.[1] La pregunta ahora es cual es la cardinalidad de S_1? ¿Es mayor que la cardinalidad de S_2? Para responder a esta pregunta, usamos los siguientes datos sobre S_2 para un N arbitrario, que se denota por $\mathbb{Z}_N^*$ en teoría de números [25, 43]:

[1] https://en.wikipedia.org/wiki/Euclidean_algorithm

Hecho 1 $\mathbb{Z}_N^*$ es un grupo multiplicativo finito módulo N.

Hecho 2 La cardinalidad de $\mathbb{Z}_N^*$ es la función indicatriz de Euler $\varphi(N)$.

El Hecho 1 es la base teórica que garantiza que existe r tal que $a^r \equiv 1 \mod N$ para cualquier $a \in \mathbb{Z}_N^*$. También garantiza que la función

$$f(\ell) = a^\ell \mod N$$

es r-periódica. Por otro lado, el Hecho 2 es la definición de la función indicatriz de Euler $\varphi(N)$, que ha sido ampliamente estudiada en teoría de números. Se sabe que $\varphi(N)$ siempre es casi N (Hardy and Wright [25]). Dado que la cardinalidad de S_1 es $N - \varphi(N) - 1$, la probabilidad de seleccionar $a \in S_1$ siempre es mucho menor que la probabilidad de seleccionar $a \in \mathbb{Z}_N^*$. Por ejemplo, suponga que $N = p_1 p_2$, donde p_1 y p_2 son números primos tales que el número de cifras de p_1 es cercano al número de cifras de la parte entera de $\sqrt{N}$. Lo mismo es cierto para p_2. Este es el caso más interesante en criptografía porque es el caso más difícil para los algoritmos de factorización clásicos. Esos números primos suelen tener 1024 bits, que está cerca de 308 cifras decimales. Para este caso, tenemos

$$S_1 \ = \ \{p_1, 2p_1, ..., (p_2 - 1)p_1, p_2, 2p_2, ..., (p_1 - 1)p_2\}$$

y la cardinalidad de S_1 es $p_1 + p_2 - 2$, que está cerca de $2\sqrt{N}$. Para N grande, vemos que hay más probabilidades de elegir a de $\mathbb{Z}_N^*$.

No todos los a en $\mathbb{Z}_N^*$ son buenos porque algunos de ellos tienen un orden multiplicativo impar. Si seleccionamos uno malo, tenemos que descartar a y elegir otro al azar. La pregunta ahora es ¿cuántos a en $\mathbb{Z}_N^*$ tienen un orden parejo? No solo eso, además de exigir un orden parejo también tenemos que exigir que el $\mathrm{mcd}\big(a^{r/2} + 1, N\big) > 1$ o el $\mathrm{mcd}\big(a^{r/2} - 1, N\big) > 1$.

Hecho 3 (Teorema A4.13 de [42]) Supongamos que $N = p_1^{\alpha_1} \cdots p_m^{\alpha_m}$ es la descomposición en factores primos de un entero positivo compuesto impar. Sea a uniformemente elegido al azar de $\mathbb{Z}_N^*$, y sea r del orden de a módulo N. Entonces a prob(r es par y $a^{r/2} + 1 \not\equiv 0 \mod N) \geq 1 - 1/2^m \geq 3/4$.

El teorema anterior establece que la probabilidad de seleccionar un buen a es al menos $3/4$. Tenga en cuenta que el caso $a^{r/2} - 1 \equiv 0 \mod N$, que es problemático, nunca sucede porque, por definición, r es el entero más pequeño tal que $a^r - 1 \equiv 0 \mod N$. Entonces, si r es par y $a^{r/2} + 1 \not\equiv 0 \mod N$, tenemos $\mathrm{mcd}\big(a^{r/2} + 1, N\big) > 1$ y $\mathrm{mcd}\big(a^{r/2} - 1, N\big) > 1$.

Tenga en cuenta que un divisor no trivial de N se obtiene fácilmente tan pronto como encontramos una solución entera no trivial para la ecuación

$$x^2 \equiv 1 \mod N,$$

tal que $x \not\equiv \pm 1 \mod N$. El objetivo del algoritmo de Shor es encontrar x ejecutando una parte cuántica que genera un r par, tal que $x^{r/2} \not\equiv \pm 1 \mod N$.

7.3. Operador cuántico para exponenciación modular

El operador unitario $U_N^{(a)}$, que calcula la exponenciación del entero a módulo N, se define como

$$U_N^{(a)}|\ell\rangle|y\rangle = |\ell\rangle\Big|y \oplus \big(a^\ell \mod N\big)\Big\rangle, \text{ para } 0 \le \ell < q,\ 0 \le y < 2^n,$$

donde $n = \lceil \log_2 N \rceil$, $\oplus$ es la operación XOR bit a bit o suma bit a bit módulo 2, y q es la potencia más pequeña de 2 tal que $q > N^2$. $U_N^{(a)}$ actúa sobre dos registradores de tamaño $\log_2 q$ y n, y es una matriz de permutación de dimensión $2^n q$. Las entradas del algoritmo, a y N, llegan a través de $U_N^{(a)}$. Reemplaza el oráculo U_f en el algoritmo de Simon, pero $U_N^{(a)}$ no es un oráculo porque es nuestra tarea implementarlo. En el algoritmo de Shor, $U_N^{(a)}$ se usa muchas veces con un a diferente cada vez. La prueba de que $U_N^{(a)}$ es unitaria es una extensión de la prueba presentada en la Proposición 3.1 de la Página 29.

Para implementar $U_N^{(a)}$ de manera eficiente, es necesario utilizar el *método de exponenciación por cuadrados repetidos*, que es un algoritmo eficiente para calcular la exponenciación modular. Por ejemplo, si queremos calcular $3^{16} \mod 7$, ingenuamente multiplicaríamos $3 \times 3 \times ... \times 3$ dieciséis veces, obtendríamos un número grande y luego calcularíamos el resto después de dividir por 7. En la exponenciación por cuadrados repetidos, calculamos el cuadrado de 3 módulo 7, luego calculamos el cuadrado del resultado módulo 7, y así sucesivamente cuatro veces, lo que requiere un número logarítmico de multiplicaciones y cada resultado nunca es demasiado grande. Cuando calculamos $a^\ell \mod N$, ℓ no es una potencia de 2 en general, pero aún se puede usar el método de elevar al cuadrado repetidamente. Usando este método, es posible encontrar un circuito eficiente de $U_N^{(a)}$ al convertir circuitos irreversibles clásicos en reversibles [41, 45].

7.4. Transformada de Fourier y su inversa

La transformada de Fourier q-dimensional F_q se define como

$$F_q|k\rangle = \frac{1}{\sqrt{q}} \sum_{\ell=0}^{q-1} \omega^{k\ell}|\ell\rangle,$$

donde $0 \le k < q$ y $\omega = \mathrm{e}^{2\pi \mathrm{i}/q}$. Tenga en cuenta que la entrada (k, ℓ) de F_q es

$$\big(F_q\big)_{k\ell} = \frac{\omega^{k\ell}}{\sqrt{q}}.$$

Entonces, F_q es una matriz simétrica. Para encontrar $F_q^\dagger$, simplemente tomamos el complejo conjugado de cada entrada, que es $\omega^{-k\ell}/\sqrt{q}$ porque el complejo conjugado de ω es ω^{-1}. Después

$$F_q^\dagger|k\rangle = \frac{1}{\sqrt{q}} \sum_{\ell=0}^{q-1} \omega^{-k\ell}|\ell\rangle,$$

donde $0 \le k < q$.

Demostremos que F_q es unitario. Usando las definiciones de F_q y $F_q^\dagger$, tenemos

$$\begin{aligned}\langle k'|F_q^\dagger F_q|k\rangle &= \left(\frac{1}{\sqrt{q}}\sum_{\ell'=0}^{q-1}\omega^{-k'\ell'}\langle\ell'|\right)\left(\frac{1}{\sqrt{q}}\sum_{\ell=0}^{q-1}\omega^{k\ell}|\ell\rangle\right)\\ &= \frac{1}{q}\sum_{\ell=0}^{q-1}\omega^{(k-k')\ell}.\end{aligned}$$

Ahora usamos la fórmula de forma cerrada para la serie geométrica con términos q, que es

$$\sum_{k=0}^{q-1}s^k \;=\; \frac{1-s^q}{1-s},$$

si $s \neq 1$. Cuando $s = 1$, la suma de la izquierda es igual a q. En nuestro caso $s = \omega^{(k-k')}$. De la definición de ω, tenemos $\omega^q = 1$ y luego $\omega^{(k-k')q} = 1$. Combinando esos resultados, obtenemos

$$\frac{1}{q}\sum_{\ell=0}^{q-1}\omega^{(k-k')\ell} \;=\; \begin{cases}1, & \text{si } k = k',\\ 0, & \text{caso contrario.}\end{cases}$$

Acabamos de demostrar que

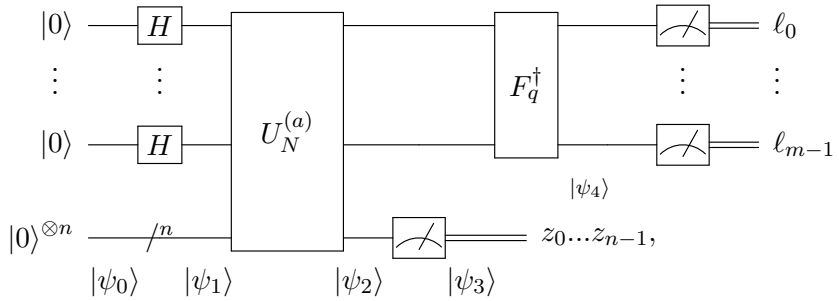
$$\langle k'|F_q^\dagger F_q|k\rangle = \delta_{kk'},$$

es decir, $F_q^\dagger F_q = I$.

En el algoritmo de Shor, dado N, q es la potencia más pequeña de 2 tal que $q > N^2$. F_q se aplica sólo al primer registrador. El número de qubits del primer registrador es $\log_2 q$, que está cerca de $2n$, mientras que el número de qubits del segundo registrador es exactamente $n = \lceil \log_2 N \rceil$. F_q juega un papel similar a las puertas de Hadamard al final del algoritmo de Simon. El circuito de F_q en términos de CNOTs y puertas de un qubit se describe en la Sec. 7.8.

7.5. El algoritmo

El algoritmo de Shor se describe en el Algoritmo 7.1 y la parte cuántica se describe en el Algoritmo 7.2. El circuito de la parte cuántica es



donde $N^2 < q < 2N^2$, $m = \log_2 q$, $n = \lceil \log_2 N \rceil$ y a es un número entero tal que el $\mathrm{mcd}(a, N) = 1$. El primer registrador tiene al menos $2n - 1$ qubits (a lo sumo $2n$) y el segundo tiene exactamente n qubits. Los estados en la parte inferior del circuito se utilizan en el análisis del algoritmo.

Algoritmo 7.1: Algoritmo de Shor

Entrada: Entero compuesto N.
Salida: Un factor no trivial de N.

1 Si N es par, devuelve 2; de lo contrario, continuar;
2 Si N es una potencia de algún número primo p, devuelve p; de lo contrario, continuar;
3 Elija uniformemente al azar un número entero a tal que $1 < a < N$;
4 Si el $\mathrm{mcd}(a, N) > 1$, devuelve el $\mathrm{mcd}(a, N)$; de lo contrario, continuar;
5 Ejecute la parte cuántica con las entradas a y N (Algoritmo 7.2, asuma la salida $\ell_0, ..., \ell_{m-1}$);
6 Si $\ell = 0$, vaya al Paso 5;
7 Calcula $b = \ell/q$ (el mismo q usado en la parte cuántica);
8 Encuentre el convergente de la expansión en fracción continua de b con el mayor denominador r' tal que $r' < N$;
9 Si r' es par, calcula $p_1 = \mathrm{mcd}(a^{r'/2} + 1, N)$; de lo contrario, vaya al Paso 3;
10 Si $1 < p_1 < N$, devuelve p_1; de lo contrario, vaya al Paso 3.

Algoritmo 7.2: Parte cuántica del algoritmo de Shor

Entrada: Un entero compuesto N y un entero $1 < a < N$ tal que el $\mathrm{mcd}(a, N) = 1$.
Salida: cadena de $(\log_2 q)$-bits de m bits, que es el entero más cercano a un múltiplo de q/r con probabilidad mayor o igual a $4/\pi^2$, donde q es la potencia de 2 más pequeña tal que $q > N^2$ y $m = \log_2 q$.

1 Prepare el estado inicial $|0\rangle^{\otimes m}|0\rangle^{\otimes n}$, donde $m = \lceil 2\log_2 N \rceil$ y $n = \lceil \log_2 N \rceil$;
2 Aplicar $H^{\otimes \log_2 q}$ al primer registrador;
3 Aplicar $U_N^{(a)}$ a ambos registradores;
4 Mida el segundo registrador en la base computacional (suponga la salida $z_0...z_{n-1}$);
5 Aplicar $F_q^\dagger$ al primer registrador;
6 Medir el primer registrador en la base computacional.

Describen los estados de los qubits después de cada paso. Tenga en cuenta que el estado $|\psi_4\rangle$ se refiere sólo al primer registrador. La notación "$/^n$" sobre un cable indica que es un registrador de n-qubits.

Hemos presentado el algoritmo de Shor como el algoritmo de Las Vegas, lo que significa que la salida siempre es correcta y el tiempo de ejecución esperado es finito. Con una pequeña modificación, puede presentarse como un algoritmo de Monte Carlo, lo que significa que la salida puede ser incorrecta.

7.6. Análisis de la parte cuántica

Cálculo de $|\psi_0\rangle$

Después del primer paso, el estado de la computadora cuántica es

$$|\psi_0\rangle = |0\rangle^{\otimes \log_2 q}|0\rangle^{\otimes n},$$

donde q es la potencia más pequeña de 2 tal que $q > N^2$.

Cálculo de $|\psi_1\rangle$

Después del segundo paso, el estado de la computadora cuántica es

$$\begin{aligned} |\psi_1\rangle &= \big(H|0\rangle\big)^{\otimes \log_2 q} \otimes |0\rangle^{\otimes n} \\ &= \frac{1}{\sqrt{q}} \sum_{\ell=0}^{q-1} |\ell\rangle \otimes |0\rangle^{\otimes n}, \end{aligned}$$

donde ℓ se escribe en notación decimal.

Cálculo de $|\psi_2\rangle$

Después del tercer paso, el estado de la computadora cuántica es

$$\begin{aligned} |\psi_2\rangle &= U_N^{(a)} |\psi_1\rangle \\ &= \frac{1}{\sqrt{q}} \sum_{\ell=0}^{q-1} U_N^{(a)} (|\ell\rangle \otimes |0 \cdots 0\rangle). \end{aligned}$$

Para simplificar $|\psi_2\rangle$, usamos la definición de $U_N^{(a)}$ para obtener

$$|\psi_2\rangle = \frac{1}{\sqrt{q}} \sum_{\ell=0}^{q-1} |\ell\rangle \otimes \Big| a^\ell \mod N \Big\rangle,$$

porque $(0 \ldots 0) \oplus a^\ell$ (XOR bit a bit) es a^ℓ. De ahora en adelante, descartamos la notación módulo N dentro del segundo ket porque no tenemos operación XOR y no hay peligro en no reconocer la aritmética correcta.

Es muy importante comprender la estructura de $|\psi_2\rangle$ antes de continuar. Desarrollando la suma, obtenemos

$$\begin{aligned} \sqrt{q}|\psi_2\rangle = \; & |0\rangle|1\rangle + |1\rangle|a\rangle + |2\rangle\big|a^2\big\rangle + \cdots + |r-1\rangle\big|a^{r-1}\big\rangle + \\ & |r\rangle|1\rangle + |r+1\rangle|a\rangle + |r+2\rangle\big|a^2\big\rangle + \cdots + |2r-1\rangle\big|a^{r-1}\big\rangle + \\ & |2r\rangle|1\rangle + |2r+1\rangle|a\rangle + |2r+2\rangle\big|a^2\big\rangle + \cdots + |3r-1\rangle\big|a^{r-1}\big\rangle + \\ & \vdots \qquad\qquad \vdots \qquad\qquad \vdots \qquad\qquad\qquad \vdots \\ & |(c-1)r\rangle|1\rangle + |(c-1)r+1\rangle|a\rangle + |(c-1)r+2\rangle\big|a^2\big\rangle + \cdots, \end{aligned}$$

donde $c = \lceil q/r \rceil$. De hecho, tenemos $q = (c-1)r + r_0$, donde r_0 es el resto de q dividido por r. La última línea tiene términos r_0 y está incompleta a menos que $r_0 = 0$.[2] Los valores posibles dentro del segundo ket son 1, a, a^2, ..., a^{r-1}. Hemos dividido $|\psi_2\rangle$ en columnas que tienen el mismo segundo ket. Las primeras columnas tienen términos c y las últimas columnas tienen términos $c-1$ términos.

[2] La última línea está completa si $r \mid q$. En este caso, r es una potencia de 2.

Cálculo de $|\psi_3\rangle$

El cuarto paso es una medición de cada qubit del segundo registrador, que suponemos que ha devuelto $z_0, ..., z_{n-1}$. Entonces existe r_1 tal que $a^{r_1} = z$, donde $0 \leq r_1 < r$. El estado $|\psi_2\rangle$ colapsa en una superposición del primer registrador con todos los términos $|\ell\rangle$ tal que $a^\ell \equiv a^{r_1}$ mód N, es decir, $\ell = kr + r_1$, para $0 \leq k < r$, dando

$$|\psi_3\rangle \;=\; \left(\frac{1}{\sqrt{c}}\sum_{k=0}^{c-1}|kr + r_1\rangle\right)|a^{r_1}\rangle,$$

donde $c = \lceil q/r \rceil$ si a^{r_1} pertenece a una de las primeras columnas ($r_1 \leq r_0$), y $c = \lfloor q/r \rfloor$ si a^{r_1} pertenece a una de las últimas columnas ($r_1 > r_0$). El análisis del algoritmo utiliza ampliamente el parámetro c. Es importante memorizar su definición y ser consciente de que es un número entero.

Tenga en cuenta que hemos vuelto a normalizar el estado $|\psi_3\rangle$ como exige el postulado de medición. No sabemos r_1 porque se selecciona al azar de 0 a $r-1$. La información que deseamos adquirir que es r, está oculta porque $kr + r_1$ es un valor aleatorio de 0 a $q-1$. Realizar una medición del primer registrador en este punto es inútil. Pero, la distribución de probabilidad del primer registrador es una función periódica r. Esto motiva la aplicación de la transformada inversa de Fourier porque el resultado es otra función (casi) periódica, cuyo periodo es cercano a q/r.

Cálculo de $|\psi_4\rangle$

En el quinto paso, solo consideramos el primer registrador. Después de aplicar $F_q^\dagger$ al estado del primer registrador,[3] obtenemos

$$\begin{aligned}|\psi_4\rangle \;&=\; \frac{1}{\sqrt{c}}\sum_{k=0}^{c-1}F_q^\dagger|kr + r_1\rangle \\ &=\; \frac{1}{\sqrt{c\,q}}\sum_{k=0}^{c-1}\sum_{\ell=0}^{q-1}\omega^{-\ell(kr+r_1)}|\ell\rangle.\end{aligned}$$

Reordenando el orden de las sumas, obtenemos

$$|\psi_4\rangle \;=\; \frac{1}{\sqrt{q}}\sum_{\ell=0}^{q-1}\omega^{-\ell r_1}\left(\frac{1}{\sqrt{c}}\sum_{k=0}^{c-1}\omega^{-\ell k r}\right)|\ell\rangle.$$

Para calcular la suma entre paréntesis, usamos nuevamente la fórmula de forma cerrada para la serie geométrica, que es

$$\sum_{k=0}^{c-1}s^k \;=\; \frac{1-s^c}{1-s},$$

si $s \neq 1$. Cuando $s = 1$, la suma de la izquierda es igual a c. Para la suma entre paréntesis, $s = \omega^{-\ell r} = \mathrm{e}^{-2\pi \mathrm{i}\ell r/q}$. Entonces

$$\sum_{k=0}^{c-1}\omega^{-\ell k r} \;=\; \begin{cases} c, & \text{si } q \mid (\ell r), \\ \frac{1-\omega^{-\ell c r}}{1-\omega^{-\ell r}}, & \text{caso contrario.}\end{cases}$$

[3] El algoritmo usa $F_q^\dagger$ y no F_q en este punto porque el objetivo es convertir un estado de superposición en un estado de la base computacional, no al revés.

El estado $|\psi_4\rangle$ se simplifica a

$$|\psi_4\rangle \;=\; \frac{\sqrt{c}}{\sqrt{q}} \sum_{\substack{\ell=0 \\ q|(\ell r)}}^{q-1} \omega^{-\ell r_1}|\ell\rangle + \frac{1}{\sqrt{qc}} \sum_{\substack{\ell=0 \\ q\nmid(\ell r)}}^{q-1} \omega^{-\ell r_1} \frac{1-\omega^{-\ell cr}}{1-\omega^{-\ell r}}|\ell\rangle, \tag{7.1}$$

donde la notación $q \nmid (\ell r)$ significa que q no divide a ℓr. La primera suma es sobre ℓ tal que q es un divisor (ℓr). La segunda suma es sobre los ℓ restantes.

Cálculo de la salida

La probabilidad de obtener $0 \leq \ell < q$ es

$$p(\ell) \;=\; \begin{cases} \frac{c}{q}, & \text{si } q \mid (\ell r), \\ \frac{1}{q\,c}\left|\frac{1-\omega^{-\ell cr}}{1-\omega^{-\ell r}}\right|^2, & \text{caso contrario.} \end{cases}$$

Usando la definición de ω y $\left|1-\mathrm{e}^{2\pi \mathrm{i}\theta}\right|^2 = 4\sin^2(\pi\theta)$, $p(\ell)$ se simplifica a

$$p(\ell) \;=\; \begin{cases} \frac{c}{q}, & \text{si } q \mid (\ell r), \\ \frac{\sin^2 \frac{\pi \ell r c}{q}}{q\,c\,\sin^2 \frac{\pi \ell r}{q}}, & \text{caso contrario.} \end{cases} \tag{7.2}$$

Los términos que más contribuyen a $p(\ell)$ cuando $q \nmid (\ell r)$ son los ℓ que cumplen lo siguiente

$$\sin^2 \frac{\pi \ell r}{q} \approx 0,$$

porque $p(\ell)$ es grande cuando el denominador es cercano a cero. Esto implica que $\pi \ell r/q$ debe estar cerca de un múltiplo de π (digamos $k\pi$) y luego los ℓ que más contribuyen son

$$\ell \approx \frac{kq}{r},$$

donde k va de 0 a $r-1$. Tenga en cuenta que si ℓ es cero o un múltiplo exacto de q/r entonces $p(\ell) = c/q \approx 1/r$; vea la definición de $p(\ell)$ en la Ec. (7.2). Este análisis explica por qué los picos de la Fig. 7.1 corresponden a ℓ que están cerca de kq/r. La salida de la parte cuántica es una cadena de $(\log_2 q)$-bits ℓ tal que ℓ está cerca de un múltiplo de q/r.

Detalles sobre la distribución de probabilidades

En el análisis anterior, falta mostrar que el numerador de $p(\ell)$ cuando $q \nmid (\ell r)$, el término

$$\sin^2 \frac{\pi \ell r c}{q},$$

no es demasiado pequeño cuando $\ell \approx kq/r$ para $0 \leq k < r-1$. Dado que este análisis es demasiado largo cuando ℓ es una variable discreta, tomamos una ruta alternativa.

Consideremos $p(\ell)$ como una función continua en términos de ℓ en el dominio $[0, q]$. Mediante el uso de identidades trigonométricas y el hecho de que c es un número entero, es sencillo demostrar que

$$p\left(\ell + \frac{q}{r}\right) = p(\ell).$$

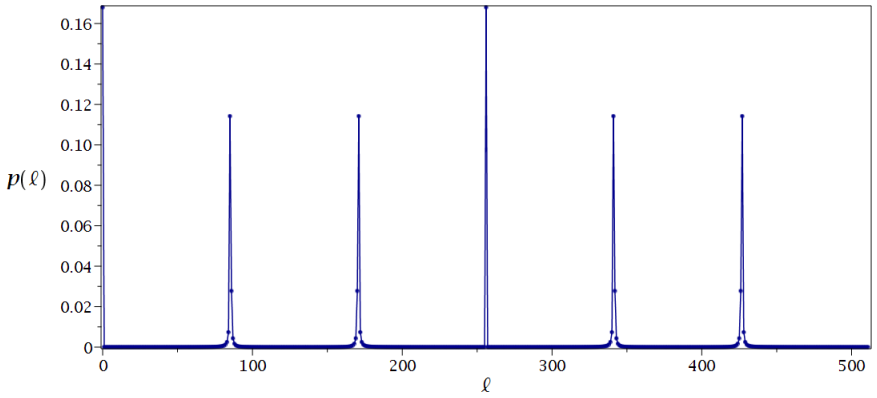


Figura 7.1: Distribución de probabilidades de $p(\ell)$ como una función de ℓ dada por la Ec. (7.2) cuando $N = 21$, $q = 2^9$, y $r = 6$. Los puntos en la parte superior corresponden a los picos $\ell = 0, 85, 171, 256, 341, 427$.

Cuando observamos $p(\ell)$ como una función continua, podemos demostrar que es una función verdaderamente periódica, mientras que la función representada en la Fig. 7.1 no lo es.

Ahora obtengamos la forma de $p(\ell)$. Dado que $p(\ell)$ es (q/r)-periódico, consideremos el intervalo $\ell \in [0, q/r]$, y además limitémonos a ℓ tal que $q \nmid (\ell r)$. Usamos la expresión

$$p(\ell) = \frac{\sin^2 \frac{\pi \ell r c}{q}}{q\, c \sin^2 \frac{\pi \ell r}{q}}. \tag{7.3}$$

El numerador de $p(\ell)$, $\sin^2(\pi \ell r c/q)$, es el cuadrado de una función sinusoidal, que es (q/rc)-periódica. Tenga en cuenta que q/rc es exactamente 1 si $r \mid q$ y está cerca de 1 si $r \nmid q$ porque c es $\lceil q/r \rceil$ o $\lfloor q/r \rfloor$. Un ejemplo del numerador de $p(\ell)$ se muestra en la Fig. 7.2(a) para $\ell \in [0, q/r]$. El denominador de $p(\ell)$ (sin qc), $\sin^2(\pi \ell r/q)$, también es el cuadrado de una función sinusoidal,

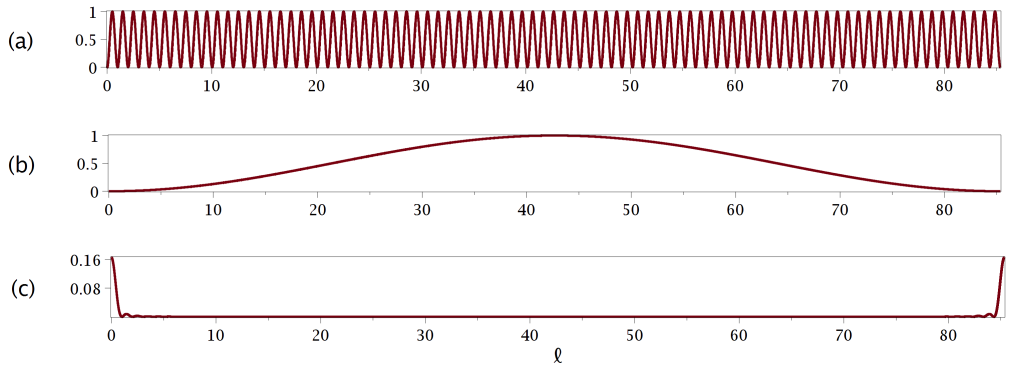


Figura 7.2: (a) Numerador de $p(\ell)$ como una función continua de ℓ para $\ell \in [0, q/r]$ cuando $N = 21$, $q = 2^9$, y $r = 6$. (b) Denominador de $p(\ell)$ como una función continua de ℓ sin el término qc. (c) $p(\ell)$ como una función continua de ℓ para $\ell \in [0, q/r]$. La figura (c) se obtiene dividiendo (a) por (b) por qc.

que es cero solo en $\ell = 0$ y $\ell = q/r$. Un ejemplo del denominador de $p(\ell)$ (sin el término qc) para los mismos valores de N, q y r se muestra en la Fig. 7.2(b) para $\ell \in [0, q/r]$.

Si dividimos el gráfico que se muestra en la Fig. 7.2(a) por el gráfico que se muestra en la Fig. 7.2(b) y dividimos el resultado por qc, obtenemos el gráfico que se muestra en la Fig. 7.2(c), que es la versión continua de la gráfica que se muestra en la Fig. 7.1 para $0 \le \ell < q/r$. Es sencillo verificar dos hechos: (1) los valores más grandes de $p(\ell)$ están cerca de $\ell = 0$ y $\ell = q/r$ porque el denominador de $p(\ell)$ está cerca de cero; y (2) $p(\ell)$ está cerca de cero en el medio (ℓ cerca de $q/2r$) porque el numerador está cerca de 1 y el denominador está cerca de qc, que es grande. Muchos otros datos sobre $p(\ell)$ se obtienen de esta división, por ejemplo, es fácil encontrar el número de ceros de $p(\ell)$. Usando nuestro conocimiento de cálculo, comprobamos que

$$\lim_{\ell=0^+} \frac{\sin^2 \frac{\pi \ell r c}{q}}{\sin^2 \frac{\pi \ell r}{q}} = \lim_{\ell=\frac{q}{r}^-} \frac{\sin^2 \frac{\pi \ell r c}{q}}{\sin^2 \frac{\pi \ell r}{q}} = c^2.$$

Luego, $p(0) = p(q/r) = c/q \approx 1/r$. Con este análisis, hemos obtenido la forma de $p(\ell)$ en todo el dominio porque $p(\ell)$ es (q/r)-periódico, es decir, si ponemos 6 gráficos de la Fig. 7.2(c) uno al lado del otro, obtenemos la versión continua de la gráfica de la Fig. 7.1.

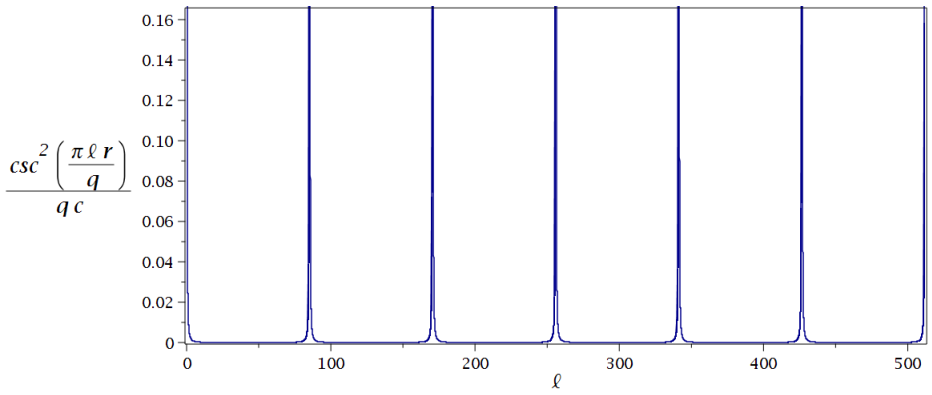


Figura 7.3: La parte superior de la envolvente de $p(\ell)$ en función de ℓ en el dominio $[0, q]$ cuando $N = 21$ ($q = 2^9$, $r = 6$, $c = 85$), que se obtiene usando el cuadrado de la función cosecante ($\csc x = 1/\sin x$).

Existe una ruta alternativa para obtener la forma de $p(\ell)$ en el dominio $[0, q]$. El numerador de $p(\ell)$, $\sin^2 \frac{\pi \ell r c}{q}$, es el cuadrado de una función sinusoidal que oscila rápidamente dentro de una envolvente, cuya parte inferior es el ℓ-eje y la parte superior es la función

$$\frac{\csc^2 \left(\frac{\pi \ell r}{q}\right)}{qc},$$

que se obtiene de Eq. (7.3) al reemplazar el numerador de $p(\ell)$ con 1. Fig. 7.3 representa la parte superior de la envolvente en el dominio $[0, q]$, que debe compararse con Fig. 7.1. Tenga en cuenta que la envolvente tiene asíntotas verticales para cada ℓ múltiplo de q/r.

Olvidémonos ahora de la envolvente y concentrémonos en la forma del primer pico de $p(\ell)$. La Fig. 7.4 muestra el primer pico en detalle cuando $N = 21$, $q = 2^9$ y $r = 6$ mostrando no solo la versión continua en rojo sino también los valores discretos en azul cuando ℓ es un número entero. Los puntos en azul son los mismos que se muestran en el primer pico de la Fig. 7.1, y al menos cuatro de ellos son claramente reconocibles. Tenga en cuenta que el primer pico de la

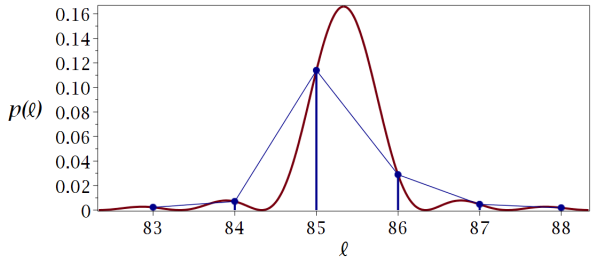


Figura 7.4: Primer pico de la distribución de probabilidades $p(\ell)$ como una función de ℓ cuando $N = 21$, $q = 2^9$, y $r = 6$. La curva roja es la versión continua de $p(\ell)$ y la altura del pico es $1/r$. Los valores de $p(\ell)$ para un entero de ℓ están resaltados en azul. La figura azul es la versión estirada del primer pico de la Fig. 7.1.

Fig. 7.1 no llega a c/q porque el pico se cancela antes de alcanzar la cima subyacente. La cumbre subyacente se alcanza sólo para ℓ tal que $q \mid (\ell r)$.

El caso particular $r \mid q$ (r es una potencia de 2) es destacable. En este caso, c es exactamente igual a q/r, que es un número entero, y $p(\ell)$ llega a la cima, cuya altura es exactamente $1/r$, para todos los ℓ que son múltiplos de q/r y $p(\ell)$ va al fondo del valle ($p(\ell) = 0$) para todos los demás valores enteros de ℓ. En la Ec. (7.1), $|\psi_4\rangle$ tiene sólo la primera suma porque todos los términos de la segunda suma desaparecen desde $\omega^{-\ell cr} = \omega^{-\ell q} = 1$. En este caso, la versión discreta de la distribución de probabilidades es verdaderamente periódica con período q/r. Sucede porque estamos usando qubits y luego la dimensión del espacio de Hilbert es una potencia de 2.

Probabilidad de éxito de la parte cuántica

La esencia del análisis es el cálculo de la probabilidad de éxito después de una ronda de la parte cuántica del algoritmo. Ya hemos acumulado suficiente información sobre la distribución de probabilidades; estamos listos para el cálculo. Hay una desigualdad trigonométrica.

$$4\alpha^2 \le \sin^2(\pi\alpha) \le \pi^2\alpha^2 \ \text{ for } \ |\alpha| \le \frac{1}{2},$$

que se puede probar usando cálculo básico. Usando ambos lados de esa desigualdad, demostramos la siguiente proposición:

Proposición 7.1. Si $|\alpha| \le 1/2c$ y $c \ge 1$, entonces

$$\frac{\sin^2 \pi\alpha c}{\sin^2 \pi\alpha} \ge \frac{4c^2}{\pi^2}.$$

Usando esta proposición, podemos encontrar un límite inferior interesante en la probabilidad de éxito. Comencemos por calcular un límite inferior en la probabilidad de $p(\ell)$ cuando ℓ es el entero más cercano a un múltiplo de q/r. Supongamos que $q \nmid (\ell r)$ y $\ell = \lfloor kq/r \rceil$ por $1 \le k < r$, donde $\lfloor\ \rceil$ es la notación del entero más cercano. Después,

$$p\left(\left\lfloor \frac{kq}{r} \right\rceil\right) = \frac{\sin^2\left(\pi \left\lfloor \frac{kq}{r} \right\rceil \frac{rc}{q}\right)}{q\, c \sin^2\left(\pi \left\lfloor \frac{kq}{r} \right\rceil \frac{r}{q}\right)}.$$

Ahora usamos la identidad trigonométrica $\sin^2\alpha = \sin^2(\pi k' - \alpha)$ válida para cualquier entero k' para obtener

$$p\left(\left\lfloor \frac{kq}{r} \right\rceil\right) = \frac{\sin^2 \pi\alpha c}{q\, c \sin^2 \pi\alpha},$$

donde

$$\alpha = k - \left\lfloor \frac{kq}{r} \right\rceil \frac{r}{q}.$$

Usando

$$\left| \frac{kq}{r} - \left\lfloor \frac{kq}{r} \right\rceil \right| \leq \frac{1}{2}, \tag{7.4}$$

obtenemos

$$|\alpha| \leq \frac{r}{2q} \leq \frac{1}{2\left\lfloor \frac{q}{r} \right\rfloor}.$$

Asumiendo que $c = \lfloor q/r \rfloor$ y usando la Proposición 7.1, obtenemos

$$p\left(\left\lfloor \frac{kq}{r} \right\rceil\right) \geq \frac{4c}{\pi^2 q} \approx \frac{4}{\pi^2 r}.$$

Podemos visualizar este resultado observando que hay dos barras azules dentro del pico de $p(\ell)$ en la Fig. 7.4. Acabamos de demostrar que la altura de la barra más grande nunca es inferior a $4/\pi^2 r$, alrededor del 40,5 % de la altura del pico.

En la parte cuántica del algoritmo, hay r posibles k. Entonces, un límite inferior en la probabilidad de éxito es

$$p_{\text{éxito}} \geq \frac{4}{\pi^2},$$

lo que significa que la parte cuántica tiene al menos un 40,5 % de posibilidades de devolver una cadena de $(\log_2 q)$-bits ℓ que es el entero más cercano a un múltiplo de q/r.

7.7. Análisis de la parte clásica

Antes de ejecutar la parte cuántica del algoritmo, debemos hacer una lista de verificación clásica. Es necesario verificar si N es un número compuesto, lo que se puede hacer de manera eficiente mediante pruebas de primalidad[4]. Comprobamos rápidamente si N es par. Además, existen algoritmos clásicos eficientes para comprobar si N es una potencia de algún número primo p [7]. Después de asegurarnos de que N es un número impar compuesto y no es una potencia de un número primo, seleccionamos un entero aleatorio a tal que $1 < a < N$ y verificamos si $\text{mcd}(a, N) = 1$, lo cual se puede hacer de manera eficiente usando el algoritmo euclidiano.

Luego, después de ejecutar la parte cuántica del algoritmo, asumimos que la salida es una cadena de $(\log_2 q)$-bits ℓ tal que ℓ es el entero más cercano a un múltiplo de q/r, es decir

$$\ell = \left\lfloor \frac{kq}{r} \right\rceil$$

para algunos k tal que $0 \leq k < r$. Definimos

$$b = \frac{\ell}{q},$$

[4] https://en.wikipedia.org/wiki/Primality_test

que obedece a $0 \leq b < 1$. Ahora usamos el método de aproximación de fracción continua para obtener la información deseada r. Una expansión en fracción continua de un número racional positivo $b < 1$ es

$$b = \cfrac{1}{b_0 + \cfrac{1}{b_1 + \cfrac{1}{\ddots + \cfrac{1}{b_m}}}},$$

donde b_0 a b_m son números enteros positivos y m es un número natural. La notación para la fracción continua es $[b_0, b_1, ..., b_m]$, y los convergentes sucesivos son $[b_0]$, $[b_0, b_1]$, $[b_0, b_1, b_2]$, y así sucesivamente, cada uno más cercano a b, hasta el último, que es igual a b. Cada convergente se convierte en un número racional al truncar la expansión de la fracción continua. En el algoritmo, tenemos que encontrar el $[b_0, b_1, ..., b_j]$ convergente que tiene el mayor j tal que el denominador del número racional equivalente sea menor que N. El denominador de este convergente es el candidato a r.

Por ejemplo, los convergentes sucesivos de ℓ/q cuando $\ell = 85$ y $q = 2^9$ son $[6] = 1/6$, $[6, 42] = 42/253$ y $[6, 42, 2] = 85/512 = \ell/q$. Cuando $N = 21$ y $a = 2$, tenemos que seleccionar el convergente $[6]$ y luego el candidato para r es el denominador de $1/6$. Afortunadamente, $a^6 \equiv 1$ mód N. Nuestra suerte fallaría si escogiéramos $\ell = 171$ (ver Fig. 7.1) porque concluiríamos que $r = 3$.

Ahora podemos ver por qué tenemos que exigir que $q > N^2$. Es una consecuencia de un teorema probado al final del capítulo sobre expansión en fracciones continuas en el libro de Hardy y Wright [25].

Teorema 7.2. (Hardy y Wright) Si k y r son enteros positivos b es un número real positivo y

$$\left| \frac{k}{r} - b \right| < \frac{1}{2r^2},$$

entonces k/r es un convergente de la expansión en fracción continua de b.

En la Ec. (7.4), hemos demostrado que la salida ℓ de la parte cuántica obedece lo siguiente

$$\left| \frac{k}{r} - \frac{\ell}{q} \right| \leq \frac{1}{2q}, \tag{7.5}$$

para algún entero k tal que $0 \leq k < r$, que nos es desconocido al igual que r. Para usar el Teorema 7.2, tenemos que exigir que $q > N^2$ porque N es un límite superior para r, es decir, $1/q < 1/N^2 < 1/r^2$. Entonces, podemos usar el teorema y estar seguros de que k/r será un convergente de ℓ/q.

Para entender por qué tenemos que elegir el $[b_0, b_1, ..., b_j]$ convergente que tiene el j más grande de modo que el denominador del número racional equivalente sea menor que N, debemos considerar los siguientes hechos. Si elegimos un $[b_0, b_1, ..., b_{j'}] = k'/r'$ convergente que obedece a la Ec. (7.5) y $r' < N$ y no tiene el j' más grande, entonces el siguiente convergente, digamos $[b_0, b_1, ..., b_{j+1}] = k''/r''$ también obedece a la Ec. (7.5) y $r'' < N$. Obtenemos una contradicción porque por un lado usando la Ec. (7.5) dos veces tenemos

$$\left| \frac{k'}{r'} - \frac{k''}{r''} \right| = \left| \left(\frac{k'}{r'} - b \right) - \left(\frac{k''}{r''} - b \right) \right| \leq \frac{1}{q} < \frac{1}{N^2},$$

y por otro lado tenemos

$$\left| \frac{k'}{r'} - \frac{k''}{r''} \right| = \frac{|k'r'' - k''r'|}{r'r''} > \frac{1}{N^2}.$$

La última desigualdad se deriva de las desigualdades $|k'r'' - k''r'| \geq 1$ y $r' < N$ y $r'' < N$. En conclusión, sólo existe un $[b_0, b_1, ..., b_j]$ convergente que obedece a la Ec. (7.5) tal que el denominador de la fracción equivalente es menor que N; es el que tiene el mayor j.

Aún no hemos terminado porque puede suceder que el $\text{mcd}(k, r) > 1$. En este caso, cuando miramos el denominador de k/r, obtenemos un factor de r, no r en sí mismo. Tenemos que descartar esos casos y preguntamos cuantos k relativamente primos con r hay. La proporción de buenos k es $\varphi(r)/r$, donde $\varphi(r)$ es la función indicatriz de Euler. Hay un límite inferior para $\varphi(r)$ dado por [25, 52]

$$\varphi(r) > \frac{r}{4\ln(\ln(r))}, \text{ para } r \geq 7.$$

Entonces, un límite inferior para la probabilidad de que la salida ℓ sea el entero más cercano a un múltiplo de q/r y el $\text{mcd}(k, r) = 1$ es

$$p\left(\ell = \left\lfloor \frac{kq}{r} \right\rceil \text{ y } \text{mcd}(k, r) = 1\right) > \frac{1}{\pi^2 \ln(\ln(r))},$$

para $r \geq 7$.

Si pensamos en el algoritmo de factorización como un algoritmo de Monte Carlo, que se obtiene eliminando las declaraciones "vaya al Paso"del Algoritmo 7.1, el algoritmo se ejecutará sólo una vez y un límite inferior en la probabilidad general de éxito es

$$\frac{3}{4\pi^2 \ln(\ln(r))},$$

que se obtiene del análisis de esta Sección y del Hecho 3 $\big(r$ debe ser par y el $\text{mcd}(a^{r/2}+1, N) > 1\big)$. Si pensamos en el algoritmo de factorización como un algoritmo de Las Vegas, tal como se describe en Algoritmo 7.1, la probabilidad de éxito es 1, pero tenemos que calcular un límite superior para el número promedio de veces que la parte cuántica del algoritmo se ejecutará hasta encontrar un factor de N. Suponga que un algoritmo tiene la probabilidad $0 < p < 1$ de generar el resultado correcto en una ejecución del algoritmo. La probabilidad de generar el resultado correcto después de ejecutar n veces el algoritmo exactamente es $(1-p)^{n-1}p$ porque debe fallar $n-1$ veces antes de tener éxito. El número promedio de veces que se ejecutará el algoritmo es[5]

$$\sum_{n=1}^{\infty} n\,(1-p)^{n-1}p \;=\; \frac{1}{p}.$$

Luego, un límite superior en el número promedio de veces que la parte cuántica se ejecutará en el Algoritmo 7.1 es $4\pi^2 \ln(\ln(r))/3$ para $r \geq 7$.

7.8. Circuito de la transformada de Fourier

La primera descomposición de la transformada de Fourier en términos de puertas básicas se encuentra en un informe de la IBM del año 1994, que estuvo ampliamente disponible en el

[5]La suma se calcula usando

$$\sum_{n=1}^{\infty} nq^n \;=\; q\,\frac{\mathrm{d}}{\mathrm{d}q}\left(\sum_{n=1}^{\infty} q^n\right),$$

donde $q = 1 - p$. La suma dentro de la derivada es la serie geométrica, cuyo valor es $q/(1-q)$ cuando $|q| < 1$.

año 2002 [15]. Una descripción de esta descomposición basada en la FFT clásica está disponible en [36].

El bloque básico del circuito de la transformada de Fourier F_{2^n}, donde n es el número de qubits, es la puerta controlada $C(R_k)$ para $k \geq 0$, donde

$$R_k = \begin{bmatrix} 1 & 0 \\ 0 & \exp\left(\frac{2\pi \mathrm{i}}{2^k}\right) \end{bmatrix}.$$

La representación matricial de $C(R_k)$ es

$$C(R_k) = \begin{bmatrix} I_2 & \\ & R_k \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \exp\left(\frac{2\pi \mathrm{i}}{2^k}\right) \end{bmatrix}.$$

El conjunto de puertas R_k tiene muchos subcasos especiales, que son

$$\begin{aligned} R_0 &= I_2, \\ R_1 &= Z, \\ R_2 &= S, \\ R_3 &= T, \end{aligned}$$

donde Z, S y T son las puertas Pauli Z, la puerta de fase y la puerta $\pi/8$ o T, respectivamente. Tenga en cuenta que, $R_{k+1} = \sqrt{R_k}$. Entonces, la secuencia anterior significa que la siguiente puerta es la raíz cuadrada de la anterior. La misma idea se aplica a $C(R_k)$, es decir, $C(R_{k+1}) = \sqrt{C(R_k)}$, pero en este último caso estamos calculando la raíz cuadrada de matrices (4×4). La Fig. 7.5 representa el circuito de F_{2^n} en términos de la puerta Hadamard, $C(R_k)$ y puertas

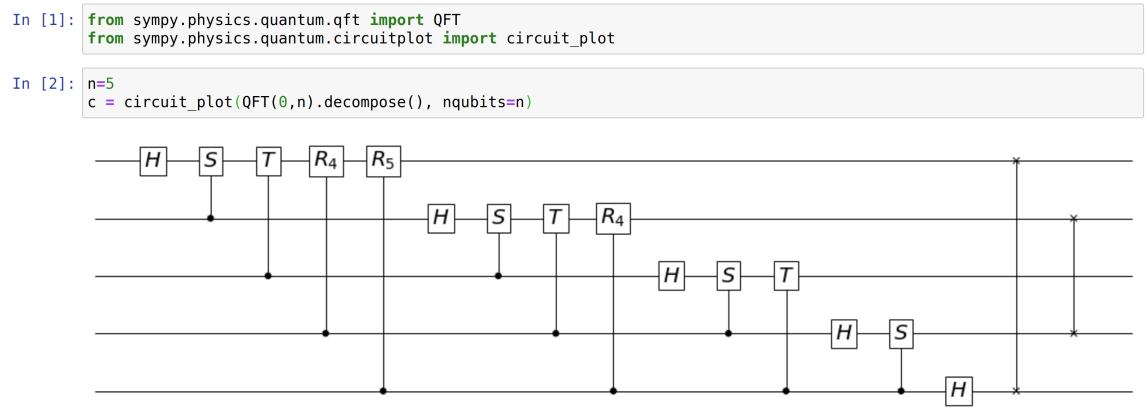


Figura 7.5: Print de un notebook de Jupyter que muestra la descomposición de la transformada de Fourier F_{2^5} usando comandos de Python *QFT* y *circuit_plot* de la librería sympy.

swaps cuando $n = 5$. La estructura del circuito se puede comprender fácilmente a partir de este ejemplo. El circuito tiene $n+1$ bloques. De izquierda a derecha, el primer bloque tiene n puertas, comenzando con H y luego con R_2, R_3,..., R_n actuando sobre el qubit 1 y controlado por el qubit 2, 3,..., n, respectivamente. El siguiente bloque comienza nuevamente con H y luego con R_2, R_3,..., R_{n-1} actuando sobre el qubit 2 y controlado por el qubit 3, 4,..., n, respectivamente. Esto

continúa hasta que llegamos al último qubit, en el que se aplica un solo bloque H (n-ésimo). El último bloque está hecho de $\lfloor n/2 \rfloor$ puertas swap y tiene una estructura simétrica simple. Si n es impar, el qubit central no se intercambia. El número de puertas es

$$n + (n-1) + \cdots + 1 + \left\lfloor \frac{n}{2} \right\rfloor \;=\; \frac{n(n+1)}{2} + \left\lfloor \frac{n}{2} \right\rfloor .$$

Demostremos que un circuito con la estructura representada en la Fig. 7.5 implementa la transformada de Fourier cuando tenemos n qubits. Suponga que la entrada es $|\ell\rangle = |\ell_1\rangle \otimes \cdots \otimes |\ell_n\rangle$. Cuando la entrada es un estado de la base computacional, la salida es un estado desentrelazado $|\psi_1\rangle \otimes \cdots \otimes |\psi_n\rangle$. Empecemos por calcular la salida $|\psi_1\rangle$ del primer qubit. Dado que hay una puerta de intercambio que invierte los estados del primer y el último qubit, tenemos

$$|\psi_1\rangle = H|\ell_n\rangle = \frac{|0\rangle + \mathrm{e}^{2\pi \mathrm{i} \frac{\ell_n}{2}}|1\rangle}{\sqrt{2}} = \frac{|0\rangle + \mathrm{e}^{2\pi \mathrm{i} \frac{\ell}{2}}|1\rangle}{\sqrt{2}}.$$

La segunda ecuación se deriva del hecho de que si ℓ_n es 0, la salida es $|+\rangle$, y si $\ell_n = 1$, la salida es $|-\rangle$ (porque $\mathrm{e}^{\pi \mathrm{i}}$ es -1). La última ecuación sigue de la descomposición de $\ell = 2^{n-1}\ell_1 + 2^{n-2}\ell_{n-2} + \cdots + 2\ell_{n-1} + \ell_n$ y de $\mathrm{e}^{2\pi \mathrm{i} k} = 1$ si k es un número entero.

La salida del segundo qubit es

$$|\psi_2\rangle = R_2^{\ell_n} H|\ell_{n-1}\rangle = R_2^{\ell_n} \left(\frac{|0\rangle + \mathrm{e}^{2\pi \mathrm{i} \frac{\ell_{n-1}}{2}}|1\rangle}{\sqrt{2}} \right) = \frac{|0\rangle + \mathrm{e}^{2\pi \mathrm{i} \left(\frac{\ell_{n-1}}{2} + \frac{\ell_n}{2^2} \right)}|1\rangle}{\sqrt{2}} = \frac{|0\rangle + \mathrm{e}^{2\pi \mathrm{i} \frac{\ell}{2^2}}|1\rangle}{\sqrt{2}}.$$

La primera ecuación se obtiene de la Fig. 7.5 usando ese $C(R_2)|\ell_n\rangle|\ell_{n-1}\rangle = |\ell_n\rangle\big(R_2^{\ell_n}|\ell_{n-1}\rangle\big)$. La segunda ecuación utiliza el mismo cálculo descrito anteriormente para el primer qubit. La tercera ecuación se sigue de

$$\begin{aligned} R_2^{\ell_n}|0\rangle &= |0\rangle, \\ R_2^{\ell_n}|1\rangle &= \mathrm{e}^{2\pi \mathrm{i} \frac{\ell_n}{2^2}}|1\rangle. \end{aligned}$$

La última ecuación utiliza la misma descomposición de ℓ descrita anteriormente.

La salida del último qubit es

$$|\psi_n\rangle = R_n^{\ell_n} \cdots R_2^{\ell_2} H|\ell_1\rangle = \frac{|0\rangle + \mathrm{e}^{2\pi \mathrm{i} \left(\frac{\ell_1}{2} + \frac{\ell_2}{2^2} + \cdots + \frac{\ell_n}{2^n} \right)}|1\rangle}{\sqrt{2}} = \frac{|0\rangle + \mathrm{e}^{2\pi \mathrm{i} \frac{\ell}{2^n}}|1\rangle}{\sqrt{2}}.$$

La primera ecuación se obtiene de la Fig. 7.5 usando que la salida del último qubit se obtiene de la acción de H, $R_2^{\ell_2}$, ..., $R_n^{\ell_n}$ en el primer qubit. La segunda y tercera ecuación se obtiene con el mismo tipo de cálculos descritos anteriormente para el primer y segundo qubit.

Entonces, la salida $|\psi_1\rangle \otimes \cdots \otimes |\psi_n\rangle$ del circuito de la Fig. 7.5 con n qubits es

$$\frac{|0\rangle + \mathrm{e}^{2\pi \mathrm{i} \frac{\ell}{2}}|1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + \mathrm{e}^{2\pi \mathrm{i} \frac{\ell}{2^2}}|1\rangle}{\sqrt{2}} \otimes \cdots \otimes \frac{|0\rangle + \mathrm{e}^{2\pi \mathrm{i} \frac{\ell}{2^n}}|1\rangle}{\sqrt{2}}.$$

Ahora convertimos cada término en una suma

$$\frac{1}{\sqrt{2}} \sum_{k_1=0}^{1} \mathrm{e}^{2\pi \mathrm{i} k_1 \frac{\ell}{2}}|k_1\rangle \otimes \frac{1}{\sqrt{2}} \sum_{k_2=0}^{1} \mathrm{e}^{2\pi \mathrm{i} k_2 \frac{\ell}{2^2}}|k_2\rangle \otimes \cdots \otimes \frac{1}{\sqrt{2}} \sum_{k_n=0}^{1} \mathrm{e}^{2\pi \mathrm{i} k_n \frac{\ell}{2^n}}|k_n\rangle.$$

Colocando todas las sumas hacia el lado derecho y combinando los exponenciales, obtenemos

$$\frac{1}{\sqrt{2^n}} \sum_{k_1,\ldots,k_n=0}^{1} \mathrm{e}^{2\pi \mathrm{i}\ell\left(\frac{k_1}{2}+\cdots+\frac{k_n}{2^n}\right)} |k_1, ..., k_n\rangle,$$

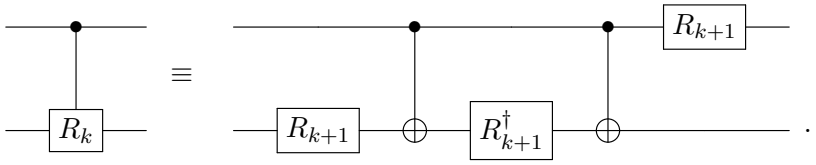
que es equivalente a

$$\frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} \mathrm{e}^{\frac{2\pi \mathrm{i}\ell k}{2^n}} |k\rangle.$$

Usando la definición de la transformada de Fourier dada en la Sec. 7.4, reconocemos que la última expresión es $F_{2^n}|\ell\rangle$.

Descomposición de $C(R_k)$

El circuito que proporciona la descomposición de $C(R_k)$ en términos de CNOTs y puertas de 1 qubit R_{k+1} es



Esta no es una descomposición en términos de puertas universales porque necesitamos escribir R_{k+1} en términos de un conjunto finito de puertas de 1 qubit. En la mayoría de los casos, no es necesario preocuparse por este paso, ya que lo realiza automáticamente el compilador de las computadoras cuánticas. R_k se puede implementar usando R_z, que es una rotación de la esfera de Bloch sobre el eje z. Tenga en cuenta que si k es grande, los errores impedirán que estas puertas funcionen correctamente a menos que existan códigos de corrección de errores.

Ahora mostramos que la descomposición de $C(R_k)$ en términos de CNOTs y puertas de 1 qubit R_{k+1} es correcta. Si la entrada a $C(R_k)$ es $|j\rangle|\ell\rangle$ entonces la salida es

$$|j\rangle|\ell\rangle \xrightarrow{\bullet - \boxed{R_k}} |j\rangle R_k^j|\ell\rangle.$$

Si $|j\ell\rangle$ es igual a $|00\rangle$ o $|01\rangle$ o $|10\rangle$, la salida es $|j\ell\rangle$ porque $R_k|0\rangle = |0\rangle$. La única salida no trivial se obtiene cuando la entrada es $|11\rangle$. En este caso, la salida es $\exp(2\pi\mathrm{i}/2^k)|11\rangle$.

Analicemos la descomposición (circuito de la derecha) para comprobar que el resultado es el mismo. Para las entradas $|00\rangle$ o $|01\rangle$ o $|10\rangle$, los CNOTs no están activados o se anulan. Además, dado que $R_k|0\rangle = |0\rangle$ y $R^\dagger_{k+1}R_{k+1} = I$ comprobamos que la salida es igual a la entrada. El caso que falta es la entrada $|11\rangle$. Sigamos cada paso a la vez usando $R_{k+1}|0\rangle = |0\rangle$ y $R_{k+1}|1\rangle = \exp(2\pi\mathrm{i}/2^{k+1})|1\rangle$:

$$|1\rangle|1\rangle \xrightarrow{I\otimes R_{k+1}} \mathrm{e}^{2\pi\mathrm{i}/2^{k+1}}|1\rangle|1\rangle \xrightarrow{\bullet-\oplus} \mathrm{e}^{2\pi\mathrm{i}/2^{k+1}}|1\rangle|0\rangle \xrightarrow{I\otimes R^\dagger_{k+1}} \mathrm{e}^{2\pi\mathrm{i}/2^{k+1}}|1\rangle|0\rangle \to \text{ - - -}$$

$$\text{- - -} \xrightarrow{\bullet-\oplus} \mathrm{e}^{2\pi\mathrm{i}/2^{k+1}}|1\rangle|1\rangle \xrightarrow{R_{k+1}\otimes I} \mathrm{e}^{2\pi\mathrm{i}/2^{k+1}}\mathrm{e}^{2\pi\mathrm{i}/2^{k+1}}|1\rangle|1\rangle = \mathrm{e}^{2\pi\mathrm{i}/2^{k}}|1\rangle|1\rangle.$$

Tenga en cuenta que la salida es $\exp(2\pi\mathrm{i}/2^k)|11\rangle$, que es igual a la salida del circuito de la izquierda.

7.9. Observaciones finales

En su artículo original [56], Shor mostró que la cantidad de veces que tenemos que volver a ejecutar el algoritmo hasta encontrar una solución es $O(\log \log r)$. Es posible reducir el número de repeticiones de la parte cuántica mejorando el posprocesamiento clásico [22].

Capítulo 8

Algoritmo de Shor para logaritmos discretos

El artículo completo sobre los algoritmos de Shor se publicó en el año 1997 [55] y describe no sólo un algoritmo para la factorización de enteros, sino también un algoritmo exponencialmente más rápido para logaritmos discretos. Es una contribución científica notable y célebre a la computación cuántica, pero el algoritmo para el logaritmo discreto no se ha descrito en muchos libros [30]. Algunos artículos aplican este algoritmo en la criptografía [29, 48].

8.1. Preliminares de teoría de números

El *logaritmo discreto* de b en base a, denotado por $\log_a b$, es el número de veces que a multiplicado por sí mismo es b, es decir, $a^s = b$. Aquí asumimos que a, b y s son números enteros positivos. Por ejemplo, $\log_2 8$ es 3 porque $2 \times 2 \times 2 = 8$. Donde $s = \log_a b$, escribimos

$$a^{\log_a b} = b.$$

Tenga en cuenta que hay opciones de a y b para que el logaritmo discreto $\log_a b$ no exista, por ejemplo, $\log_2 7$ porque no hay ningún entero s que satisfaga $2^s = 7$. Dos propiedades importantes del logaritmo en base a, cuando $\log_a b$ y $\log_a c$ existen, son

- $\log_a(bc) = \log_a b + \log_a c$;
- $\log_a(b^c) = c \log_a b$.

Hay dos inconvenientes en el contexto de los algoritmos cuánticos cuando usamos el conjunto de números enteros positivos $\mathbb{Z}^+$ para definir la noción de logaritmo discreto: (1) Dependiendo de la elección de a y b, $\log_a b$ puede no existir, y (2) si $\log_a b$ existe, existen métodos eficientes para calcular $\log_a b$ usando el hecho de que el logaritmo discreto es igual a la definición estándar de *logaritmo* de números reales.

En lugar de usar $\mathbb{Z}^+$, usemos un subconjunto del conjunto $\mathbb{Z}_N = \{0, 1, ..., N-1\}$, que tiene dos operaciones modulares: suma y multiplicación. Básicamente nos interesa el módulo de multiplicación N, donde $N > 1$. Para seleccionar el subconjunto deseado de $\mathbb{Z}_N$, necesitamos saber qué números de $\mathbb{Z}_N$ tienen inverso. Un número $a \in \mathbb{Z}_N$ tiene un inverso a^{-1} ($aa^{-1} \equiv 1$ mód N) sí y solo sí el máximo común divisor de a y N es 1. Si a tiene un inverso, el conjunto $G_a = \{a, a^2, ..., a^r\}$ es un *grupo cíclico*, donde r es el orden de a módulo N (order(a) es el

entero positivo más pequeño para que $a^r \equiv 1 \mod N$). El producto del grupo en este caso es el módulo de multiplicación N. Si seleccionamos cualquier elemento b en G_a, el $\log_a b$ existe. Por ejemplo, para $N = 34$ tenemos $G_{15} = \{15, 21, 9, 33, 19, 13, 25, 1\}$ y $\log_{15} 15 = 1$, $\log_{15} 21 = 2$, $\log_{15} 9 = 3$, $\log_{15} 33 = 4$, etc. En este contexto, nadie conoce un algoritmo clásico que calcule $\log_a b$ de manera eficiente (en tiempo polinomial) para un b arbitrario en G_a.

El algoritmo de factorización de Shor puede calcular $\log_a b$ de manera eficiente si el order(b) es un factor no trivial de order(a), porque $\log_a b$ =order(a)/order(b) en este caso. Antes de intentar calcular $\log_a b$ utilizando algoritmos de logaritmos discretos, verificamos si el orden de b divide el orden de a. De lo contrario, usamos un algoritmo cuántico que calcula $\log_a b$ de manera eficiente para un b arbitrario en G_a.

La estrategia principal en la parte del algoritmo de Shor que calcula el orden de $a \in \mathbb{Z}_N$ es definir la función

$$\begin{aligned} f\colon \mathbb{Z}_{2^m} &\to \mathbb{Z}_N \\ x &\mapsto a^x, \end{aligned}$$

donde $m \approx 2n$, $n = \lceil \log_2 N \rceil$, y aprovechar el hecho de que f es r-periódico. Sabemos que esta función se puede implementar en una computadora cuántica con alrededor de $3n$ qubits usando el operador unitario U_f definido por $U_f|x\rangle|y\rangle = |x\rangle|y \oplus (a^x \mod N)\rangle$. Es posible determinar r de manera eficiente con una aplicación de la transformada discreta inversa de Fourier $F_{2^m}^\dagger$ al primer registrador después de haber aplicado U_f a una superposición de todos los estados de la base computacional del primer registrador (el segundo registrador se establece inicialmente en $|0\rangle$) . El método funciona porque f tiene el período r.

Es natural preguntarse si se puede usar la misma estrategia para calcular s sabiendo que $a^s \equiv b \mod N$. El problema es que no existe la función s-periódica $f(x)$ con dominio $\mathbb{Z}_{2^m}$ y codominio $\mathbb{Z}_N$ que utilice únicamente los parámetros a y b. La salida es usar una función $f(x, y)$ con dos variables definidas como

$$\begin{aligned} f\colon \mathbb{Z}_r \times \mathbb{Z}_r &\to \mathbb{Z}_N \\ (x, y) &\mapsto a^x b^y. \end{aligned}$$

Esta definición tiene dos diferencias principales: (1) El dominio de cada variable es $\mathbb{Z}_r$ en lugar de $\mathbb{Z}_{2^m}$, y (2) f tiene dos variables. Respecto a (1), es importante utilizar el dominio $\mathbb{Z}_r$; de lo contrario, la aritmética modular no proporcionaría la respuesta correcta al final del algoritmo. Esto significa que r debe calcularse previamente y si r no es una potencia de 2, la implementación de la transformada de Fourier F_r requiere trabajo adicional. Respecto a (2), f es periódico de la siguiente forma:

$$f(x, y) = f(x + kr - \ell s, y + \ell),$$

donde $k, \ell \in \mathbb{Z}_r$. De hecho,

$$a^x b^y \equiv a^x (b)^{-\ell} b^y b^\ell \equiv a^x a^{kr} (a^s)^{-\ell} b^y b^\ell \equiv a^{x+kr-\ell s} b^{y+\ell} \mod N,$$

donde hemos usado que $a^r \equiv 1$ y $a^s \equiv b \mod N$. Para comprender mejor la periodicidad de $f(x, y)$, tenemos que utilizar el concepto de redes y subredes en espacios vectoriales finitos.

8.2. Red en espacios vectoriales finitos

Hay muchos tipos de redes en matemáticas. Las que nos interesan son las redes en espacios vectoriales finitos, en particular, las redes bidimensionales que son subgrupos aditivos del espacio

vectorial $\mathbb{Z}_r \times \mathbb{Z}_r$ (denominado $\mathbb{Z}_r^2$ en adelante). Este espacio vectorial es el conjunto de vectores con dos entradas cada uno en $\mathbb{Z}_r$, cuya base canónica es $\{\mathbf{e}_0, \mathbf{e}_1\}$, donde

$$\mathbf{e}_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \text{ y } \mathbf{e}_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Esto significa que cualquier vector $\mathbf{v}$ en $\mathbb{Z}_r^2$ es una combinación lineal de $\mathbf{e}_0$ y $\mathbf{e}_1$, es decir, hay escalares $k, \ell \in \mathbb{Z}_r$ de modo que $\mathbf{v} = k\mathbf{e}_0 + \ell\mathbf{e}_1$.

Una red L en $\mathbb{Z}_r^2$ se define como un conjunto

$$L = \{k\mathbf{r} + \ell\mathbf{s} : k, \ell \in \mathbb{Z}_r\},$$

donde $\mathbf{r}$ y $\mathbf{s}$ son dos vectores linealmente independientes en $\mathbb{Z}_r^2$. El conjunto $\{\mathbf{r}, \mathbf{s}\}$ es una base de L. Diferentes bases pueden abarcar la misma red. Tenga en cuenta que $\mathbb{Z}_r^2$ en sí mismo es una red en $\mathbb{Z}_r^2$. Entonces, L es una subred de $\mathbb{Z}_r^2$.

Los vectores se pueden representar por puntos. Por ejemplo, los vectores

$$\mathbf{r} = \begin{pmatrix} 16 \\ 0 \end{pmatrix} \text{ y } \mathbf{s} = \begin{pmatrix} 5 \\ 1 \end{pmatrix}$$

se muestran en la Fig. 8.1 como puntos $(x, y) = (16, 0)$ y $(5, 1)$. La figura también muestra la red con base $\{(16, 0), (5, 1)\}$ en $\mathbb{Z}_{16}^2$. Como ejemplo de aplicación al cálculo de logaritmos discretos,

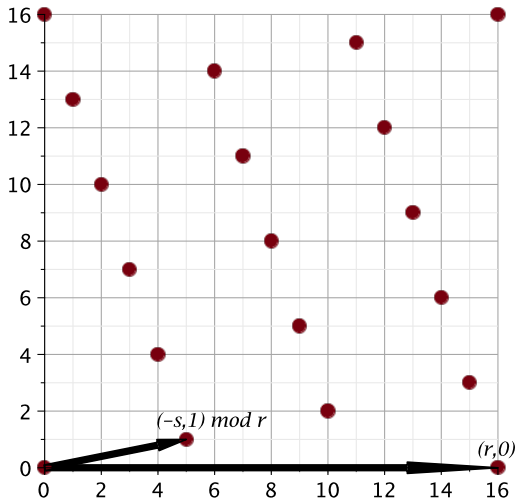


Figura 8.1: Red con las bases $\mathbf{r} = (16, 0)$ y $\mathbf{s} = (-11, 1)$ módulo 16. Los bordes son cíclicos.

considere nuevamente $N = 34$ pero esta vez tome $a = 27$ y

$$G_{27} = \{27, 15, 31, 21, 23, 9, 5, 33, 7, 19, 3, 13, 11, 25, 29, 1\}.$$

Supongamos que queremos calcular $\log_{27} 3$, cuya respuesta sabemos que es $s = 11$. Dado que el orden de $a = 27$ es $r = 16$ módulo 34, la red L con base $\mathbf{r} = (r, 0)$ y $\mathbf{s} = (-s, 1)$ que se muestra en la Fig. 8.1 representa todos los puntos (x, y) en $\mathbb{Z}_{16}^2$ de modo que $f(x, y) = f(0, 0)$, donde $f(x, y) = a^x b^y \mod 34$, $a = 27$, y $b = 3$. De hecho, $f(k\mathbf{r} + \ell\mathbf{s}) = f(kr - \ell s, \ell) = f(0, 0)$. Entonces, f es una función periódica donde el período depende tanto de s como de r. La red L

es un subgrupo del grupo aditivo $\mathbb{Z}_{16}^2$. Esto significa que podemos particionar $\mathbb{Z}_{16}^2$ en clases de equivalencia, las clases laterales de L en $\mathbb{Z}_{16}^2$, y podemos seleccionar los siguientes representantes: $(0,0)$, $(1,0)$, ..., $(r-1,0)$. La imagen de f en cualquier punto $(x,y) \in \mathbb{Z}_r^2$ es igual a la imagen f en uno de los representantes. Es decir, $f(x - \ell s, \ell) = f(x,0)$ para $0 \leq \ell < r$, que corresponde a desplazar las unidades x de la red a la derecha con condiciones de frontera cíclicas, es decir, sumar $(x,0)$ a cada elemento de L módulo 16. Cuando arreglamos x y ejecutamos ℓ de 0 a $r-1$ cubrimos el conjunto con el representante $(x,0)$.

8.3. Caso especial: El orden de a es una potencia de 2

Sean N, a y b enteros positivos conocidos y sea s un entero positivo tal que $a^s \equiv b \mod N$ y el $\text{mcd}(a,N) = 1$. Nuestro objetivo es encontrar s dados N, a y b como entrada. Sea r el orden de a módulo N, que puede determinarse eficientemente usando el algoritmo de factorización de Shor. Abordamos en esta Sección el caso $r = 2^m$ para algún entero m, lo que significa que r es una potencia de 2. En este caso, hay un algoritmo clásico eficiente llamado *El algoritmo de Pohlig–Hellman* que es capaz de calcular s en tiempo polinomial. Describimos el algoritmo cuántico para este caso porque la transformada de Fourier F_r se puede implementar de manera sencilla en una computadora cuántica basada en qubits y el análisis del algoritmo es más fácil que el caso general.

Sea f una función de dos variables con dominio $\mathbb{Z}_r \times \mathbb{Z}_r$ y codominio $\mathbb{Z}_N$ definidos como

$$f(x,y) = a^x b^y \mod N.$$

Hemos demostrado que f es periódico de la siguiente manera:

$$f(x,y) = f(x + kr - \ell s, y + \ell).$$

Usando f, definimos un operador unitario de 3 registradores

$$U_f|x\rangle|y\rangle|z\rangle \;=\; |x\rangle|y\rangle|z \oplus f(x,y)\rangle, \tag{8.1}$$

donde el primer y segundo registrador tienen m qubits cada uno y el tercer registrador tiene $n = \lceil \log_2 N \rceil$ qubits. La aritmética con las variables del primer y segundo registrador se realiza con módulo r. La aritmética para calcular la imagen $f(x,y)$ se realiza con módulo N. El símbolo $\oplus$ representa la operación xor bit a bit. El algoritmo que calcula el logaritmo discreto cuando r es una potencia de 2 se describe en el Algoritmo 8.1 y el circuito se muestra en la Fig. 8.2.

Análisis del algoritmo

En el paso 2, aplicamos $H^{\otimes m} \otimes H^{\otimes m} \otimes I$ a $|0\rangle|0\rangle|0\rangle$ obteniendo

$$|\psi_1\rangle = \frac{1}{r} \sum_{x,y=0}^{r-1} |x\rangle|y\rangle|0\rangle.$$

En el paso 3, aplicamos U_f a $|\psi_1\rangle$ obteniendo

$$|\psi_2\rangle = \frac{1}{r} \sum_{x,y=0}^{r-1} |x\rangle|y\rangle|f(x,y)\rangle.$$

Algoritmo 8.1: Algoritmo del logaritmo discreto cuando r es una potencia de 2

Entrada: N, a, b y r (orden de a).
Salida: $s = \log_a b$ con probabilidad 1/2, donde $a^s \equiv b \mod N$

1 Prepare el estado inicial $|0\rangle^{\otimes m}|0\rangle^{\otimes m}|0\rangle^{\otimes n}$, donde $m = \log_2 r$ y $n = \lceil \log_2 N \rceil$;
2 Aplicar $H^{\otimes m} \otimes H^{\otimes m}$ al primer y segundo registrador;
3 Aplique U_f como se define en la Ec. (8.1);
4 Mida el tercer registrador en la base computacional;
5 Aplicar $F_r^\dagger \otimes F_r^\dagger$ al primer y segundo registrador;
6 Mida el primer y segundo registrador en la base computacional, donde r_1 y r_2 son los resultados;
7 Si el $\mathrm{mcd}(r_1, r) = 1$, devuelve $s \equiv r_2/r_1 \mod r$; de lo contrario, Falla.

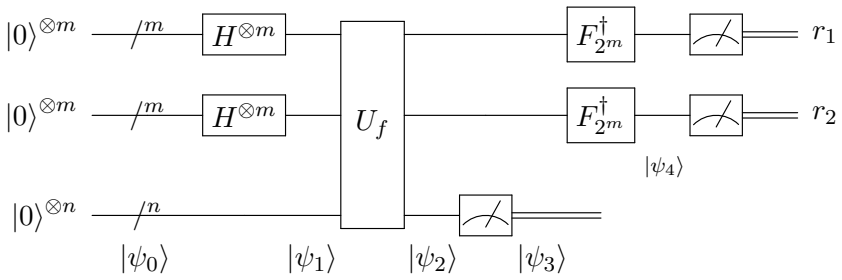


Figura 8.2: Circuito del algoritmo del logaritmo discreto cuando r es una potencia de 2, donde $m = \log_2 r$, $n = \lceil \log_2 N \rceil$, y U_f es definido por la Ec. (8.1). Si el $\mathrm{mcd}(r_1, r) = 1$, retorna $s \equiv r_2/r_1$ mód r; caso contrario, Falla. La probabilidad de devolver el resultado correcto es 1/2.

Antes de continuar, simplificamos $|\psi_2\rangle$ lo más posible usando la periodicidad de la función f. Dado que la imagen de f en todos los puntos $(x - \ell s, \ell)$ para $0 \leq \ell < r$ es igual a la imagen de $(x, 0)$, podemos recopilar el tercer registrador como

$$|\psi_2\rangle = \frac{1}{r} \sum_{x=0}^{r-1} \left(\sum_{\ell=0}^{r-1} |x - \ell s\rangle |\ell\rangle \right) |f(x, 0)\rangle.$$

En el paso 4, medimos el tercer registrador en la base computacional, obteniendo una imagen de uno de los representantes de las clases laterales de L en $\mathbb{Z}_r^2$ de la siguiente manera

$$|\psi_3\rangle = \frac{1}{\sqrt{r}} \left(\sum_{\ell=0}^{r-1} |x_0 - \ell s\rangle |\ell\rangle \right) |f(x_0, 0)\rangle.$$

En el paso 5, aplicamos $F_r^\dagger \otimes F_r^\dagger$ al primer y segundo registrador (despreciamos el tercer registrador) obteniendo

$$|\psi_4\rangle = \frac{1}{\sqrt{r}} \sum_{\ell=0}^{r-1} \left(F_r^\dagger |x_0 - \ell s\rangle \right) \left(F_r^\dagger |\ell\rangle \right).$$

Ahora simplificamos $|\psi_4\rangle$ tanto como sea posible. Usando la definición de la transformada de Fourier $F_r^\dagger$, obtenemos

$$|\psi_4\rangle = \frac{1}{\sqrt{r}} \sum_{\ell=0}^{r-1} \left(\frac{1}{\sqrt{r}} \sum_{x=0}^{r-1} \omega_r^{x(x_0 - \ell s)} |x\rangle \right) \left(\frac{1}{\sqrt{r}} \sum_{y=0}^{r-1} \omega_r^{y\ell} |y\rangle \right),$$

donde $\omega_r = \exp(2\pi \mathrm{i}/r)$. Cambiando el orden de las sumas colocando $\sum_{x,y}$ hacia la izquierda, y colocando $\sum_\ell$ y el término $\omega_r^{-x\ell s}$ al segundo registrador, obtenemos

$$|\psi_4\rangle = \frac{1}{r\sqrt{r}} \sum_{x,y=0}^{r-1} \omega_r^{xx_0} |x\rangle \sum_{\ell=0}^{r-1} \omega_r^{\ell(-xs+y)} |y\rangle.$$

Usando

$$\frac{1}{r} \sum_{\ell=0}^{r-1} \left(\omega_r^{-xs+y}\right)^\ell = \begin{cases} 1, & \text{si } y = xs, \\ 0, & \text{caso contrario,} \end{cases}$$

obtenemos

$$|\psi_4\rangle = \frac{1}{\sqrt{r}} \sum_{x,y=0}^{r-1} \omega_r^{xx_0} |x\rangle \, \delta_{y,xs} |y\rangle,$$

y simplificando la suma sobre y, obtenemos

$$|\psi_4\rangle = \frac{1}{\sqrt{r}} \sum_{x=0}^{r-1} \omega_r^{xx_0} |x\rangle \, |xs\rangle.$$

En el paso 6, medimos el primer y segundo registrador en la base computacional y obtenemos dos resultados: r_1 y $r_2 = r_1 s$, donde r_1 se elige en el intervalo $[0, r-1]$ uniformemente al azar.

En el paso 7, si el $\mathrm{mcd}(r_1, r) = 1$, calculamos $s \equiv r_2/r_1 \mod r$. Dado que $s \leq r < N$, s satisface $a^s \equiv b \mod N$. Para cualquier impar r_1, el $\mathrm{mcd}(r_1, r) = 1$. Dado que la mitad de los valores de r_1 son impares, la probabilidad de éxito es $1/2$.

Capítulo 9

Algoritmo de Grover

El algoritmo de Grover [23, 24] es un algoritmo de búsqueda desarrollado inicialmente para datos no estructurados. También se puede describir en términos de un oráculo, que es una función con alguna promesa o propiedad que se puede evaluar tantas veces como queramos, y nuestro objetivo es determinar la propiedad que tiene la función. Este Capítulo sigue la última descripción con el foco en el modelo de circuito. El algoritmo de Grover es óptimo, es decir, no se puede mejorar [5, 66]. Se describe en muchos libros [3, 4, 6, 26, 28, 30, 32, 33, 37, 42, 46, 51, 60, 62].

9.1. Formulación del problema en términos de un oráculo

Sea N una potencia de 2 para algún entero n, es decir, $N = 2^n$. Supongamos que $f : \{0, \ldots, N-1\} \to \{0, 1\}$ es una función booleana tal que $f(x) = 1$ sí y solo sí $x = x_0$ para algún valor fijo x_0, es decir,

$$f(x) = \begin{cases} 1, & \text{si } x = x_0, \\ 0, & \text{caso contrario.} \end{cases}$$

Supongamos que x_0 nos es desconocido. ¿Cómo podemos encontrar x_0 evaluando f? Desde un punto de vista computacional, queremos evaluar f lo menos posible.

Clásicamente, el algoritmo más eficiente consulta esta función N veces en el peor de los casos, es decir, la complejidad del algoritmo clásico en cuanto al número de consultas es $O(N)$. ¿Cómo se hace en la práctica? Le pedimos a otra persona que genere un número aleatorio de$n$-bits x_0. Esta persona nos oculta x_0 y hace una subrutina compilada de f. Podemos usar la subrutina tantas veces como queramos, pero no podemos hackear el código en busca de x_0. El algoritmo clásico que resuelve este problema es una iteración que consulta $f(x)$ por x de 0 a $2^n - 1$. Tan pronto como $f(x)$ es 1, el programa devuelve x_0.

Cuánticamente, es posible mejorar la complejidad de la consulta a $O(\sqrt{N})$. ¿Cómo se hace en la práctica? Tenemos que pedirle a alguien nuevamente que genere un número aleatorio de n-bits x_0 y defina f. Esta persona, el *oráculo*, implementa f a través de una matriz unitaria U_f en una computadora cuántica. Podemos usar U_f, pero no podemos ver los detalles de la implementación de U_f. Cada vez que usamos U_f, agregamos una unidad al conteo. Podemos usar puertas adicionales que obviamente no dependen de x_0.

Tenemos el mismo problema que se puede resolver mediante algoritmos ejecutados en dos máquinas diferentes. En el primer caso, se utiliza una computadora clásica con $O(n)$ bits y la solución se encuentra después de $O(N)$ pasos. En el segundo caso, se utiliza una computadora cuántica con $O(n)$ qubits y la solución se encuentra después de $O(\sqrt{N})$ pasos. Esta ganancia en

complejidad justifica la inversión en hardware cuántico y el estudio de técnicas para desarrollar algoritmos cuánticos, que necesariamente tienen que utilizar la superposición de estados. En el caso del algoritmo de Grover, U_f actuando sobre una superposición de estados evalúa f en más de un punto del dominio al mismo tiempo con los mismos recursos disponibles. La computadora clásica necesita un número exponencial de procesadores para realizar esta tarea con la misma eficiencia.

9.2. Como implementar el oráculo en una computadora cuántica

El primer paso para desarrollar un algoritmo cuántico que resuelva el problema de Grover es la implementación de la función f. Dado que f es una función booleana cuya tabla de verdad tiene una sola fila con salida 1, f se puede implementar con una puerta Toffoli multiqubit activada por x_0, como se describe en la Sec. 2.7 de la Página 23. Esta puerta tiene una matriz unitaria asociada U_f, que se define por su acción en la base computacional como

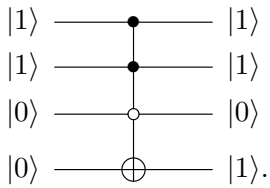
$$U_f|x\rangle|i\rangle = |x\rangle|i \oplus f(x)\rangle,$$

donde x es una cadena de n-bits y i es un bit. El primer registrador tiene n qubits y el segundo registrador tiene un qubit. Tenga en cuenta que si tomamos $i = 0$, la ecuación anterior se reduce a

$$U_f|x\rangle|0\rangle = \begin{cases} |x_0\rangle|1\rangle, & \text{si } x = x_0, \\ |x\rangle|0\rangle, & \text{caso contrario,} \end{cases}$$

que describe la salida de una puerta Toffoli multiqubit activada por x_0 (y sólo por x_0) cuando la entrada es $|x\rangle|0\rangle$. El resultado del cálculo de $f(x)$ se almacena en el segundo registrador mientras que el estado del primer registrador permanece sin cambios.

Por ejemplo, el circuito que implementa U_f en el caso $N = 8$ y $x_0 = 6$, es decir $f(110) = 1$ y $f(j) = 0$ si $j \neq 110$, es



El primer registrador tiene tres qubits. Tenga en cuenta que el estado del segundo registrador cambia de $|0\rangle$ a $|1\rangle$ sólo si la entrada al primer registrador es $|110\rangle$ porque 110 activa los tres controles y todas las demás cadenas de 3 bits no lo hacen.

En el algoritmo de Grover, el estado del segundo registrador siempre es

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

Usando linealidad, la acción de U_f viene dada por

$$U_f|x\rangle|-\rangle = \begin{cases} -|x_0\rangle|-\rangle, & \text{si } x = x_0, \\ |x\rangle|-\rangle, & \text{caso contrario.} \end{cases}$$

El mismo resultado se obtiene de la Proposición 3.1 de la Página 29, que establece que

$$U_f|x\rangle|-\rangle = (-1)^{f(x)}|x\rangle|-\rangle.$$

9.3. El algoritmo

El algoritmo de Grover usa un operador adicional definido como

$$G = \left(2\,|\mathrm{d}\rangle\langle\mathrm{d}| - I_N\right) \otimes I_2,$$

donde

$$|\mathrm{d}\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} |j\rangle.$$

La notación "$|\mathrm{d}\rangle\langle\mathrm{d}|$" se llama *producto externo* entre el vector $|\mathrm{d}\rangle$ (una matriz $N \times 1$) y el vector dual $\langle\mathrm{d}|$ (una matriz $1 \times N$). El producto externo es equivalente al producto de matrices. Multiplicar una matriz $N \times 1$ por una matriz $1 \times N$ da como resultado una matriz $N \times N$. Por lo tanto, $|\mathrm{d}\rangle\langle\mathrm{d}|$ es una matriz $N \times N$, dada por

$$|\mathrm{d}\rangle\langle\mathrm{d}| = \frac{1}{N} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix},$$

y

$$\left(2\,|\mathrm{d}\rangle\langle\mathrm{d}| - I_N\right) = \frac{1}{N} \begin{bmatrix} (2-N) & 2 & \cdots & 2 \\ 2 & (2-N) & \cdots & 2 \\ \vdots & \vdots & \ddots & \vdots \\ 2 & 2 & \cdots & (2-N) \end{bmatrix}$$

que se llama matriz de Grover (u operador de Grover).

El algoritmo de Grover se describe en el Algoritmo 9.1.

Algoritmo 9.1: Algoritmo de Grover

Entrada: Un entero N y una función $f : \{0, ..., N-1\} \to \{0, 1\}$ tal que $f(x) = 1$ sólo para un punto $x = x_0$ en el dominio.

Salida: x_0 con probabilidad igual o mayor que $1 - \frac{1}{N}$.

1 Preparar el estado inicial $|\mathrm{d}\rangle|-\rangle$;

2 Aplicar $(G\,U_f)^t$, donde $t = \left\lfloor \frac{\pi}{4}\sqrt{N} \right\rfloor$;

3 Medir el primer registrador en la base computacional.

9.4. Circuito no económico del algoritmo de Grover

El objetivo de esta sección es encontrar el circuito que implementa el operador de Grover utilizando nuestro conocimiento de la implementación de funciones booleanas. El circuito usa más qubits de los necesarios, pero más adelante mostramos como obtener una versión económica del circuito.

Para obtener el circuito, tenemos que hacer una manipulación algebraica con la expresión de la matriz de Grover $\left(2\,|\mathrm{d}\rangle\langle\mathrm{d}| - I_N\right)$. Tenga en cuenta que

$$|\mathrm{d}\rangle = H^{\otimes n}|0\rangle$$

donde $|0\rangle$ está en notación decimal y $H^{\otimes n} = H \otimes \cdots \otimes H$. Transponiendo la ecuación anterior, obtenemos

$$\langle \mathrm{d}| = \langle 0|H^{\otimes n}.$$

Usando $(H^{\otimes n}) \cdot (H^{\otimes n}) = (H \cdot H)^{\otimes n} = (I_2)^{\otimes n} = I_N$, obtenemos

$$\big(2\,|\mathrm{d}\rangle\langle \mathrm{d}| - I_N\big) = H^{\otimes n}\big(2\,|0\rangle\langle 0| - I_N\big)H^{\otimes n},$$

donde

$$\big(2\,|0\rangle\langle 0| - I_N\big) = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & -1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & -1 \end{bmatrix}.$$

La matriz $\big(2\,|0\rangle\langle 0| - I_N\big)$ actúa sólo sobre el primer registrador. Sin embargo, es más sencillo implementar esta matriz utilizando ambos registradores. Demostremos que está implementado por una puerta Toffoli multiqubit activada por 0. De hecho, la acción de $\big(2\,|0\rangle\langle 0| - I_N\big)$ en $|x\rangle$, donde x es una cadena de n-bits, es

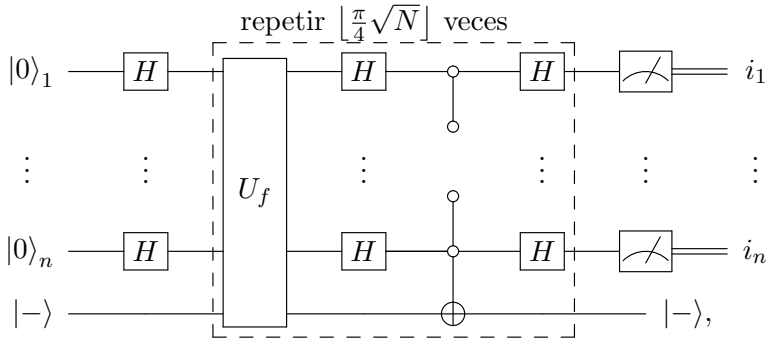
$$\big(2\,|0\rangle\langle 0| - I_N\big)|x\rangle = \begin{cases} |0\rangle, & \text{si } x = 0, \\ -|x\rangle, & \text{caso contrario.} \end{cases}$$

Por lo tanto, la acción de $\big(2\,|0\rangle\langle 0| - I_N\big)$ en el primer registrador es la misma que la acción de $(-U_{f'})$ en ambos registradores, cuando $x_0 = 0$ (el estado del segundo registrador debe ser $|-\rangle$), donde

$$f'(x) = \begin{cases} 1, & \text{si } x = 0, \\ 0, & \text{caso contrario.} \end{cases}$$

El signo menos en $(-U_{f'})$ no cambia ni el resultado del algoritmo, ni la probabilidad final. Es decir, usar G o $-G$ en el algoritmo de Grover no cambia el resultado final.

Usando estos resultados algebraicos, concluimos que un circuito que implementa el algoritmo de Grover es

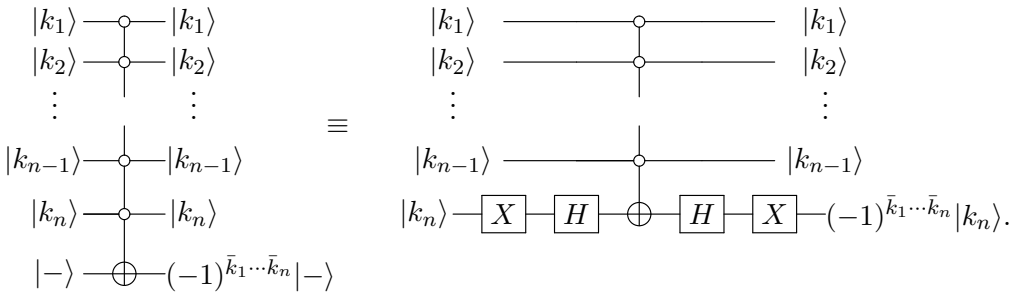


donde los bits i_1, ..., i_n son las salidas de las mediciones. Esos bits son el bit de x_0, es decir $x_0 = (i_1 \ldots i_n)_2$, con alta probabilidad.

9.5. Circuito económico del algoritmo de Grover

El segundo registrador del algoritmo de Grover se puede descartar ya que es posible hacer una implementación más económica del oráculo [34]. Comencemos mostrando como implementar

el operador $\left(2\left|0\right\rangle\langle 0\right| - I_N\right)$ (módulo de la fase global), que nos permite implementar el operador G usando sólo el primer registrador. Demostremos la equivalencia de los siguientes circuitos:



La salida del circuito de la izquierda es el estado de $(n+1)$-qubits

$$|k_1\rangle \otimes \cdots \otimes |k_n\rangle \otimes \left((-1)^{\bar{k}_1\cdots\bar{k}_n}|-\rangle\right),$$

que se obtiene de la definición de la puerta Toffoli multiqubit activada solo cuando los qubits $k_1, \ldots, k_n$ se establecen en 0. El producto de Kronecker tiene la propiedad

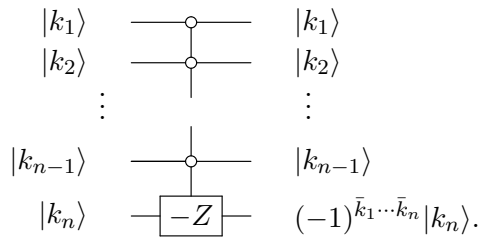
$$|v_1\rangle \otimes (a|v_2\rangle) = (a|v_1\rangle) \otimes |v_2\rangle = a(|v_1\rangle \otimes |v_2\rangle),$$

para cualquier vector $|v_1\rangle$, $|v_2\rangle$ y cualquier escalar a. Luego, movemos el término escalar $(-1)^{\bar{k}_1\cdots\bar{k}_n}$ al primer registrador cuya salida es

$$|k_1\rangle \otimes \cdots \otimes |k_{n-1}\rangle \otimes \left((-1)^{\bar{k}_1\cdots\bar{k}_n}|k_n\rangle\right) \;=\; (-1)^{\bar{k}_1\cdots\bar{k}_n}|k_1\rangle \otimes \cdots \otimes |k_n\rangle.$$

El estado del segundo registrador es $|-\rangle$ y este registrador se descartará al final del proceso.

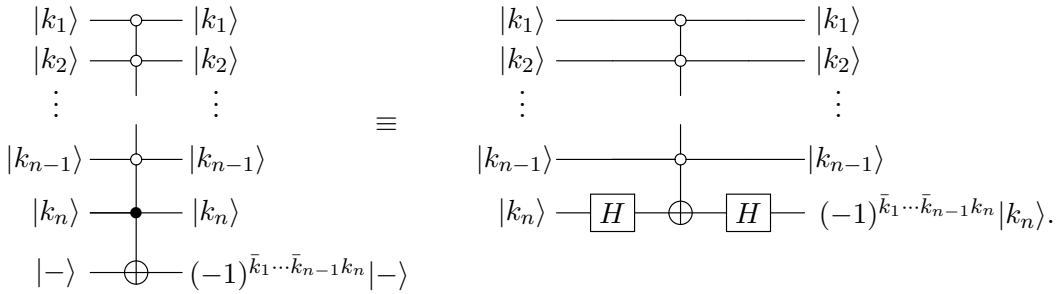
Ahora mostramos que la salida del circuito de la derecha es la misma. Usamos el hecho de que $XHXHX = -Z$, por lo tanto, el circuito de la derecha es equivalente a



La salida se obtiene usando $(-Z)|k_n\rangle = (-1)^{\bar{k}_n}|k_n\rangle$, y dado que $-Z$ se activa sólo cuando los qubits $k_1, \ldots, k_{n-1}$ se establecen en 0, obtenemos la salida general $(-1)^{\bar{k}_1\cdots\bar{k}_{n-1}\bar{k}_n}|k_1\cdots k_{n-1}k_n\rangle$. En conclusión, el resultado del primer circuito (después de la eliminación del segundo registrador) es el mismo que el resultado del segundo circuito. Ya que vamos a realizar una medición del primer registrador solamente, esto completa la prueba de que podemos reemplazar *sin ninguna pérdida* el primer circuito con el segundo en el algoritmo de Grover. Este reemplazo no es válido en todos los algoritmos. Hasta ahora hemos demostrado que G (módulo de la fase global) se puede implementar usando sólo el primer registrador.

Consideremos el oráculo. Si el oráculo elige $x_0 = 0$, el circuito de U_f es una puerta Toffoli multiqubit que se activa solo cuando todos los qubits del primer registrador se establecen en

0. En este caso, ya hemos mostrado como implementar U_f usando sólo el primer registrador. El oráculo usaría el circuito de la mano derecha representado al comienzo de esta Sección. Si el oráculo elige $x_0 = 1$, el circuito de U_f es una puerta de Toffoli multiqubit que se activa solo cuando todos los qubits del primer registrador se establecen en 0 excepto en el n-ésimo qubit, que se establece en 1. En este caso, tenemos el siguiente circuito de equivalencia:



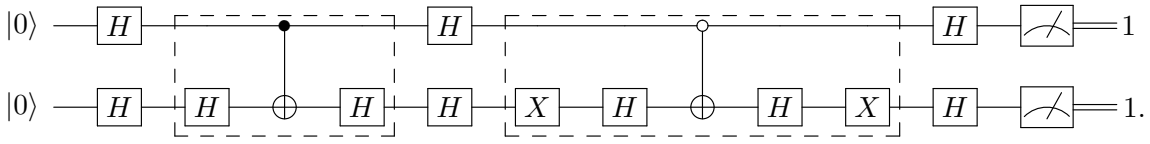
La verificación de equivalencia es similar al caso anterior, pero ahora se obtiene usando $HXH = Z$ y $Z|k_n\rangle = (-1)^{k_n}|k_n\rangle$, y dado que Z se activa sólo cuando los qubits $k_1, k_2, \ldots, k_{n-1}$ se establecen en 0, obtenemos la salida general $(-1)^{\bar{k}_1\bar{k}_2\cdots\bar{k}_{n-1}k_n}|k_1k_2\cdots k_n\rangle$. Esto concluye la prueba de que el oráculo usaría un circuito de n-qubits si $x_0 = 1$.

El resto de casos, $x_0 \geq 2$, se obtiene de los resultados anteriores. Si el (último) bit más a la derecha de x_0 es 0, usamos la equivalencia de circuito descrita al principio de esta Sección de la siguiente manera: Si cualquier control (excepto el n-ésimo) en el circuito de la izquierda cambia de círculo vacío a un círculo lleno, lo mismo debe ocurrir con los controles del circuito de la derecha. Si el bit más a la derecha de x_0 es 1, usamos la equivalencia del segundo circuito de esta Sección, y de la misma manera si cualquier control (excepto el n-ésimo) en el circuito de la izquierda cambia de círculo vacío a círculo completo, lo mismo debe ocurrir. suceda con los controles en el circuito de la derecha. La expresión de la salida cambia en consecuencia.

Así, no solo G sino también U_f se puede implementar con n qubits eliminando el segundo registrador e introduciendo dos puertas Hadamard más dos puertas X Pauli, si el n-ésimo qubit está activado por 0, e introduciendo sólo dos puertas Hadamard si el n-ésimo qubit es activado por 1.

Circuito económico cuando $N = 4$

El circuito del algoritmo de Grover en la forma económica cuando $N = 4$ y $x_0 = 11$ es



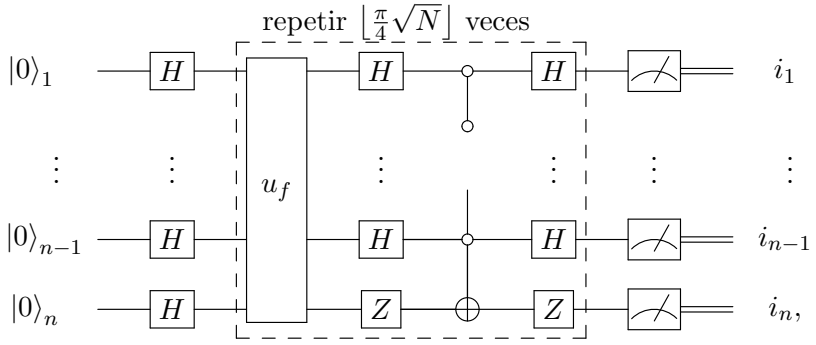
Las puertas dentro del primer cuadro discontinuo implementan el oráculo y las puertas dentro del segundo cuadro punteado implementan $\left(2\,|0\rangle\langle 0| - I_N\right)$ (módulo de la fase global). Este circuito se puede simplificar sustituyendo HXH en el segundo qubit con Z en dos lugares.

El objetivo del algoritmo de Grover es determinar x_0 consultando el oráculo, es decir, utilizando el primer cuadro discontinuo, sin mirar los detalles de implementación. Tenemos que suponer que el primer cuadro punteado es un cuadro negro. Cuando $N = 4$, hay cuatro cajas negras posibles, el caso $x_0 = 11$ es una de ellas. Las puertas que implementan el oráculo cuando $x_0 = 00$, $x_0 = 01$ y $x_0 = 10$ son $(X \otimes XH)\text{CNOT}(X \otimes HX)$, $(X \otimes H)\text{CNOT}(X \otimes H)$

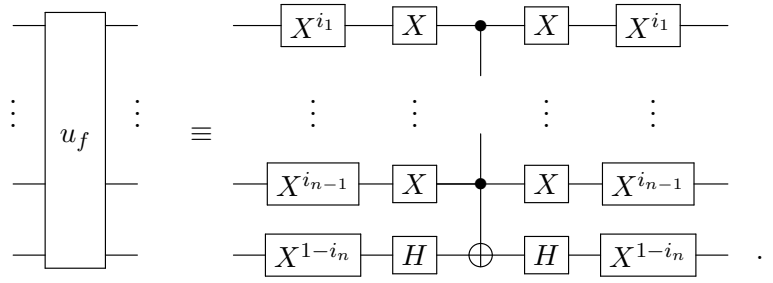
y $(I \otimes XH)\text{CNOT}(I \otimes HX)$, respectivamente. Cuando $N = 4$, la salida es la correcta con probabilidad 1.

Circuito económico para un N arbitrario

Para un N arbitrario ($N = 2^n$ y $n \geq 2$), el circuito del algoritmo de Grover con solo n qubits es



donde la matriz u_f es la versión económica de U_f y el circuito para u_f para un $x_0 = (i_1 \dots i_n)_2$ arbitrario es



9.6. Análisis del algoritmo de Grover

¿Por qué el algoritmo de Grover funciona correctamente? Respondemos a esta pregunta utilizando la forma económica del algoritmo. Los operadores en este caso son u_f, que se define como

$$u_f = \sum_x (-1)^{f(x)} |x\rangle\langle x|,$$

y

$$g = 2\,|\mathrm{d}\rangle\langle\mathrm{d}| - I_N.$$

El operador u_f es la versión económica de U_f, es decir, u_f es un operador N-dimensional cuya acción sobre la base computacional es

$$u_f|x\rangle = \begin{cases} -|x_0\rangle, & \text{si } x = x_0, \\ |x\rangle, & \text{caso contrario.} \end{cases}$$

Por su parte, g es la versión económica del operador G. La versión económica del algoritmo de Grover se describe en el Algoritmo 9.2.

El objetivo del algoritmo es encontrar x_0, que es una cadena de n bits. Se logrará si el estado de los qubits justo antes de la medición es $|x_0\rangle$ porque la medición en este caso devuelve x_0. El

Algoritmo 9.2: Algoritmo de Grover (versión económica)

Entrada: Un entero N (potencia de 2) y una función $f : \{0, ..., N-1\} \to \{0, 1\}$ tal que $f(x) = 1$ sólo para un punto $x = x_0$ en el dominio.

Salida: x_0 con probabilidad igual o mayor que $1 - \frac{1}{N}$.

1 Preparar el estado inicial $|\mathrm{d}\rangle$;

2 Aplicar $(g\, u_f)^t$, donde $t = \left\lfloor \frac{\pi}{4}\sqrt{N} \right\rfloor$;

3 Mida todos los qubits en la base computacional.

análisis del algoritmo que ahora empezamos a describir se basa en una interpretación geométrica de reflexiones vectoriales [1]. Al comienzo del algoritmo, el estado de los qubits es $|\mathrm{d}\rangle$. Para N grande, $|\mathrm{d}\rangle$ es casi ortogonal a $|x_0\rangle$. La Fig. 9.1 muestra los vectores $|\mathrm{d}\rangle$ y $|x_0\rangle$, donde $\theta/2$ es el ángulo entre $|\mathrm{d}\rangle$ y el eje horizontal. Cualquier otra representación de esos vectores se puede utilizar en el análisis del algoritmo siempre que $|\mathrm{d}\rangle$ sea casi ortogonal a $|x_0\rangle$.

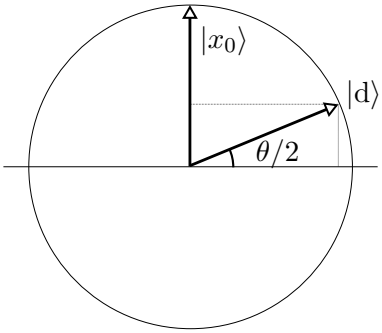


Figura 9.1: Representación del vector $|x_0\rangle$ y $|\mathrm{d}\rangle$.

El ángulo θ es muy pequeño para N grande y en este caso, $\theta/2$ es una buena aproximación de $\sin(\theta/2)$. Además, el seno de un ángulo es igual al coseno del complemento, es decir,

$$\frac{\theta}{2} \approx \sin\frac{\theta}{2} = \cos\left(\frac{\pi}{2} - \frac{\theta}{2}\right).$$

Como $(\pi - \theta)/2$ es el ángulo entre $|x_0\rangle$ y $|\mathrm{d}\rangle$, por definición del producto interior, $\cos(\pi - \theta)/2$ es el producto interior de los vectores $|x_0\rangle$ y $|\mathrm{d}\rangle$, cuyo resultado es

$$\frac{\theta}{2} \approx \sin\frac{\theta}{2} = \cos\left(\frac{\pi}{2} - \frac{\theta}{2}\right) = \langle x_0 | \mathrm{d}\rangle = \frac{1}{\sqrt{N}}.$$

Por lo tanto,

$$\theta \approx \frac{2}{\sqrt{N}}.$$

El primer paso del Algoritmo 9.2 es la preparación del estado inicial $|\mathrm{d}\rangle$. El siguiente paso es aplicar u_f a $|\mathrm{d}\rangle$. La acción de u_f sobre $|\mathrm{d}\rangle$ (escrito en la base computacional) invierte el signo de la amplitud de $|x_0\rangle$ y no cambia las otras amplitudes. La amplitud de $|x_0\rangle$ es la proyección ortogonal de $|\mathrm{d}\rangle$ en el eje vertical; ver la Fig. 9.1, que se invierte por la acción de u_f. Geométricamente, la acción de u_f está representada por un reflexión de $|\mathrm{d}\rangle$ sobre el eje horizontal. El ángulo entre los vectores $|\mathrm{d}\rangle$ y $(u_f|\mathrm{d}\rangle)$ es θ, como se muestra en la Fig. 9.2.

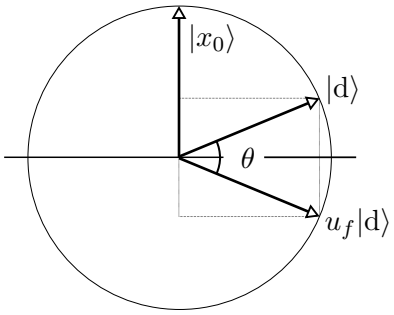


Figura 9.2: Vector $u_f|\mathrm{d}\rangle$ is a reflexion of $|\mathrm{d}\rangle$ about the horizontal axis.

El siguiente paso es aplicar $g = \big(2\,|\mathrm{d}\rangle\langle\mathrm{d}| - I_N\big)$. Demostremos que la acción de g es una reflexión sobre el eje definido por $|\mathrm{d}\rangle$. Esta demostración se realiza en dos partes. Primero, mostramos que $|\mathrm{d}\rangle$ es invariante bajo la acción de g. En segundo lugar, mostramos que la acción de g sobre $\big|\mathrm{d}^\perp\big\rangle$ invierte el signo de $\big|\mathrm{d}^\perp\big\rangle$, donde $\big|\mathrm{d}^\perp\big\rangle$ es cualquier vector ortogonal a $|\mathrm{d}\rangle$. El primer paso sigue de

$$g|\mathrm{d}\rangle = \big(2\,|\mathrm{d}\rangle\langle\mathrm{d}| - I_N\big)|\mathrm{d}\rangle = 2\,|\mathrm{d}\rangle\,\big\langle\mathrm{d}\big|\mathrm{d}\big\rangle - |\mathrm{d}\rangle = |\mathrm{d}\rangle,$$

porque $\big\langle\mathrm{d}\big|\mathrm{d}\big\rangle = 1$. El segundo paso se sigue de

$$g\Big|\mathrm{d}^\perp\Big\rangle = \big(2\,|\mathrm{d}\rangle\langle\mathrm{d}| - I_N\big)\Big|\mathrm{d}^\perp\Big\rangle = 2\,|\mathrm{d}\rangle\,\Big\langle\mathrm{d}\Big|\mathrm{d}^\perp\Big\rangle - \Big|\mathrm{d}^\perp\Big\rangle = -\Big|\mathrm{d}^\perp\Big\rangle,$$

porque $\big\langle\mathrm{d}\big|\mathrm{d}^\perp\big\rangle = 0$.

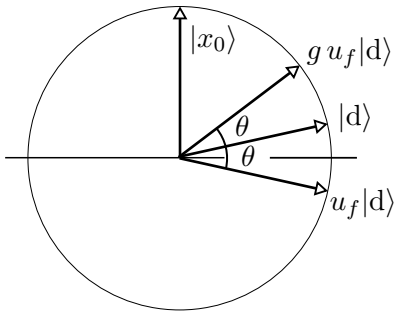


Figura 9.3: El Vector $g\,u_f|\mathrm{d}\rangle$ es un reflexión de $u_f|\mathrm{d}\rangle$ sobre $|\mathrm{d}\rangle$.

La Fig. 9.3 representa a $g\,u_f|\mathrm{d}\rangle$ y muestra que la acción de gu_f gira el estado inicial θ grados hacia $|x_0\rangle$. Dado que θ es un ángulo pequeño, este logro es modesto, pero prometedor. Es fácil ver que la segunda acción de gu_f repite el proceso de rotar θ grados hacia $|x_0\rangle$. Queremos saber cuántas iteraciones r se necesitan como $r\theta = \pi/2$. El número de iteraciones es

$$r = \left\lfloor \frac{\pi}{2\theta} \right\rfloor = \left\lfloor \frac{\pi}{4}\sqrt{N} \right\rfloor.$$

Aún falta el cálculo de la probabilidad de éxito. Después de r iteraciones, el estado de los qubits es

$$|\psi\rangle = (g\,u_f)^{\left\lfloor \frac{\pi}{4}\sqrt{N} \right\rfloor}|\mathrm{d}\rangle.$$

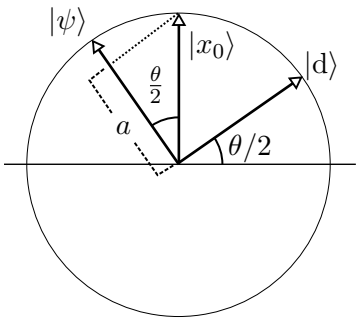


Figura 9.4: El Vector $|\psi\rangle$ es el estado final antes de la medición y a es la proyección de $|x_0\rangle$ on $|\psi\rangle$. The angle between $|\psi\rangle$ y $|x_0\rangle$ es menor o igual que $\theta/2$.

El vector $|\psi\rangle$ es casi ortogonal a $|\mathrm{d}\rangle$ en este punto, como se muestra en la Fig. 9.4. El ángulo entre $|\psi\rangle$ y $|x_0\rangle$ es menor o igual que $\theta/2$. La probabilidad de éxito es mayor o igual al cuadrado absoluto de la amplitud de $|x_0\rangle$ en la descomposición de $|\psi\rangle$ en la base computacional. Esta amplitud es a como se muestra en la Fig. 9.4. La proyección ortogonal de $|x_0\rangle$ sobre el estado final es como máximo $\cos(\theta/2)$. Por lo tanto, la probabilidad de éxito $p = |a|^2$ satisface

$$p \;\geq\; \cos^2\frac{\theta}{2} \;\geq\; 1 - \sin^2\frac{\theta}{2} \;\geq\; 1 - \frac{1}{N}.$$

El caso $N = 4$ es especial porque $\theta = 60°$, desde que $\sin(\theta/2) = 1/\sqrt{N}$. Con una aplicación de gu_f, el vector $|\mathrm{d}\rangle$ gira $60°$ y coincide con $|x_0\rangle$. En este caso, la probabilidad de éxito es exactamente $p = 1$.

9.7. Observaciones finales

La misma técnica para implementar el oráculo usando n qubits analizada en este Capítulo se puede aplicar a la implementación del algoritmo Deutsch-Jozsa con solo n qubits. Tenga en cuenta que el estado del segundo registrador en el circuito de Deutsch-Jozsa antes de aplicar U_f es $|-\rangle$. Lo que tenemos que hacer es descartar el segundo registrador y reemplazar U_f en el circuito de Deutsch-Jozsa por u_f descrito al final de la Sec. 9.5.

Capítulo 10

Estimación de fase y aplicaciones

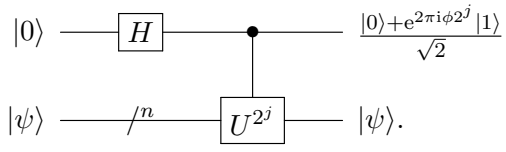
Kitaev publicó el algoritmo de estimación de fase como un preprint en el año 1995 [31], y más tarde como una sección de un libro en ruso, que se tradujo al inglés [32]. El método de Kitaev se basa en un procedimiento para medir un valor propio de un operador unitario, es decir, dado un operador unitario U y uno de sus vectores propios $|\psi\rangle$, el algoritmo encuentra el valor propio $\exp(2\pi\phi)$, de modo que $U|\psi\rangle = \exp(2\pi\phi)|\psi\rangle$, donde ϕ es la fase del valor propio. Este algoritmo proporciona una forma alternativa de factorizar números enteros y calcular logaritmos discretos. No sólo eso, se usa en muchas aplicaciones, como el conteo cuántico. Este algoritmo ha sido descrito en muchos libros [4, 6, 27, 30, 60].

10.1. Algoritmo de estimación de fase

Supongamos que tenemos un operador unitario de n-qubits U y conocemos uno de sus vectores propios $|\psi\rangle$. No conocemos el valor propio asociado con $|\psi\rangle$, pero sabemos que su expresión analítica es $e^{2\pi i\phi}$, donde $0 \leq \phi < 1$ (se desconoce ϕ), porque U es unitario. Suponemos por ahora que $\phi = 0.\phi_1 \cdots \phi_m$ para algún entero m, donde ϕ_1, ..., ϕ_m son bits, es decir, la fase del valor propio $e^{2\pi i\phi}$ es un múltiplo racional de 2π. El objetivo del algoritmo de estimación de fase es determinar ϕ utilizando U como oráculo y $|\psi\rangle$ como entrada.

Bloque básico de estimación de fase

El bloque básico del circuito depende de un número entero $0 \leq j < m$ y viene dado por



Para verificar la corrección de la salida del bloque básico, tenemos que usar

$$U|\psi\rangle = e^{2\pi i\phi}|\psi\rangle,$$

y también

$$U^{2^j}|\psi\rangle = e^{2\pi i\phi\, 2^j}|\psi\rangle.$$

La salida del bloque básico se obtiene aplicando el operador controlado U^{2^j} sobre $(H|0\rangle)\otimes|\psi\rangle$, es decir,

$$C\big(U^{2^j}\big)\left(\frac{|0\rangle|\psi\rangle+|1\rangle|\psi\rangle}{\sqrt{2}}\right) \;=\; \frac{|0\rangle|\psi\rangle+|1\rangle U^{2^j}|\psi\rangle}{\sqrt{2}} \;=\; \frac{|0\rangle+\mathrm{e}^{2\pi\mathrm{i}\phi 2^j}|1\rangle}{\sqrt{2}}\otimes|\psi\rangle.$$

Aquí vemos un ejemplo del *proceso de retroceder la fase* porque la fase fue producida por la acción de U en el segundo registrador pero aparece como una fase relativa del primer qubit después de que $|\psi\rangle$ ha sido recolectado.

Existe una forma alternativa de escribir el valor propio de U^{2^j} asociado con $|\psi\rangle$. Usando ese $\phi = 0.\phi_1\cdots\phi_m$ en binario, entonces

$$\phi \;=\; \frac{\phi_1}{2}+\frac{\phi_2}{2^2}+\cdots+\frac{\phi_m}{2^m}.$$

Multiplicando por 2^j, obtenemos

$$\phi\, 2^j \;=\; 2^{j-1}\phi_1+\cdots+2\phi_{j-1}+\phi_j+\frac{\phi_{j+1}}{2}+\cdots+\frac{\phi_m}{2^{m-j}}.$$

Es sencillo comprobar que

$$\exp\left(2\pi\mathrm{i}\phi\, 2^j\right) \;=\; \exp\left(2\pi\mathrm{i}\left(\frac{\phi_{j+1}}{2}+\cdots+\frac{\phi_m}{2^{m-j}}\right)\right)$$

porque $\exp\left(2\pi\mathrm{i}\, 2^{j-1}\phi_1\right)=\cdots=\exp\left(2\pi\mathrm{i}\phi_j\right)=1$. Entonces,

$$\exp\left(2\pi\mathrm{i}\phi\, 2^j\right) \;=\; \exp\left(2\pi\mathrm{i}\, 0.\phi_{j+1}\cdots\phi_m\right).$$

Tenga en cuenta que se eliminaron los primeros dígitos de ϕ.

La implementación de U^{2^j} no necesariamente la realizan las aplicaciones 2^j de U. Este método es ineficaz si m es grande. La implementación depende de aplicaciones específicas del algoritmo de estimación de fase. Por ejemplo, si U realiza aritmética modular, se emplea el *método de exponenciación de cuadrados repetidos.*

Primer bloque

El primer bloque del circuito de estimación de fase se representa en la Fig. 10.1. El circuito tiene dos registradores: El primero tiene m qubits con entrada $|0\rangle^{\otimes m}$ y el segundo n qubits con entrada $|\psi\rangle$. La salida es una consecuencia directa de cada bloque básico, donde j va de 0 a $m-1$. El orden de las operaciones controladas es irrelevante, pero j debe ser 0 para el qubit m-ésimo, j debe ser 1 para el qubit $(m-1)$-ésimo, y así sucesivamente. La salida del primer registrador del primer bloque es

$$\frac{|0\rangle+\mathrm{e}^{2\pi\mathrm{i}\phi 2^{m-1}}|1\rangle}{\sqrt{2}}\otimes\frac{|0\rangle+\mathrm{e}^{2\pi\mathrm{i}\phi 2^{m-2}}|1\rangle}{\sqrt{2}}\otimes\cdots\otimes\frac{|0\rangle+\mathrm{e}^{2\pi\mathrm{i}\phi 2^{0}}|1\rangle}{\sqrt{2}}.$$

Esta salida se puede simplificar en una expresión muy clara. Para hacerlo, reemplacemos cada término con un término equivalente usando una suma binaria y reuniendo los denominadores

$$\frac{1}{\sqrt{2^m}}\sum_{\ell_1=0}^{1}\mathrm{e}^{2\pi\mathrm{i}\phi 2^{m-1}\ell_1}|\ell_1\rangle\otimes\sum_{\ell_2=0}^{1}\mathrm{e}^{2\pi\mathrm{i}\phi 2^{m-2}\ell_2}|\ell_2\rangle\otimes\cdots\otimes\sum_{\ell_1=0}^{m}\mathrm{e}^{2\pi\mathrm{i}\phi 2^{0}\ell_m}|\ell_m\rangle.$$

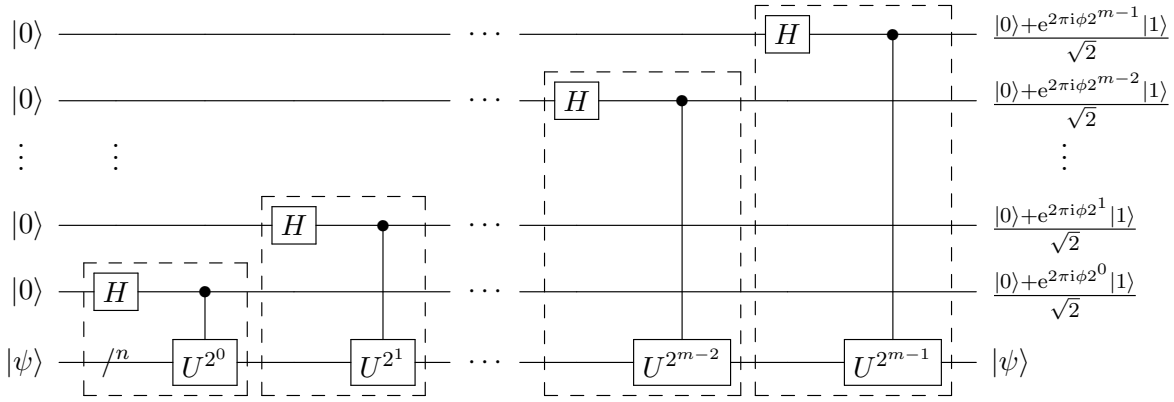


Figura 10.1: El primer bloque del circuito de estimación de fase está formado por m bloques básicos.

Llevando todas las sumas al principio de la expresión y combinando todas las exponenciales, obtenemos

$$\frac{1}{\sqrt{2^m}} \sum_{\ell_1,\ldots,\ell_m=0}^{1} \mathrm{e}^{2\pi \mathrm{i}\phi(2^{m-1}\ell_1+\cdots+2^0\ell_m)}|\ell_1\rangle \otimes \ldots \otimes |\ell_m\rangle.$$

Convirtiendo el binario en decimal, obtenemos

$$\frac{1}{\sqrt{2^m}} \sum_{\ell=0}^{2^m-1} \mathrm{e}^{2\pi \mathrm{i}\,\phi\,\ell}|\ell\rangle.$$

Esta es la expresión ordenada que estábamos buscando. Resumamos el primer bloque:

$$|0\rangle^{\otimes m} \otimes |\psi\rangle \xrightarrow[\text{bloque}]{\text{primer}} \left(\frac{1}{\sqrt{2^m}} \sum_{\ell=0}^{2^m-1} \mathrm{e}^{2\pi \mathrm{i}\,\phi\,\ell}|\ell\rangle \right) \otimes |\psi\rangle,$$

donde m es el número de qubits del primer registrador, $|\psi\rangle$ es un vector propio de U con valor propio $\exp(2\pi \mathrm{i}\,\phi)$ y $\phi = 0.\phi_1 \ldots \phi_m$. En la siguiente subsección, mostramos que la salida del primer registrador es

$$F_{2^m}|\phi_1, \ldots, \phi_m\rangle,$$

donde F_{2^m} es la transformada de Fourier, definida en la Sec. 7.4.

Circuito completo de estimación de fase

En la última Subsección, hemos mostrado que la salida del primer registrador del primer bloque es

$$\frac{1}{\sqrt{2^m}} \sum_{\ell=0}^{2^m-1} \mathrm{e}^{2\pi \mathrm{i}\,\phi\,\ell}|\ell\rangle.$$

Por otro lado, la acción de la transformada de Fourier F_{2^m} sobre un estado genérico $|j\rangle$ de la base computacional es

$$F_{2^m}|j\rangle \;=\; \frac{1}{\sqrt{2^m}} \sum_{\ell=0}^{2^m-1} \mathrm{e}^{\frac{2\pi \mathrm{i} j\ell}{2^m}}|\ell\rangle.$$

Invirtiendo la ecuación, tomando $j = (\phi_1 ... \phi_m)_2 = (2^m \phi)_{10}$ y $|j\rangle = |\phi_1\rangle \otimes ... \otimes |\phi_m\rangle$, obtenemos

$$F_{2^m}^{\dagger} \left(\frac{1}{\sqrt{2^m}} \sum_{\ell=0}^{2^m-1} \mathrm{e}^{2\pi \mathrm{i}\, \phi\, \ell} |\ell\rangle \right) \; = \; |2^m \phi\rangle \; = \; |\phi_1\rangle \otimes \cdots \otimes |\phi_m\rangle.$$

Hemos demostrado que si aplicamos la transformada inversa de Fourier a la salida del primer bloque, el resultado es un estado $|j\rangle$ de la base computacional igual a $|\phi_1\rangle \otimes ... \otimes |\phi_m\rangle$. Esto significa que una medición en la base computacional revela con certeza cada bit fraccionario de ϕ porque estamos asumiendo que ϕ se ha dado con m bits. En el caso general, el resultado del algoritmo es una buena estimación de m-bits de ϕ, que denotamos por $\tilde{\phi}$. El circuito completo del algoritmo de estimación de fase se muestra en la Fig. 10.2. El algoritmo se describe en el Algoritmo 10.1.

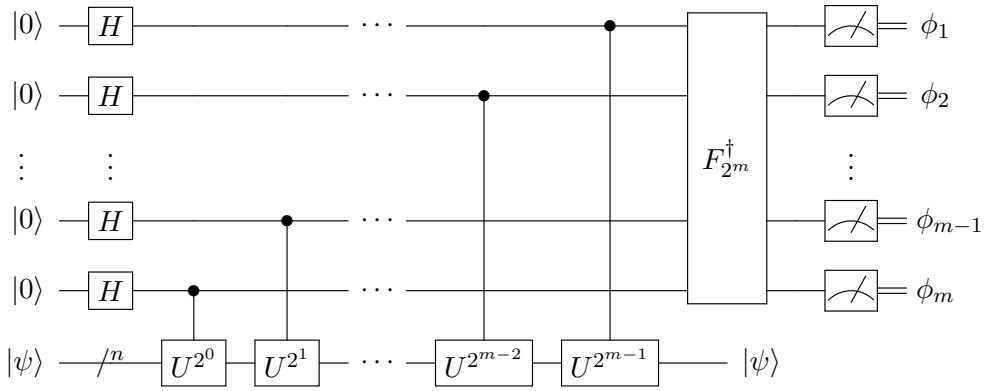


Figura 10.2: Circuito completo del algoritmo de estimación de fase.

Algoritmo 10.1: Algoritmo de estimación de fase

Entrada: Vector propio $|\psi\rangle$ de U.
Salida: Número $2^m \phi$, donde $\exp(2\pi \mathrm{i} \phi)$ es el valor propio de $|\psi\rangle$.

1 Prepare el estado inicial $|0\rangle^{\otimes m} \otimes |\psi\rangle$;
2 Aplicar $H^{\otimes m}$ al primer registrador;
3 Para ℓ en el intervalo $[0, m-1]$ aplique la operación controlada $C^{m-\ell}\left(U^{2^\ell}\right)$, donde el qubit de control es $m - \ell$ y el objetivo es el segundo registrador ;
4 Aplicar $F_{2^m}^{\dagger}$ al primer registrador ;
5 Medir el primer registrador en la base computacional.

10.2. Aplicación para encontrar encontrar orden

Sean N y x enteros positivos de modo que $1 < x < N$ y el $\mathrm{mcd}(x, N) = 1$. El orden de x módulo N es el entero positivo más pequeño r que obedece

$$x^r \equiv 1 \mod N.$$

Dados x y N, la búsqueda de oden es el problema de calcular r. En esta sección, mostramos cómo resolver el problema de búsqueda de orden de manera eficiente utilizando el algoritmo de estimación de fase que proporciona una alternativa al algoritmo de factorización de Shor.

La estrategia es reemplazar $|\psi\rangle$ en el Algoritmo 10.1 por $|1\rangle$ (el segundo vector de la base computacional del segundo registrador) y elegir U como el operador unitario que multiplica la entrada por x módulo N, es decir,

$$U|y\rangle \;=\; |xy \mod N\rangle.$$

La entrada es un vector $|y\rangle$ de la base computacional del segundo registrador. Podemos pensar que y está representado en el sistema decimal. La salida es también un vector $|y'\rangle$ de la base del cálculo del segundo registrador, que se obtiene calculando $xy \equiv y'$ módulo N. U es un operador unitario porque $\text{mcd}(x, N) = 1$. $U^\dagger$ se define en consecuencia usando x^{-1} módulo N, es decir,

$$U^\dagger|y\rangle \;=\; \left|x^{-1}y \mod N\right\rangle.$$

La motivación de usar U aquí es que la aplicación repetida de U produce potencias sucesivas de x, de hecho, $U^j|y\rangle = \left|x^j y\right\rangle$. El número de n qubits del segundo registrador debe poder alojar U, luego tomamos $n = \lceil \log_2 N \rceil$.

Para entender la búsqueda de orden como un algoritmo de estimación de fase, encontramos los vectores propios de U. Es sencillo obtener 1-vector propio, porque el conjunto $\{x^0, x^1, ...x^{r-1}\}$, donde r es el orden de x módulo N, es invariante bajo la multiplicación por x. Entonces, el vector normalizado

$$|\psi_0\rangle = \frac{1}{\sqrt{r}} \sum_{\ell=0}^{r-1} \left|x^\ell\right\rangle$$

es un vector propio de U. Dado que este es el primer vector de la base de Fourier (F_r) cuando consideramos la transformación del subconjunto $\left\{\left|x^0\right\rangle, \left|x^1\right\rangle, ...\left|x^{r-1}\right\rangle\right\}$ de la base computacional, definamos los restantes:

$$|\psi_k\rangle \;=\; \frac{1}{\sqrt{r}} \sum_{\ell=0}^{r-1} \mathrm{e}^{-\frac{2\pi i k\ell}{r}} \left|x^\ell\right\rangle. \tag{10.1}$$

Ahora comprobemos que cada $|\psi_k\rangle$ es un vector propio de U. En realidad,

$$\begin{aligned} U|\psi_k\rangle &= \frac{1}{\sqrt{r}} \sum_{\ell=0}^{r-1} \mathrm{e}^{-\frac{2\pi i k\ell}{r}} \left|x^{\ell+1}\right\rangle \\ &= \frac{1}{\sqrt{r}} \sum_{\ell=0}^{r-1} \mathrm{e}^{-\frac{2\pi i k(\ell-1)}{r}} \left|x^\ell\right\rangle \\ &= \mathrm{e}^{\frac{2\pi i k}{r}} |\psi_k\rangle. \end{aligned}$$

Concluimos que $|\psi_k\rangle$ es un vector propio de U con valor propio $\exp(2\pi i k/r)$ para $0 \le k < r$. Si somos capaces de preparar la entrada al segundo registrador del algoritmo de estimación de fase como $|\psi_k\rangle$ para algún k, obtendremos como salida una aproximación de k/r. Si la entrada al segundo registrador es $|\psi_k\rangle$, la salida del primer bloque será

$$|0\rangle^{\otimes m}|\psi_k\rangle \xrightarrow[\text{bloque}]{\text{primer}} \frac{1}{\sqrt{2^m}} \sum_{\ell=0}^{2^m-1} \mathrm{e}^{\frac{2\pi i k\ell}{r}} |\ell\rangle|\psi_k\rangle. \tag{10.2}$$

No podemos preparar $|\psi_k\rangle$ como una entrada para el algoritmo de estimación de fase, pero podemos encontrar un vector conocido que esté abarcado por el conjunto de vectores $\{|\psi_0\rangle, ...|\psi_r\rangle\}$. Usando la transformada inversa de Fourier, tenemos

$$\left|x^\ell\right\rangle \;=\; \frac{1}{\sqrt{r}}\sum_{k=0}^{r-1} \mathrm{e}^{\frac{2\pi i k\ell}{r}}|\psi_k\rangle.$$

El candidato más simple es $\left|x^0\right\rangle = |1\rangle$, que viene dado por

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

Usando la transformación (10.2) para cada k, la salida del primer bloque es

$$|0\rangle^{\otimes m}|1\rangle \xrightarrow[\text{bloque}]{\text{primer}} \frac{1}{\sqrt{r2^m}}\sum_{k=0}^{r-1}\sum_{\ell=0}^{2^m-1} \mathrm{e}^{\frac{2\pi i k\ell}{r}}|\ell\rangle|\psi_k\rangle.$$

Para simplificar la salida, usamos la Ec. (10.1)

$$\text{salida} \;\;=\;\; \frac{1}{\sqrt{r2^m}}\sum_{k=0}^{r-1}\sum_{\ell=0}^{2^m-1} \mathrm{e}^{\frac{2\pi i k\ell}{r}}|\ell\rangle\left(\frac{1}{\sqrt{r}}\sum_{\ell'=0}^{r-1}\mathrm{e}^{-\frac{2\pi i k\ell'}{r}}\left|x^{\ell'}\right\rangle\right).$$

Invirtiendo el orden de las sumas y combinando los exponentes, obtenemos

$$\text{salida} \;\;=\;\; \frac{1}{\sqrt{2^m}}\sum_{\ell=0}^{2^m-1}\sum_{\ell'=0}^{r-1}\left(\frac{1}{r}\sum_{k=0}^{r-1}\mathrm{e}^{\frac{2\pi i k(\ell-\ell')}{r}}\right)|\ell\rangle\left|x^{\ell'}\right\rangle.$$

La expresión entre paréntesis es 1 si $\ell = \ell'$ y 0 en caso contrario. Esto significa que la salida del primer bloque es

$$|0\rangle^{\otimes m}|1\rangle \xrightarrow[\text{bloque}]{\text{primer}} \frac{1}{\sqrt{2^m}}\sum_{\ell=0}^{2^m-1}|\ell\rangle\left|x^\ell\right\rangle.$$

Este es el mismo estado que en el algoritmo de factorización de Shor estándar justo antes de aplicar la transformada inversa de Fourier F_{2^m}, es decir, consideramos la parte del algoritmo donde la entrada es $|0\rangle^{\otimes m}|0\rangle$ y luego se aplica la puerta de Hadamard en cada qubit del primer registrador y luego U_f, donde $f(x) = a^x \mod N$:

$$|0\rangle^{\otimes m}|0\rangle \xrightarrow{U_f\cdot(H^{\otimes m}\otimes I)} \frac{1}{\sqrt{2^m}}\sum_{\ell=0}^{2^m-1}|\ell\rangle\left|x^\ell\right\rangle.$$

Esto significa que la versión de estimación de fase arroja el mismo resultado y el análisis de la probabilidad de éxito es exactamente igual que en el algoritmo de factorización de Shor si elegimos m de modo que $m \approx 2\lceil\log_2 N\rceil$. El número de qubits del primer registrador debe estar cerca del doble del número de qubits del segundo registrador.

Hay un caso especial interesante. Si sabemos de alguna manera que el orden r es una potencia de 2, podemos tomar $m = n = \lceil\log_2 N\rceil$. Si r es una potencia de 2, la fase del valor propio

$\exp(2\pi ik/r)$ es un múltiplo racional de 2π y el algoritmo de estimación de fase devuelve un valor exacto $2^n k/r$ cuando $m = n$.

¿Cómo implementamos U^{2^j} de manera eficiente para $0 \leq j < m$? Tenga en cuenta que

$$U^{2^j}|y\rangle \;=\; \Big|x^{2^j} y \mod N\Big\rangle.$$

Dado que x^{2^j} se puede calcular de manera eficiente en $O(n^2)$ pasos utilizando el método de exponenciación por cuadrados repetidos, en lugar de aplicar U repetidamente 2^j veces, para cada j implementamos un operador $U_j|y\rangle = |zy\rangle$ después de calcular $z = x^{2^j}$ utilizando el método de exponenciación por cuadrados repetidos. En este caso, el primer bloque se puede calcular en $O(n^3)$ pasos.

10.3. Aplicación al logaritmo discreto

Sea N, a y b enteros positivos conocidos y sea s un entero positivo tal que $a^s \equiv b \mod N$ y el $\mathrm{mcd}(a, N) = 1$. Nuestro objetivo es encontrar s dados N, a y b como entrada. Este es el mismo problema abordado en la Sec. 8.3 de la Página 78. Ahora mostramos como resolver el problema del logaritmo discreto utilizando el algoritmo de estimación de fase que proporciona una versión alternativa al algoritmo de Shor para el logaritmo discreto.

La estrategia que usamos aquí es la misma que se usa en el algoritmo de búsqueda de orden, basado en la estimación de fase. Recuerde que, cuando describimos el algoritmo de búsqueda de orden, la salida del primer bloque es la misma que en el algoritmo de factorización de Shor original justo antes de la acción de la transformada inversa de Fourier. Ahora, el estado de la computadora cuántica justo antes de la acción de $F_r^\dagger \otimes F_r^\dagger$ en el algoritmo del logaritmo discreto de Shor descrito en la Sec. 8.3 es

$$\frac{1}{r}\sum_{x,y=0}^{r-1} |x\rangle|y\rangle|a^x b^y \mod N\rangle.$$

Si deseamos producir este estado usando el algoritmo de estimación de fase, necesitamos usar tres registradores y dos operadores unitarios:

$$\begin{aligned} U_a|x\rangle &= |ax \mod N\rangle, \\ U_b|y\rangle &= |by \mod N\rangle. \end{aligned}$$

Donde U_b es un operador unitario porque $\mathrm{mcd}(b, N) = 1$. De hecho, el inverso de b es a^{r-s}, donde r es el orden de a módulo N. En el nuevo algoritmo, la acción de U_a está controlada por el primer registrador y la acción de U_b está controlada por el segundo registrador, como se describe en la Fig. 10.3. Tenga en cuenta que U_a y U_b actúan en el mismo registrador.

Verifiquemos que los vectores

$$|\psi_k\rangle \;=\; \frac{1}{\sqrt{r}}\sum_{\ell=0}^{r-1} \mathrm{e}^{-\frac{2\pi i k\ell}{r}}\Big|a^\ell\Big\rangle,$$

que son vectores propios de U_a, también son vectores propios de U_b. De hecho, usando ese $b = a^s$,

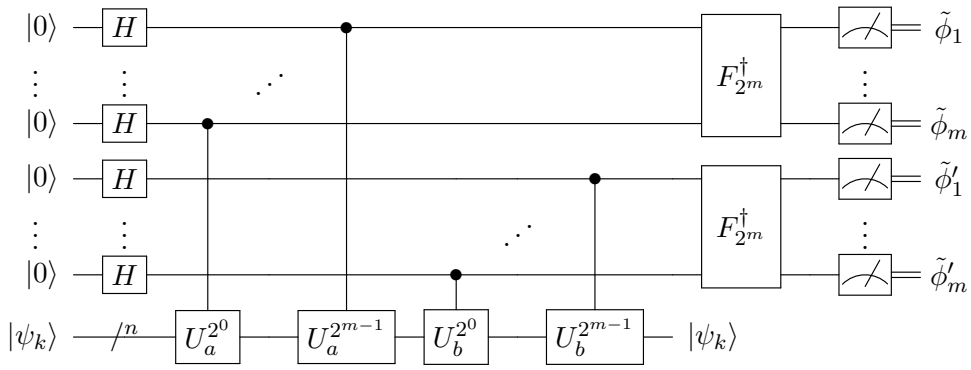


Figura 10.3: Circuito del algoritmo del logaritmo discreto mediante estimación de fase, donde $|\psi_k\rangle$ es un vector propio común de U_a y U_b. Dado que normalmente no somos capaces de preparar $|\psi_k\rangle$ como entrada al tercer registrador, usamos $|1\rangle$ en su lugar, que puede ser representado como una combinación lineal de $|\psi_k\rangle$, para $0 \leq k < r$.

tenemos

$$\begin{aligned} U_b|\psi_k\rangle &= \frac{1}{\sqrt{r}} \sum_{\ell=0}^{r-1} \mathrm{e}^{-\frac{2\pi \mathrm{i} k\ell}{r}} \left|a^{\ell+s}\right\rangle \\ &= \frac{1}{\sqrt{r}} \sum_{\ell=0}^{r-1} \mathrm{e}^{-\frac{2\pi \mathrm{i} k(\ell-s)}{r}} \left|a^{\ell}\right\rangle \\ &= \mathrm{e}^{\frac{2\pi \mathrm{i} ks}{r}} |\psi_k\rangle. \end{aligned}$$

Concluimos que $|\psi_k\rangle$ es un vector propio de U_b con valor propio $\exp(2\pi \mathrm{i} ks/r)$ para $0 \leq k < r$. Mejor aún, $|\psi_k\rangle$ es un vector propio de U_a y U_b simultáneamente. Si podemos preparar la entrada al tercer registrador del circuito representado en la Fig. 10.3 como $|\psi_k\rangle$ para algunos k, obtendremos una estimación de $\tilde{\phi} \approx k/r$ como salida del primer registrador y una estimación de $\tilde{\phi}' \approx ks/r$ como salida del segundo registrador módulo N. Si la entrada al tercer registrador es $|\psi_k\rangle$, la salida del primer bloque será

$$|0\rangle^{\otimes m}|0\rangle^{\otimes m}|\psi_k\rangle \xrightarrow[\text{bloque}]{\text{primer}} \left(\frac{1}{\sqrt{2^m}} \sum_{\ell=0}^{2^m-1} \mathrm{e}^{\frac{2\pi \mathrm{i} k\ell}{r}} |\ell\rangle\right) \left(\frac{1}{\sqrt{2^m}} \sum_{\ell'=0}^{2^m-1} \mathrm{e}^{\frac{2\pi \mathrm{i} ks\ell'}{r}} \left|\ell'\right\rangle\right) |\psi_k\rangle,$$

donde m es el número de qubits del primer y segundo registrador, y $n = \lceil \log_2 N \rceil$ del tercer registrador. El primer término entre paréntesis se obtiene reemplazando ϕ por k/r y el segundo término reemplazando ϕ por ks/r en la salida del primer bloque del algoritmo de estimación de fase. Simplificando la salida, tenemos

$$|0\rangle^{\otimes m}|0\rangle^{\otimes m}|\psi_k\rangle \xrightarrow[\text{bloque}]{\text{primer}} \left(\frac{1}{2^m} \sum_{\ell,\ell'=0}^{2^m-1} \mathrm{e}^{\frac{2\pi \mathrm{i} k(\ell+s\ell')}{r}} |\ell\rangle\left|\ell'\right\rangle\right) |\psi_k\rangle.$$

Por lo general, no podemos preparar $|\psi_k\rangle$ como entrada para el algoritmo de estimación de fase,

pero podemos usarla nuevamente

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

En este caso, la entrada y la salida del primer bloque son

$$|0\rangle^{\otimes m}|0\rangle^{\otimes m}|1\rangle \xrightarrow[\text{bloque}]{\text{primer}} \frac{1}{\sqrt{r}}\sum_{k=0}^{r-1}\left(\frac{1}{2^m}\sum_{\ell,\ell'=0}^{2^m-1}\mathrm{e}^{\frac{2\pi ik(\ell+s\ell')}{r}}|\ell\rangle\big|\ell'\big\rangle\right)|\psi_k\rangle.$$

Usamos la definición de $|\psi_k\rangle$ dada por la Ec. (10.1) para simplificar la salida. empezamos a escribir

$$\frac{1}{\sqrt{r}}\sum_{k=0}^{r-1}\left(\frac{1}{2^m}\sum_{\ell,\ell'=0}^{2^m-1}\mathrm{e}^{\frac{2\pi ik(\ell+s\ell')}{r}}|\ell\rangle\big|\ell'\big\rangle\right)\left(\frac{1}{\sqrt{r}}\sum_{k'=0}^{r-1}\mathrm{e}^{-\frac{2\pi ikk'}{r}}\Big|a^{k'}\Big\rangle\right),$$

y colocando las sumas sobre ℓ, ℓ', k' hacia la izquierda y combinando los exponentes, obtenemos

$$\frac{1}{2^m}\sum_{\ell,\ell'=0}^{2^m-1}\sum_{k'=0}^{r-1}\left(\frac{1}{r}\sum_{k=0}^{r-1}\mathrm{e}^{\frac{2\pi ik(\ell+s\ell'-k')}{r}}\right)|\ell\rangle\big|\ell'\big\rangle\Big|a^{k'}\Big\rangle.$$

La expresión entre paréntesis es 1 si $\ell + s\ell' = k'$ y 0 en caso contrario. Esto significa que la acción del primer bloque es

$$|0\rangle^{\otimes m}|0\rangle^{\otimes m}|1\rangle \xrightarrow[\text{bloque}]{\text{primer}} \frac{1}{2^m}\sum_{\ell,\ell'=0}^{2^m-1}|\ell\rangle\big|\ell'\big\rangle\Big|a^{\ell+s\ell'}\Big\rangle.$$

Usando ese $a^s = b \mod N$, $\ell \to x$ y $\ell' \to y$, tenemos

$$|0\rangle^{\otimes m}|0\rangle^{\otimes m}|1\rangle \xrightarrow[\text{bloque}]{\text{primer}} \frac{1}{2^m}\sum_{x,y=0}^{2^m-1}|x\rangle|y\rangle|a^x b^y\rangle.$$

La salida del primer bloque tiene el mismo estado que en el algoritmo del logaritmo discreto de Shor estándar justo antes de aplicar las transformadas inversas de Fourier $F_{2^m}\otimes F_{2^m}$ (estado $|\psi_2\rangle$ del Algoritmo 8.1 de la Página 79). Esto significa que la versión de estimación de fase arroja el mismo resultado y el análisis de la probabilidad de éxito es exactamente igual que en el algoritmo de Shor si elegimos m adecuadamente.

10.4. Aplicación al conteo cuántico

En el contexto del algoritmo de Grover, tenemos un oráculo $f : \{0, ..., N-1\} \to \{0, 1\}$ que es una función booleana definida como

$$f(x) = \begin{cases} 1, & \text{si } x \in M, \\ 0, & \text{caso contrario,} \end{cases}$$

donde M es un subconjunto del dominio y $N = 2^n$. Decimos que x está marcado si $x \in M$. El número óptimo de pasos del algoritmo de Grover depende de $|M|$; de hecho, está dada por

$\frac{\pi}{4}\sqrt{N/|M|}$. Si se desconoce la cardinalidad de M, es posible encontrar un elemento marcado adivinando repetidamente el tiempo de ejecución del algoritmo de Grover [10]. Un método alternativo es resolver el problema de conteo cuántico [11].

El problema de conteo cuántico pregunta cuál es la cardinalidad de M dada la función f como un oráculo. Una solución clásica no puede funcionar mejor que las $\Omega(N)$ consultas al oráculo porque se deben verificar todos los elementos del dominio. El algoritmo cuántico puede encontrar la solución en $O\big(\sqrt{|M|\,N}\big)$ consultas al oráculo.

Antes de abordar el problema del conteo cuántico, repasemos algunos puntos clave del algoritmo de Grover. Existe una versión económica del algoritmo, que utiliza solo un registrador de n-qubits. El estado inicial es la superposición uniforme de la base computacional dada por

$$|\mathrm{d}\rangle \;=\; \frac{1}{\sqrt{N}}\sum_{j=0}^{N-1}|j\rangle,$$

y el algoritmo consta de $\frac{\pi}{4}\sqrt{N/|M|}$ aplicaciones del operador de evolución

$$U \;=\; G\,U_f,$$

donde

$$G \;=\; 2|\mathrm{d}\rangle\langle\mathrm{d}| - I$$

y

$$U_f \;=\; \sum_{x=0}^{N-1}(-1)^{f(x)}|x\rangle\langle x|.$$

El análisis del algoritmo se realiza utilizando los vectores propios $\mathrm{e}^{\pm\mathrm{i}\theta}$ de U, que se dan en términos de la superposición de estados marcados[1] $|M\rangle$ y la superposición de estados no marcados $\big|M^{\perp}\big\rangle$:

$$\big|\psi^{\pm}\big\rangle \;=\; \frac{|M\rangle \pm \mathrm{i}\,\big|M^{\perp}\big\rangle}{\sqrt{2}}, \tag{10.3}$$

donde

$$\sin\frac{\theta}{2} \;=\; \sqrt{\frac{|M|}{N}},$$

y

$$\begin{aligned} |M\rangle &= \frac{1}{\sqrt{|M|}}\sum_{x\in M}|x\rangle, \\ \Big|M^{\perp}\Big\rangle &= \frac{1}{\sqrt{N-|M|}}\sum_{x\notin M}|x\rangle. \end{aligned}$$

Es sencillo comprobar que $\big\langle M^{\perp}\big|M\big\rangle = 0$, y

$$|\mathrm{d}\rangle \;=\; \sqrt{\frac{|M|}{N}}\,|M\rangle + \sqrt{1-\frac{|M|}{N}}\,\Big|M^{\perp}\Big\rangle.$$

[1] Algunas referencias llaman a $|M\rangle$ como "buen estado" y $\big|M^{\perp}\big\rangle$ como "mal estado".

Usando la ecuación anterior, la definición de $\sin(\theta/2)$ y la Ec. (10.3), obtenemos

$$|\mathrm{d}\rangle \;=\; \frac{\mathrm{e}^{\frac{\mathrm{i}\theta}{2}}|\psi^+\rangle - \mathrm{e}^{-\frac{\mathrm{i}\theta}{2}}|\psi^-\rangle}{\mathrm{i}\sqrt{2}}.$$

Ahora volvemos al problema del conteo cuántico usando el algoritmo de estimación de fase. Dado que el valor propio de $|\psi^+\rangle$ es $\exp(\mathrm{i}\theta)$, donde $\sin(\theta/2) = \sqrt{|M|/N}$, obtendríamos una aproximación para $|M|$ si usamos $|\psi^+\rangle$ como entrada al segundo registrador del algoritmo de estimación de fase con $U = GU_f$. No sabemos como preparar $|\psi^+\rangle$, entonces la estrategia es reemplazar $|\psi\rangle$ en el Algoritmo 10.1 por un vector conocido que pertenece al subespacio abarcado por $|\psi^+\rangle$ y $|\psi^-\rangle$. El mejor candidato es $|\mathrm{d}\rangle$, que se puede preparar fácilmente aplicando $H^{\otimes n}$ a $|0\rangle^{\otimes n}$. En este caso, el número de qubits del segundo registrador debe ser $n = \log_2 N$ y la salida del primer bloque es

$$|0\rangle^{\otimes m}|\mathrm{d}\rangle \xrightarrow[\text{bloque}]{\text{primer}} \frac{\mathrm{e}^{\frac{\mathrm{i}\theta}{2}}}{\mathrm{i}\sqrt{2^{m+1}}}\sum_{\ell=0}^{2^m-1}\mathrm{e}^{2\pi\mathrm{i}\phi^+\ell}|\ell\rangle\big|\psi^+\big\rangle - \frac{\mathrm{e}^{-\frac{\mathrm{i}\theta}{2}}}{\mathrm{i}\sqrt{2^{m+1}}}\sum_{\ell=0}^{2^m-1}\mathrm{e}^{2\pi\mathrm{i}\phi^-\ell}|\ell\rangle\big|\psi^-\big\rangle,$$

donde $\phi^+ = \theta/2\pi$ para el primer término y $\phi^- = (2\pi-\theta)/2\pi$ para el segundo término. Después de aplicar la transformada inversa de Fourier $F^\dagger_{2^m}$, obtenemos la siguiente salida del circuito completo

$$\frac{\mathrm{e}^{\frac{\mathrm{i}\theta}{2}}}{\mathrm{i}\sqrt{2}}\left|2^m\tilde{\phi}^+\right\rangle - \frac{\mathrm{e}^{-\frac{\mathrm{i}\theta}{2}}}{\mathrm{i}\sqrt{2}}\left|2^m\tilde{\phi}^-\right\rangle,$$

donde $\tilde{\phi}$ es una estimación de m-bits de ϕ. Después de una medición en la base computacional, aprendemos una estimación de ϕ^+ o ϕ^- con la misma probabilidad. Sea $\tilde{\phi}$ el resultado de la medición. Usando $\sin(\theta/2) = \sqrt{|M|/N}$, $\phi^+ = \theta/2\pi$ y $\phi^- = (2\pi-\theta)/2\pi$, la estimación de $|M|$ es $N\sin^2(\pi\tilde{\phi})$ porque para el primer caso obtenemos una estimación de $|M|$ usando $|\widetilde{M}| = N\sin^2(\pi\tilde{\phi}^+)$ y para el segundo caso $|\widetilde{M}| = N\sin^2(\pi - \pi\tilde{\phi}^-) = N\sin^2(\pi\tilde{\phi}^-)$.

¿Cuántos qubits tiene el primer registrador? Esta es la parte difícil. Tenga en cuenta que m no puede ser igual a n porque el número de aplicaciones de U sería $2^0 + \cdots + 2^{n-1} = 2^n - 1$. Entonces, el número de consultas a f sería $O(N)$. Si elegimos $m = n/2$, el número de consultas a f sería $\sqrt{N}$, pero en este caso obtenemos una estimación $|\widetilde{M}|$ tal que

$$\big|\,|\widetilde{M}| - |M|\,\big| \;=\; O\big(\sqrt{|M|}\big). \tag{10.4}$$

Esta estimación no es buena. Por ejemplo, suponga que $|M|$ está alrededor de $N/2$. Si queremos saber el número de elementos marcados, y obtenemos $|\widetilde{M}|$ con un error tan grande como $O(\sqrt{N})$, no tenemos un buen resultado. Para entender cuál es el problema aquí, que no surge en los algoritmos de factorización y logaritmo discreto, tenemos que analizar cuidadosamente el rango de valores del ángulo θ que estamos tratando de estimar.

Para este análisis, supongamos que $0 < |M| \ll N$, o más formalmente, $|M| = O(\sqrt{N})$. La expresión $\sin(\theta/2) = \sqrt{|M|/N}$ se puede escribir asintóticamente como

$$\theta = \frac{2\sqrt{|M|}}{\sqrt{N}} + O\left(\frac{|M|}{N}\right).$$

Esto significa que $\theta/2\pi$ representado en términos de dígitos binarios tiene la forma $0{,}0\cdots01\cdots$, donde el número de 0 antes del primer 1 es alrededor de $n/2 - \log_2(|M|)/2$. Si elegimos el tamaño

del primer registrador para que m sea menor que $n/2-\log_2(|M|)/2$, es probable que obtengamos un 0 como salida del algoritmo de conteo, lo cual es incorrecto. Si elegimos $m = n/2$, obtendremos alrededor de $\log_2(|M|)/2$ bits significativos correctos de $\theta/2\pi$. Esta es una estimación imprecisa de $|M|$ compatible con la Ec. (10.4). De hecho, $|M|$ tiene $\log_2(|M|)$ bits, y necesitamos conocer la mayoría de los bits significativos para tener una buena estimación de $|M|$.

Capítulo 11

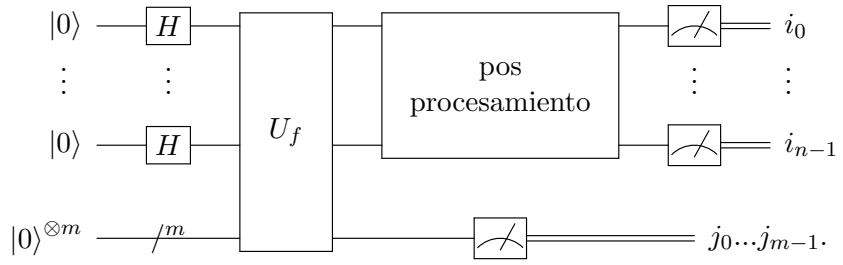
Observaciones finales

La mayoría de los algoritmos cuánticos analizados en este trabajo pueden integrarse en el marco basado en un Oráculo. La complejidad de la consulta de un algoritmo basado en un oráculo o una caja negra es el número de consultas. No importa lo difícil que sea implementar el oráculo a menos que apuntemos a resolver un problema práctico. En problemas prácticos, es nuestra tarea implementar el oráculo, y luego el costo de cada evaluación importa. Tome el algoritmo de factorización de Shor como ejemplo. El oráculo en este caso es una función r periódica y nuestro objetivo es encontrar r. Hemos visto que la función en el algoritmo de Shor es la exponenciación modular, que se puede implementar de manera eficiente en términos del tamaño de entrada utilizando el método de exponenciación por cuadrados repetidos.

Cualquier algoritmo determinístico clásico es una función con n-entradas y m-salidas; $f : \{0,1\}^n \longrightarrow \{0,1\}^m$, que es una colección de funciones booleanas de m y n-bits. Entonces, cualquier algoritmo clásico puede implementarse en una computadora cuántica con dos registradores de tamaños n y m usando el operador

$$U_f|x\rangle|y\rangle \;=\; |x\rangle|y \oplus f(x)\rangle.$$

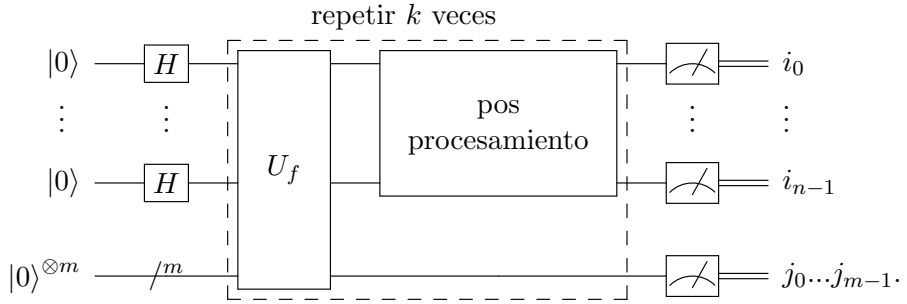
Para explotar el paralelismo cuántico, debemos aplicar $H^{\otimes n}$ al primer registrador antes de aplicar U_f. Después de aplicar U_f, tenemos un estado de superposición, que no es útil a menos que realicemos algún procesamiento pos-cuántico que produzca el resultado deseado. La mayoría de los algoritmos cuánticos que hemos analizado se pueden convertir en el siguiente circuito:



Para los algoritmos Deutsch-Jozsa, Bernstein-Vazirani, Simon y Shor (factorización), el pos-procesamiento cuántico es $H^{\otimes n}$ o la transformada inversa de Fourier. Tienen la estructura descrita anteriormente con algunas adaptaciones.

El algoritmo de Grover no tiene la estructura descrita anteriormente porque U_f y el procesamiento posterior se repiten muchas veces antes de la medición. Por otro lado, el algoritmo de

Grover tiene una ganancia polinomial en contraste con la ganancia exponencial de los algoritmos de Simon y Shor. La extensión de la estructura general que incluye el algoritmo de Grover es

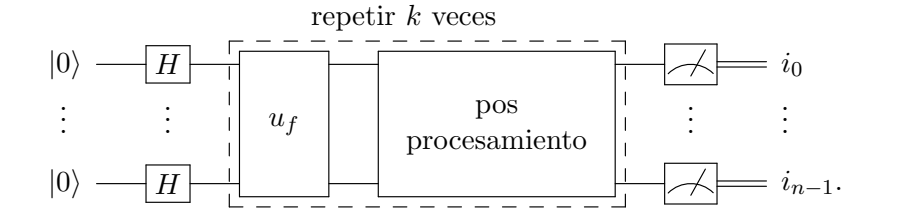


El número de repeticiones de k es 1 para Deutsch-Jozsa, Bernstein-Vazirani, Simon y Shor; y k es $\lfloor \pi\sqrt{2^n}/4 \rfloor$ para el algoritmo de Grover. La medición del segundo registrador es innecesaria. Está ahí porque ayuda en el análisis del algoritmo.

El segundo registrador de los algoritmos de Deutsch-Jozsa, Bernstein-Vazirani y Grover tiene un solo qubit ($m = 1$), cuyo estado durante el cómputo es $|-\rangle$, que se obtiene aplicando X y H sobre el último qubit antes de U_f. El oráculo para esos casos obedece

$$U_f|x\rangle|-\rangle \;=\; (-1)^{f(x)}|x\rangle|-\rangle.$$

Esto significa que se puede eliminar el segundo registrador y existe una versión económica del circuito con la siguiente forma:

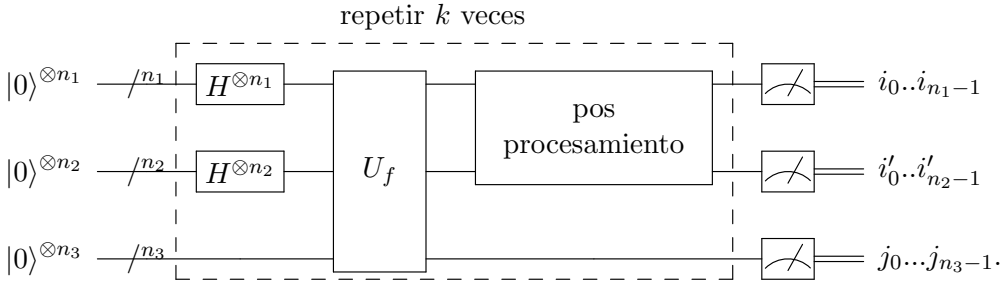


Como antes, tenemos $k = 1$ para los algoritmos Deutsch-Jozsa y Bernstein-Vazirani, $k = \lfloor \pi\sqrt{2^n}/4 \rfloor$ para el algoritmo de Grover y $u_f|x\rangle = (-1)^{f(x)}|x\rangle$.

El algoritmo de Shor para el logaritmo discreto nos muestra como extender la estructura del circuito cuando la función f tiene más de una variable. Supongamos que f tiene dos variables. Entonces, U_f se define como

$$U_f|x_1\rangle|x_2\rangle|y\rangle \;=\; |x_1\rangle|x_2\rangle|y \oplus f(x_1, x_2)\rangle.$$

Esto significa que necesitamos un circuito con tres registradores y la estructura general del algoritmo es la misma salvo pequeños cambios, como sigue:



Esas recetas son útiles para comprender algoritmos avanzados.

Bibliografía

[1] D. Aharonov. Quantum Computation. In *Annual Review of Computational Physics*, pages 259–346. World Scientific, vol. VI, 1999.

[2] S. Axler. *Linear Algebra Done Right*. Springer, New York, 1997.

[3] S. Barnett. *Quantum Information*. Oxford University Press, New York, 2009.

[4] G. Benenti, G. Casati, and G. Strini. *Principles of Quantum Computation and Information: Basic Tools and Special Topics*. World Scientific Publishing, River Edge, 2007.

[5] C. H. Bennett, E. Bernstein, G. Brassard, and U. V. Vazirani. Strengths and weaknesses of quantum computing. *SIAM J. Comput.*, 26(5):1510–1523, 1997.

[6] J. A. Bergou and M. Hillery. *Introduction to the Theory of Quantum Information Processing*. Springer, 2013.

[7] D. J. Bernstein. Detecting perfect powers in essentially linear time. *Math. Comput.*, 67(223):1253–1283, 1998.

[8] E. Bernstein and U. Vazirani. Quantum complexity theory. In *Proc. of the 25th Annual ACM Symposium on Theory of Computing*, STOC '93, page 11–20. ACM, New York, 1993.

[9] E. Bernstein and U. Vazirani. Quantum complexity theory. *SIAM Journal on Computing*, 26(5):1411–1473, 1997.

[10] M. Boyer, G. Brassard, P. Høyer, and A. Tapp. Tight bounds on quantum searching. *Forstschritte Der Physik*, 4:820–831, 1998.

[11] G. Brassard, P. Høyer, M. Mosca, and A. Tapp. Quantum amplitude amplification and estimation. *Quantum Computation and Quantum Information Science, AMS Contemporary Mathematics Series*, 305:53–74, 2002.

[12] G. Brassard, P. Høyer, and A. Tapp. Quantum cryptanalysis of hash and claw-free functions. In *Proc. 3rd Latin American Symposium LATIN'98*, pages 163–169. Springer, 1998.

[13] Guangya Cai and Daowen Qiu. Optimal separation in exact query complexities for Simon's problem. *Journal of Computer and System Sciences*, 97:83–93, 2018.

[14] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca. Quantum algorithms revisited. *Proc. Royal Society London Ser. A*, 454(1969):339–354, 1998.

[15] D. Coppersmith. An approximate Fourier transform useful in quantum factoring. *Arxiv:quant-ph/0201067*, 2002.

[16] A. DasGupta. The matching, birthday and the strong birthday problem: a contemporary review. *Journal of Statistical Planning and Inference*, 130(1):377–389, 2005.

[17] D. Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proc. Royal Society London Ser. A*, pages 96–117, 1985.

[18] D. Deutsch. Quantum computational networks. *Proc. Royal Society London Ser. A*, 425(1868):73–90, 1989.

[19] D. Deutsch and R. Jozsa. Rapid solution of problems by quantum computation. *Proc. Royal Society London Ser. A*, 439(1907):553–558, 1992.

[20] D. Dieks. Communication by EPR devices. *Physics Letters A*, 92(6):271 – 272, 1982.

[21] Jiangfeng Du, Mingjun Shi, Xianyi Zhou, Yangmei Fan, BangJiao Ye, Rongdian Han, and Jihui Wu. Implementation of a quantum algorithm to solve the Bernstein-Vazirani parity problem without entanglement on an ensemble quantum computer. *Phys. Rev. A*, 64:042306, 2001.

[22] M. Ekerå. On the success probability of quantum order finding. *Arxiv:2201.07791*, 2022.

[23] L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proc. 28th annual ACM symposium on theory of computing*, STOC '96, pages 212–219, ACM, New York, 1996.

[24] L. K. Grover. Quantum mechanics helps in searching for a needle in a haystack. *Phys. Rev. Lett.*, 79(2):325–328, 1997.

[25] G. H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers.* Oxford, 4th edition, 1975.

[26] M. Hayashi, S. Ishizaka, A. Kawachi, G. Kimura, and T. Ogawa. *Introduction to Quantum Information Science.* Springer, 2014.

[27] J. Hidary. *Quantum Computing: An Applied Approach.* Springer, 2019.

[28] M. Hirvensalo. *Quantum Computing.* Springer, 2010.

[29] Yan Huang, Zhaofeng Su, Fangguo Zhang, Yong Ding, and Rong Cheng. Quantum algorithm for solving hyperelliptic curve discrete logarithm problem. *Quantum Information Processing*, 19(2):62, 2020.

[30] P. Kaye, R. Laflamme, and M. Mosca. *An Introduction to Quantum Computing.* Oxford University Press, New York, 2007.

[31] A. Yu. Kitaev. Quantum measurements and the Abelian stabilizer problem. *Arxiv:quant-ph/9511026*, 1995.

[32] A. Yu. Kitaev, A. H. Shen, and M. N. Vyalyi. *Classical and Quantum Computation.* American Mathematical Society, Boston, 2002.

[33] R. J. Lipton and K. W. Regan. *Quantum Algorithms via Linear Algebra: A Primer*. MIT Press, 2014.

[34] A. Mandviwalla, K. Ohshiro, and B. Ji. Implementing Grover's algorithm on the IBM quantum computers. In *2018 IEEE International Conference on Big Data*, pages 2531–2537, 2018.

[35] D. C. Marinescu and G. M. Marinescu. *Approaching Quantum Computing*. Pearson/Prentice Hall, Michigan, 2005.

[36] F. L. Marquezino, R. Portugal, and F. D. Sasse. Obtaining the quantum Fourier transform from the classical FFT with QR decomposition. *Journal of Computational and Applied Mathematics*, 235(1):74–81, 2010.

[37] N. D. Mermin. *Quantum Computer Science: An Introduction*. Cambridge University Press, New York, 2007.

[38] D. A. Meyer. Sophisticated quantum search without entanglement. *Phys. Rev. Lett.*, 85:2014–2017, 2000.

[39] Takashi Mihara and Shao Chin Sung. Deterministic polynomial-time quantum algorithms for Simon's problem. *Computational Complexity*, 12(3):162–175, 2003.

[40] A. Montanaro, R. Jozsa, and G. Mitchison. On exact quantum query complexity. *Algorithmica*, 71(4):775–796, 2015.

[41] M. Nakahara and T. Ohmi. *Quantum Computing: From Linear Algebra to Physical Realizations*. CRC Press, 2008.

[42] M. A. Nielsen and I. L. Chuang. *Quantum computation and quantum information*. Cambridge University Press, New York, 2000.

[43] I. Niven, H. S. Zuckerman, and H. L. Montgomery. *An Introduction to the Theory of Numbers*. Wiley, 5th edition, 1991.

[44] J. L. Park. The concept of transition in quantum mechanics. *Foundations of Physics*, 1(1):23–33, 1970.

[45] A. Pavlidis and D. Gizopoulos. Fast quantum modular exponentiation architecture for Shor's factoring algorithm. *Quantum Info. Comput.*, 14(7&8):649–682, 2014.

[46] R. Portugal. *Quantum Walks and Search Algorithms*. Springer, Cham, 2th edition, 2018.

[47] R. Portugal and F. L. Marquezino. Introdução à Programação de Computadores Quânticos. In *Conference CSBC 2019 – 38º JAI*, pages 1–51, Belém, Pará, 2019.

[48] J. Proos and C. Zalka. Shor's discrete logarithm quantum algorithm for elliptic curves. *Quantum Information and Computation*, 3(4):317–344, 2003.

[49] Daowen Qiu and Shenggen Zheng. Generalized Deutsch-Jozsa problem and the optimal quantum algorithm. *Phys. Rev. A*, 97:062331, 2018.

[50] Daowen Qiu and Shenggen Zheng. Revisiting Deutsch-Jozsa algorithm. *Information and Computation*, 275:104605, 2020.

[51] E. Rieffel and W. Polak. *Quantum Computing, a Gentle Introduction.* MIT Press, Cambridge, 2011.

[52] J. B. Rosser and L. Schoenfeld. Approximate formulas for some functions of prime numbers. *Illinois Journal of Mathematics*, 6(1):64 – 94, 1962.

[53] W. Scherer. *Mathematics of Quantum Computing: An Introduction.* Springer, 2019.

[54] P. W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proc. 35th Annual Symposium on Foundations of Computer Science*, pages 124 –134, 1994.

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

[56] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, 41(2):303–332, 1999.

[57] D. R. Simon. On the power of quantum computation. In *Proc. 35th Annual Symposium on Foundations of Computer Science*, pages 116–123, 1994.

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

[59] U. Skosana and M. Tame. Demonstration of Shor's factoring algorithm for $N = 21$ on IBM quantum processors. *Scientific Reports*, 11(1):16599, 2021.

[60] J. Stolze and D. Suter. *Quantum Computing, Revised and Enlarged: A Short Course from Theory to Experiment.* Wiley-VCH, 2008.

[61] G. Strang. *Linear Algebra and Its Applications.* Brooks Cole, 1988.

[62] C. P. Williams. *Explorations in Quantum Computing.* Springer, 2008.

[63] W. K. Wootters and W. H. Zurek. A single quantum cannot be cloned. *Nature*, 299:802–803, 1982.

[64] N. S. Yanofsky and M. Mannucci. *Quantum Computing for Computer Scientists.* Cambridge University Press, 2008.

[65] Zekun Ye, Yunqi Huang, Lvzhou Li, and Yuyi Wang. Query complexity of generalized Simon's problem. *Information and Computation*, 281:104790, 2021.

[66] C. Zalka. Grover's quantum searching algorithm is optimal. *Phys. Rev. A*, 60:2746–2751, 1999.