Computación Cuántica

Algoritmos cuánticos

Índice

Esquema

Algoritmos cuánticos

Estimación de Fase

Algoritmo de Shor

Amplificación de Amplitud

Algoritmo de Grover

Entornos de desarrollo

Ideas clave

10.1 Introducción y objetivos

La subrutina de estimación de fase (Quantum Phase Estimation, QPE, o simplemente Phase Estimation) permite extraer información no del registro cuántico, como hace la mayoría de subrutinas, sino de una operación que actúa sobre el registro cuántico, en concreto, permite determinar las *fases* asociadas a los autoestados de un operador unitario. La fase global, normalmente, no es observable, sin embargo QPE permite trasladar la fase global a otro registro cuántico de tal forma que sea observable.

El algoritmo de Shor es un algoritmo cuántico probabilístico que permite encontrar los factores de un número N de forma eficiente, en concreto en tiempo $O((logN)^3)$ y espacio $O(logN)$.

Muchas sistemas criptográficos de clave pública actuales, tales como RSA, quedarían obsoletos si fuera posible ejecutar el algoritmo de Shor en un procesador cuántico para ese tamaño del problema (RSA 2048) ya que podría romper RSA (y otros sistemas) en tiempo polinómico. En 2001, un grupo en IBM utilizando una máquina de 7 cúbits realizó la factorización del número 15 en sus factores 3 y 5.

El algoritmo de Grover resuelve un problema de tipo oráculo y encuentra la solución con una ventaja cuadrática frente a los mejores algoritmos clásicos. Es un algoritmo más simple e intuitivo que el de Shor, y tiene una elegante interpretación geométrica.

Inicialmente fue propuesto para encontrar un elemento en una base de datos no estructurada, sin embargo, su generalización ha extendido su campo de acción a otros algoritmos, las ideas básicas que componen este algoritmo son aplicables en un contexto mucho más amplio. Por ejemplo, podría usarse para buscar dos enteros $1 < a < b$ tales que $ab = n$ para algún número n, resultando en un algoritmo de factorización.

El algoritmo de Grover se basa en el concepto de amplificación de la amplitud y, aun-

que este es el término que se utiliza, en realidad sería más apropiado el de amplificación de la magnitud, como se verá. En cualquier caso es un algoritmo que demuestra la superioridad de la computación cuántica frente a la clásica en esta tarea.

Finalmente se presentarán algunos de los entornos de desarrollo para computación cuántica en la nube.

- Estimación de la fase
- Algoritmo de Shor
- Algoritmo de Grover y amplificación de amplitud
- Entornos de desarrollo

10.2 Estimación de fase

El algoritmo de estimación de fase, también conocido como algoritmo para la estimación del autovalor de un autovector de un operator unitario, fue desarrollado en su forma inicial por Alexei Kitaev en 1995. Es una de las subrutinas más importantes en computación cuántica y se utiliza como pieza fundamental de varios algoritmos cuánticos como el algoritmo de Shor o el algoritmo cuántico para resolver sistemas de ecuaciones lineales.

La puerta H, como se ha estudiado en temas anteriores, realiza la siguiente transformación:

H :

$|0\rangle \longrightarrow \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$

$|1\rangle \longrightarrow \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$

En general el operador transforma un estado en otro estado distinto.

$$H\begin{bmatrix}\alpha\\ \beta\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}\alpha\\ \beta\end{bmatrix} = \frac{1}{\sqrt{2}}(\alpha+\beta)|0\rangle + \frac{1}{\sqrt{2}}(\alpha-\beta)|1\rangle$$

Sin embargo, existen ciertos estados, los autoestados de H, que no son transformados por el operador.

Autoestados de H:

$$\lambda_{\pm} = \pm\frac{\sqrt{2\pm\sqrt{2}}}{2}|0\rangle + \frac{1}{\sqrt{2(2\pm\sqrt{2})}}|1\rangle$$

En concreto:

$$H\left[\frac{\sqrt{2+\sqrt{2}}}{2}|0\rangle + \frac{1}{\sqrt{2(2+\sqrt{2})}}|1\rangle\right] = \frac{\sqrt{2+\sqrt{2}}}{2}|0\rangle + \frac{1}{\sqrt{2(2+\sqrt{2})}}|1\rangle$$

$$H\left[-\frac{\sqrt{2-\sqrt{2}}}{2}|0\rangle + \frac{1}{\sqrt{2(2-\sqrt{2})}}|1\rangle\right] = \frac{\sqrt{2-\sqrt{2}}}{2}|0\rangle - \frac{1}{\sqrt{2(2-\sqrt{2})}}|1\rangle$$

El primer autoestado tiene ambos componentes con la misma fase mientras que el segundo tiene una diferencia de fase relativa entre sus componentes de π radianes. Cuando la puerta H actúa sobre el primer autoestado no lo modifica, sin embargo, cuando actúa sobre el segundo, el estado adquiere una fase global de π radianes.

Se ha estudiado que la fase global no es observable de forma que se puede afirmar que el estado queda sin cambios. Estos estado que bajo la acción de un operador U permanecen inalterados, excepto por una fase global, se denominan autoestados del operado U. Todo operador tiene un conjunto de estados distintos y únicos para los cuales el operador solo aplica una fase global, que se denomina autofase.

La subrutina de estimación de fase permite encontrar las autofases correspondientes a cada autoestado de un operador U. De forma más precisa, dado un operador unitario U y un estado cuántico $|\psi\rangle$, tal que $U|\psi\rangle = e^{2\pi i\theta}|\psi\rangle$, es decir, $|\psi\rangle$ es un autovector de U y $e^{2\pi i\theta}$ su correspondiente autovalor, el algoritmo estima el valor de la fase θ con alta probabilidad, ya que U es un operador unitario sobre un espacio vectorial complejo, sus autovalores son números complejos de norma 1.

El circuito que implementa el algoritmo está formado por dos registros, uno superior que contendrá la fase estimada y otro inferior en el estado $|\psi\rangle$, el estado del cual se

desea estimar su autovalor, como se muestra en la siguiente figura.

El algoritmo utiliza Phase kickback, para representar, utilizando la base de Fourier, la fase de U en el registro superior. A continuación, se utiliza la función descrita en el tema anterior, la inversa de la transformada cuántica de Fourier, para convertir el resultado a la base computacional, y poder así, realizar la medida que proporcionará la fase buscada.

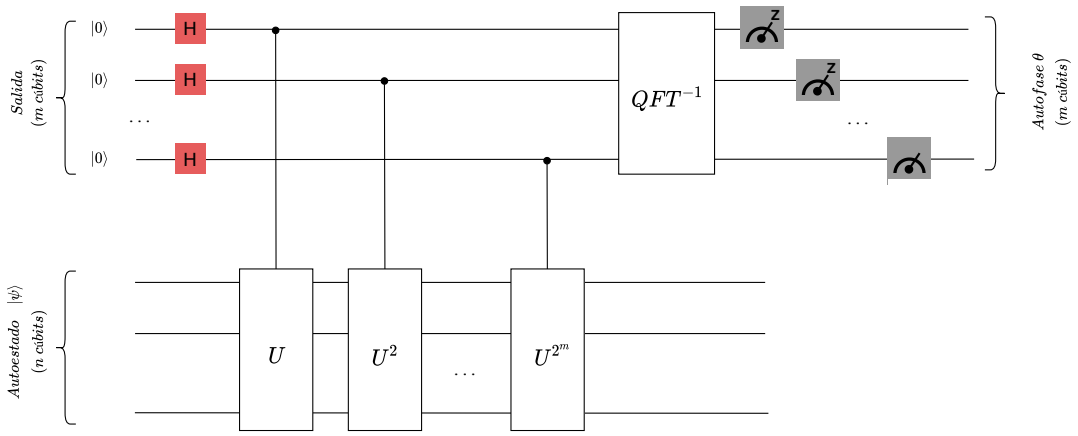

Figura 1: Algoritmo de Estimación de Fase. Elaboración propia.

Cuando se utiliza un cúbit de control para controlar una puerta U, debido al efecto de Phase Kickback, la fase del cúbit de control realizará una rotación proporcional a la fase $e^{2i\pi\theta}$.

Se puede aplicar una secuencia de puertas controladas con objeto de repetir esa rotación un número apropiado de veces hasta que la fase θ quede codificada como un número entre 0 y 2^m en la base de Fourier en el registro superior de m cúbits. Finalmente, utilizando la función $QFT^{\dagger}$el número, codificado en la base de Fourier, se convierte a la base computacional para, mediante el proceso de medida, obtener θ.

El circuito opera de la siguiente forma, tiene dos registros de entrada, el superior de m bits, inicializado a $|0\rangle^{\oplus m}$, contendrá el valor $2^n\theta$ al finalizar el algoritmo, el registro inferior, de n cúbits contiene el autoestado del operador, es decir, se parte del siguiente estado inicial del sistema: $|0\rangle^{\otimes m}|\psi\rangle$.

Se aplican puertas de Hadamard a todos los cúbits del registro superior, es decir $H^{\otimes m}|0\rangle^{\otimes m}|\psi\rangle$ de forma que se obtiene el estado:

$\frac{1}{\sqrt{2^n}}(|0\rangle + |1\rangle)^{\oplus m}|\psi\rangle$

A continuación se aplican las puertas controladas que aplican el operador unitario U al registro inferior de n cúbits solo en el caso de que el cúbit de control correspondiente sea 1.

Puesto que $U|\psi\rangle = e^{2\pi i\theta}|\psi\rangle$, entonces: $U^{2^j}|\psi\rangle = U^{2^j-1}U|\psi\rangle = U^{2^j-1}Ue^{2\pi i\theta}|\psi\rangle = \ldots = e^{2\pi i\theta}|\psi\rangle$

Aplicando todas las m operaciones controladas CU^{2^j} con $0 \leq j \leq m-1$ y utilizando la igualdad:

$|0\rangle \otimes |\psi\rangle + |1\rangle \otimes e^{2\pi i\theta}|\psi\rangle = (|0\rangle + e^{2\pi i\theta}|1\rangle) \otimes |\psi\rangle$

El estado evoluciona a:

$\frac{1}{\sqrt{2^n}}(|0\rangle + e^{2\pi i\theta 2^{m-1}}|1\rangle) \otimes \ldots \otimes (|0\rangle + e^{2\pi i\theta 2^1}|1\rangle) \otimes (|0\rangle + e^{2\pi i\theta 2^0}|1\rangle) \otimes |\psi\rangle =$

$\frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^m-1} e^{2\pi i\theta k}|k\rangle \otimes |\psi\rangle$

La expresión anterior coincide con la correspondiente a la aplicación de QFT derivada en el tema anterior:

$QFT|x\rangle = \frac{1}{\sqrt{N}}(|0\rangle + e^{2\pi i\frac{x}{2}}|1\rangle) \otimes (|0\rangle + e^{2\pi i\frac{x}{2^2}}|1\rangle) \otimes (|0\rangle + e^{2\pi i\frac{x}{2^3}}|1\rangle) \otimes \ldots \otimes (|0\rangle + e^{2\pi i\frac{x}{2^n}}|1\rangle)$

Con tan solo reemplazar x por $2^n\theta$, por lo tanto, para recuperar el estado $|2^n\theta\rangle$, bastará con aplicar QFT^{-1} sobre el registro superior llegando al estado:

$\frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^m-1} e^{2\pi i\theta k}|k\rangle \otimes |\psi\rangle \xrightarrow{QFT_m^{-1}} \frac{1}{\sqrt{2^n}} \sum_{r=0}^{2^m-1} \sum_{k=0}^{2^m-1} e^{-\frac{2\pi ik}{2^n}(r-2^m\theta)}|r\rangle \otimes |\psi\rangle$

Finalmente se aplica el proceso de medición de forma que la expresión anterior tiene un máximo próximo a $r = 2^n\theta$. Cuando $2^n\theta$ es un entero, midiendo en la base computacional se obtiene la fase buscada con alta probabilidad, si no lo es se obtiene un valor cercano.

Si como operador unitario tomamos la puerta T cuya fase queremos conocer, como el operador T tiene la siguiente representación matricial:

$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{bmatrix}$ y asigna:

T :

$|0\rangle \longrightarrow |0\rangle$

$|1\rangle \longrightarrow e^{\frac{i\pi}{4}}|1\rangle$

Por tanto, para el autoestado $|1\rangle$, $T|1\rangle = e^{2i\pi\theta}|1\rangle$ se esperaría encontrar $\theta = \frac{1}{8}$ ya que T es una rotación de $\pi/4$ alrededor del eje Z y como $2\pi\theta = \frac{\pi}{4} \Longrightarrow \theta = \frac{1}{8}$.

Serán necesarios tres cúbits para el registro superior donde se codificará la fase y un cúbit para el registro inferior donde se proporcionará como entrada al algoritmo el autoestado, que corresponde a $|1\rangle$, y para ello se aplica una puerta X al cúbit del registro inferior previamente inicializado a $|0\rangle$.

Puesto que $T = P(\pi/4)$, el circuito que implementa QPE se muestra en la siguiente figura, tras ejecutar el algoritmo se obtiene como resultado, con muy alta probabilidad, el número binario 001, que en representación decimal es 1.

Para finalmente obtener θ se deberá dividir el resultado observado entre 2^m con $m = 3$ por tanto: $\theta = \frac{1}{8} \longrightarrow \frac{2\pi}{8} = \frac{\pi}{4}$ que es exactamente el ángulo de rotación de la puerta T.

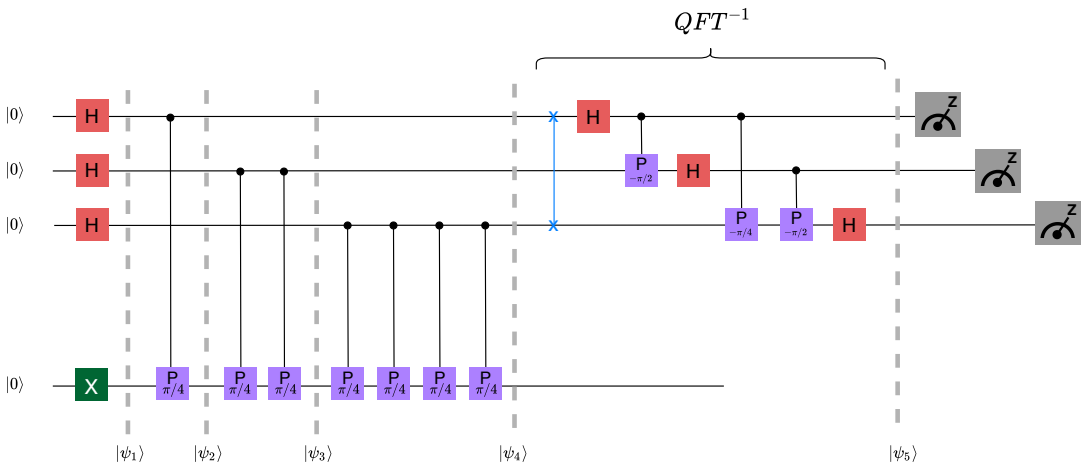

Figura 2: Algoritmo de Estimación de Fase para el operador T. Elaboración propia.

La siguiente figura muestra la evolución del estado cuántico para ambos registro utilizando la notación de círculos.

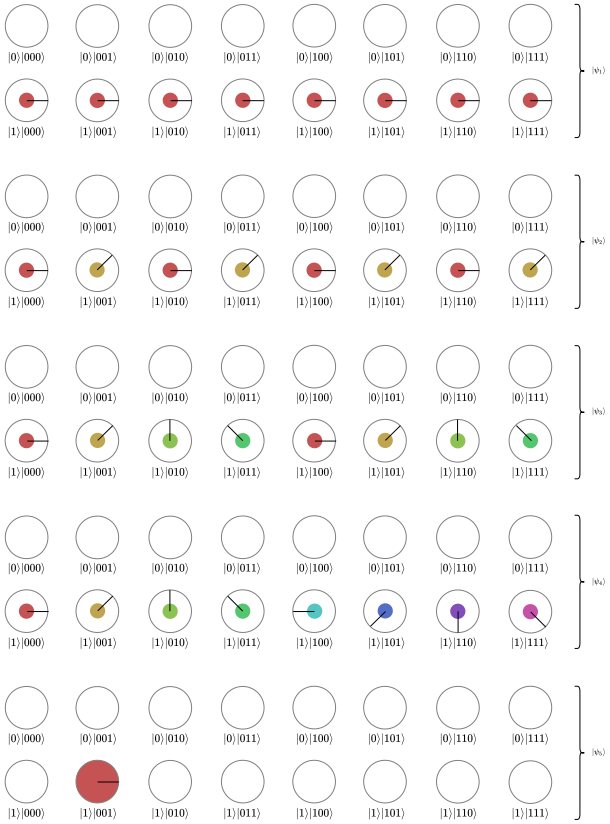

Figura 3: Evolución del estado cuántico en el algoritmo de Estimación de Fase para el operador T. Elaboración propia.

Para el caso en el que la probabilidad de obtener el resultado no sea tan alta como en el caso anterior, para poder mejorar el resultado se incrementa el número de cúbits en el registro superior, aumentando la precisión del resultado.

10.3 Algoritmo de Shor

El algoritmo de Shor es famoso por factorizar números enteros en tiempo polinomial. Dado que el algoritmo clásico más conocido requiere un tiempo superpolinomial para factorizar el producto de dos números primos, el esquema criptográfico RSA, se basa en que la factorización es computacionalmente imposible para números enteros lo suficientemente grandes.

Dado un número entero N, el algoritmo pretende encontrar otro número p, donde: $1 < p < N$, que divida N.

El algoritmo está estructurado en dos partes: una parte clásica que resuelve el problema de encontrar el orden y una parte cuántica para encontrar el periodo.

La estrategia consiste en reducir el problema de factorización a la búsqueda del período.

Una forma de factorizar un número entero es mediante la exponenciación modular. Específicamente, sea un número entero impar $N = N_1 N_2$, donde $1 < N_1, N_2 < N$. Se selecciona un número entero $k < N$ tal que su máximo común divisor $mcd(k, N) = 1$.

Se puede demostrar que existe un exponente $p > 0$ tal que $k^p \equiv 1(mod\ N)$.

Por definición, $x \equiv y(mod\ m)$ si y solo si m divide $x - y$.

Sea p el menor de esos númenros. Si p es par, entonces, según la definición de la operación módulo, N divide $k^p - 1 = (k^{p/2} - 1)(k^{p/2} + 1)$. Pero dado que la diferencia entre $n_1 = k^{p/2} + 1$ y $n_2 = k^{p/2} - 1$ es 2, n_1 y n_2 no tienen un factor común mayor que 2 y además, ambos números son distintos de cero.

Dado que se asumió que $N = N_1 N_2$ era impar, entonces N_1 es un factor de n_1 o n_2.

Si N_1 es un factor de n_1, dado que N_1 también es un factor de N, entonces N_1 divide tanto a n_1 como a N y se puede encontrar N_1 calculando $mcd(n_1, N)$.

Por lo tanto, si se puede calcular es número p, se pueden encontrar los factores de N de manera eficiente, ya que el máximo común divisor se puede calcular en tiempo polinomial.

Para encontrar p, se considera la siguiente secuencia de exponenciación modular:

$A = a_0, a_1, ...$, donde $a_i = k^i(mod\ N)$. Cada uno de los números a_i pertenece al conjunto finito 0, ..., $N - 1$, y por lo tanto, existen índices q y r tales que $a_q = a_r$. Si q y r son los indices más pequeños, se puede demostrar que $q = 0$ y A es una secuencia periódica con período r.

Como ejemplo, para $N = 15$ y $k = 7$, la secuencia de exponenciación modular es la siguiente:

1, 7, 4, 13, 1, 7, 4, 13, 1, ...

que presenta período 4, y dado que es un número par, se puede aplicar la idea anterior para encontrar:

$7^4 mod\ 15 \equiv 1 \Rightarrow 7^4 - 1\ mod\ 15 \equiv 0 \Rightarrow$

$\Rightarrow (7^2 - 1)(7^2 + 1)\, mod\ 15 \equiv 0 \Rightarrow 15\ divide\ 48 \cdot 50$, lo que implica que 15 divide a $48 \cdot 50$

que se puede usar para calcular los factores de 15 como $mcd(48, 15) = 3$ y $mcd(50, 15) = 5$.

Sin embargo, encontrar el período de la secuencia A no es clásicamente más fácil que buscar directamente factores de N, ya que puede ser necesario comprobar hasta $\sqrt{N}$ valores distintos de A antes de encontrar la repetición.

Sin embargo, con la computación cuántica, el período se puede encontrar en tiempo polinomial utilizando la Transformada Cuántica de Fourier (QFT) estudiada en el tema anterior.

La propiedad esencial de la QFT en el algoritmo de factorización es que puede calcular el período de una entrada periódica.

Dado que, como se ha visto, un problema de factorización se puede convertir en un problema de encontrar el período en tiempo polinómico, también se puede utilizar un algoritmo eficiente de búsqueda del período para factorizar números enteros de manera eficiente.

Sea $f(x) = a^x\ mod\ N$ donde $a < N$ y no tienen factores comunes. El periodo, u orden (r), es el entero más pequeño distinto de cero tal que $a^r\ mod\ N = 1$.

El algoritmo de shor utiliza QPE, estudiada en el apartado anterior, sobre el siguiente operador unitario $U|y\rangle \equiv |ay\ mod\ N\rangle$. Para entender de que forma tiene aplicación al problema de factorización el anterior operador se debe investigar cuales son sus

autoestados. Si se comienza con el estado $|1\rangle$, se puede observar que cada aplicaicón sucesiva del operador U multiplica el estado del registro por $a(mod\ N)$ y que tras r aplicaciones se llega nuevamente al estado inicial $|1\rangle$. Como ejemplo, $a = 5$ y $N = 29$:

$U|1\rangle \equiv |5 \cdot 1\ mod\ 29\rangle = 5$

$U^2|1\rangle \equiv |5 \cdot 5\ mod\ 29\rangle = 25$

$U^3|1\rangle \equiv |5 \cdot 25\ mod\ 29\rangle = 9$

$U^4|1\rangle \equiv |5 \cdot 9\ mod\ 29\rangle = 7$

$U^5|1\rangle \equiv |5 \cdot 7\ mod\ 29\rangle = 6$

$U^6|1\rangle \equiv |5 \cdot 6\ mod\ 29\rangle = 1$

$U^7|1\rangle \equiv |5 \cdot 1\ mod\ 29\rangle = 5$

$U^8|1\rangle \equiv |5 \cdot 5\ mod\ 29\rangle = 25$

...

Donde se observa un periodo de 6: 5, 25, 9, 7, 6, 1, 5, 25, 9, 7, 6, 1, 5, 25, ...

Por tanto, una superposición de estados en este ciclo ($|u_0\rangle$) correspondería a un autoestado de U.

$|u_0\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} |a^k\ mod\ N\rangle$

Siguiendo con el ejemplo anterior:

$|u_0\rangle = \frac{1}{\sqrt{6}}(|1\rangle + |5\rangle + |25\rangle + |9\rangle + |7\rangle + |6\rangle)$

$U|u_0\rangle = \frac{1}{\sqrt{6}}(U|1\rangle + U|5\rangle + U|25\rangle + U|9\rangle + U|7\rangle + U|6\rangle) = \frac{1}{\sqrt{6}}(|5\rangle + |25\rangle + |9\rangle + |7\rangle + |6\rangle + |1\rangle) = |u_0\rangle$

Este autoestado tiene un autovalor de 1, lo que no es muy interesante. Un estado propio más interesante podría ser uno en el que la fase sea diferente para cada uno de estos estados de la base computacional. En concreto el caso en el que la fase del estado k es proporcional a k:

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

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

Para el caso anterior:

$$|u_1\rangle = \frac{1}{\sqrt{6}}(|1\rangle + e^{-\frac{2\pi i}{6}}|5\rangle + e^{-\frac{4\pi i}{6}}|25\rangle + e^{-\frac{6\pi i}{6}}|9\rangle + e^{-\frac{8\pi i}{6}}|7\rangle + e^{-\frac{10\pi i}{6}}|6\rangle)$$

$$U|u_1\rangle = \frac{1}{\sqrt{6}}(|5\rangle + e^{-\frac{2\pi i}{6}}|25\rangle + e^{-\frac{4\pi i}{6}}|9\rangle + e^{-\frac{6\pi i}{6}}|7\rangle + e^{-\frac{8\pi i}{6}}|6\rangle + e^{-\frac{10\pi i}{6}}|1\rangle)$$

$$U|u_1\rangle = \frac{1}{\sqrt{6}} e^{\frac{2\pi i}{6}} (e^{-\frac{2\pi i}{6}}|5\rangle + e^{-\frac{4\pi i}{6}}|25\rangle + e^{-\frac{6\pi i}{6}}|9\rangle + e^{-\frac{8\pi i}{6}}|7\rangle + e^{-\frac{10\pi i}{6}}|6\rangle + e^{-\frac{12\pi i}{6}}|1\rangle)$$

$$U|u_1\rangle = e^{\frac{2\pi i}{6}} |u_1\rangle$$

Donde el periodo $r = 6$ aparece en el denominador de la fase. Se trata de un autovalor particularmente interesante ya que contiene el periodo . De hecho, r debe incluirse para asegurarse de que las diferencias de fase entre los r estados de la base computacional son iguales.

Este estado anterior, no es el único estado propio con este comportamiento, para generalizarlo, se puede multiplicar un número entero, s, a esta diferencia de fase, que aparecerá en el autovalor:

$$|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{\frac{-2\pi i s k}{r}} |a^k \ mod \ N\rangle$$

$$|u_s\rangle = e^{\frac{2\pi i s}{r}} |u_s\rangle$$

Siguiendo con el ejemplo anterior:

$$|u_s\rangle = \frac{1}{\sqrt{6}}(|1\rangle + e^{-\frac{2\pi i s}{6}}|5\rangle + e^{-\frac{4\pi i s}{6}}|25\rangle + e^{-\frac{6\pi i s}{6}}|9\rangle + e^{-\frac{8\pi i s}{6}}|7\rangle + e^{-\frac{10\pi i s}{6}}|6\rangle)$$

$$U|u_s\rangle = \frac{1}{\sqrt{6}}(|5\rangle + e^{-\frac{2\pi i s}{6}}|25\rangle + e^{-\frac{4\pi i s}{6}}|9\rangle + e^{-\frac{6\pi i s}{6}}|7\rangle + e^{-\frac{8\pi i s}{6}}|6\rangle + e^{-\frac{10\pi i s}{6}}|1\rangle)$$

$$U|u_s\rangle = \frac{1}{\sqrt{6}} e^{\frac{2\pi i s}{6}} (e^{-\frac{2\pi i s}{6}}|5\rangle + e^{-\frac{4\pi i s}{6}}|25\rangle + e^{-\frac{6\pi i s}{6}}|9\rangle + e^{-\frac{8\pi i s}{6}}|7\rangle + e^{-\frac{10\pi i s}{6}}|6\rangle + e^{-\frac{12\pi i s}{6}}|1\rangle)$$

$$U|u_s\rangle = e^{\frac{2\pi i s}{6}} |u_s\rangle$$

Ahora se tiene un autoestado único para cada valor entero s donde $0 \leq s \leq r - 1$

S iahroa se suman todos estos autoestados, las diferencias de fase cancelan todos los estados de la base computacional excepto $|1\rangle$:

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

Siguiendo con el ejemplo anterior:

$$\frac{1}{\sqrt{6}}(\quad |u_0\rangle = \frac{1}{\sqrt{6}}(|1\rangle + |5\rangle + |25\rangle + |9\rangle + |7\rangle + |6\rangle)...$$

$$+|u_1\rangle = \frac{1}{\sqrt{6}}(|1\rangle + e^{-\frac{2\pi i}{6}}|5\rangle + e^{-\frac{4\pi i}{6}}|25\rangle + e^{-\frac{6\pi i}{6}}|9\rangle + e^{-\frac{8\pi i s}{6}}|7\rangle + e^{-\frac{10\pi i s}{6}}|6\rangle)...$$

$$+|u_2\rangle = \frac{1}{\sqrt{6}}(|1\rangle + e^{-\frac{4\pi i}{6}}|5\rangle + e^{-\frac{8\pi i}{6}}|25\rangle + e^{-\frac{12\pi i}{6}}|9\rangle + e^{-\frac{16\pi i s}{6}}|7\rangle + e^{-\frac{20\pi i s}{6}}|6\rangle)...$$

$$+|u_3\rangle = \frac{1}{\sqrt{6}}(|1\rangle + e^{-\frac{6\pi i}{6}}|5\rangle + e^{-\frac{12\pi i}{6}}|25\rangle + e^{-\frac{18\pi i}{6}}|9\rangle + e^{-\frac{24\pi i s}{6}}|7\rangle + e^{-\frac{30\pi i s}{6}}|6\rangle)...$$

$$+|u_4\rangle = \frac{1}{\sqrt{6}}(|1\rangle + e^{-\frac{8\pi i}{6}}|5\rangle + e^{-\frac{16\pi i}{6}}|25\rangle + e^{-\frac{24\pi i}{6}}|9\rangle + e^{-\frac{32\pi i s}{6}}|7\rangle + e^{-\frac{40\pi i s}{6}}|6\rangle)...$$

$$+|u_5\rangle = \frac{1}{\sqrt{6}}(|1\rangle + e^{-\frac{10\pi i}{6}}|5\rangle + e^{-\frac{20\pi i}{6}}|25\rangle + e^{-\frac{30\pi i}{6}}|9\rangle + e^{-\frac{40\pi i s}{6}}|7\rangle + e^{-\frac{50\pi i s}{6}}|6\rangle) \ ) = |1\rangle$$

Puesto que el vector de la base computacional $|1\rangle$ es una superposición de estos autoestados, si se aplica el algoritmo de estimación de fase para el operador U utilizando como autoestado $|1\rangle$ se podrá medir la fase $\theta = \frac{s}{r}$ donde s es un valor aleatorio entero tal que $0 < s < r - 1$.

Finalmente se utiliza el algoritmo de fracción continua para θ para determinar r.

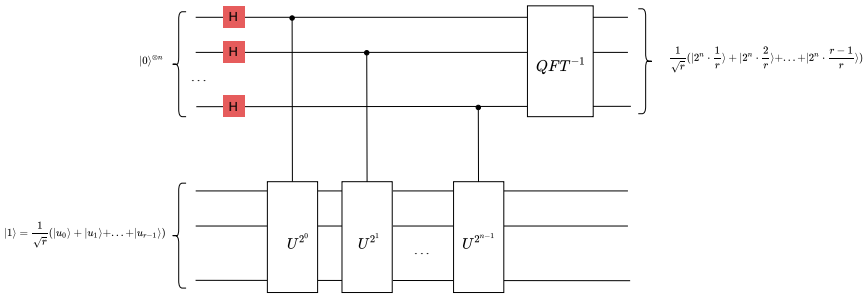

Figura 4: Algoritmo de Shor. Elaboración propia.

10.4 Algoritmo de Grover y amplificación de la amplitud

El algoritmo de Grover fue introducido por Lov Grover en 1996 y es uno de los algoritmos cuánticos más famosos. Inicialmente se propuso para problemas de búsqueda de un elemento marcado en una base de datos no estructurada. Sin embargo, el algoritmo de Grover es ahora una subrutina de varios otros algoritmos, como Grover Adaptive Search.

El algoritmo de Grover resuelve un problema de tipo oráculo y encuentra la solución realizando $O(\sqrt{N})$ consultas, mientras que los mejores algoritmos clásicos requieren llamadas $O(N)$. Es por tanto posible demostrar que el algoritmo de Grover es mejor que cualquier algoritmo clásico.

La mejora en la complejidad de la consulta se traduce en una ventaja solo bajo ciertas condiciones pues depende de la eficiencia de la implementación del oráculo y de si existe una estructura adicional en el problema que pueda ser explotada por algoritmos clásicos o cuánticos.

Se puede demostrar que la complejidad de la consulta del algoritmo de Grover es óptima, ningún algoritmo cuántico puede hacerlo mejor. Esta restricción es tan importante como el propio algoritmo ya que limita el poder de la computación cuántica.

El algoritmo de Grover es más simple y más fácil de comprender que el de Shor, y tiene una elegante interpretación geométrica.

Búsqueda en una base de datos no estructurada

El algoritmo de Grover, permite encontrar un elemento concreto dentro de una base de datos de N elementos, ordenada de forma aleatoria.

Se supone una lista de elementos de forma que entre ellos existe uno con unas características concretas que se desea localizar. Este elemento objetivo de la búsqueda, que es uno entre N elementos, se nombrará como x^*.

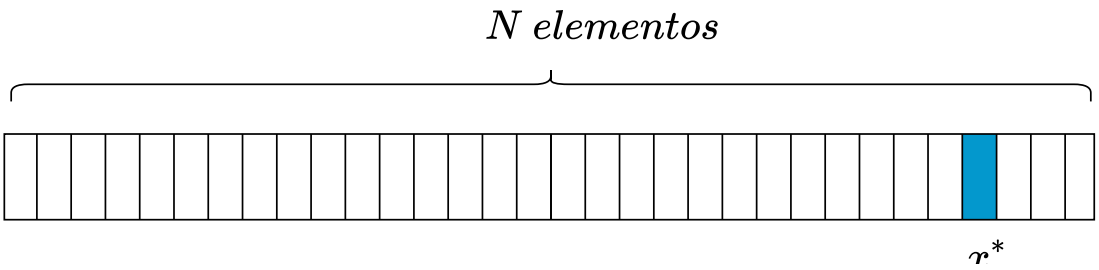

Figura 5: Elemento x^*, con las características deseadas, dentro de la lista de N elementos. Elaboración propia.

Para encontrar el elemento marcado, x^*, mediante un algoritmo clásico, sería necesario realizar una media de $\frac{N}{2}$ consultas a la base de datos y, en el caso peor habría que realizar N consultas. Sin embargo, el algoritmo de Grover permite encontrar el elemento deseado en $\sqrt{N}$ consultas, lo que supone una ventaja cuadrática frente al mejor algoritmo clásico y un ahorro de tiempo sustancial en listas grandes. Además, el algoritmo no se basa en identificar alguna estructura interna de los datos que se pueda aprovechar, lo que lo hace genérico y por ello proporciona una ventaja cuadrática para la resolución de muchos problemas frente a sus versiones clásicas.

Sea una función $f(x)$, cuyo dominio es el de los números enteros entre 0 y 2^{n-1}, es decir:

$$f(x),\ x \in 0, 1, 2, ..., 2^{n-1}$$

de forma que:

$$f(x) = \begin{cases} 0 & x \neq x^* \\ 1 & x = x^* \end{cases}$$

Lo que hace el algoritmo de Grover es encontrar ese valor x^*. Es decir, dada $f(x)$ el objetivo es encontrar x^*.

La función $f(x)$ se desconoce completamente, es un oráculo, lo único que puede hacerse es evaluarla, es decir realizar consultas al oráculo.

Una estrategia de fuerza bruta consistiría en evaluar $f(x)$ de forma secuencial hasta encontrar el valor deseado x^*.

$f(0) = 0,\ f(1) = 0,\ f(2) = 0, \ldots f(x^*) = 1$

Para la versión cuántica del problema lo primero es no utilizar números sino vectores, es decir:

$x \longrightarrow |x\rangle$

Por otro lado las funciones pasan a ser transformaciones lineales:

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

Por ello, la versión cuántica del enunciado del problema sería:

$$O|x\rangle = \begin{cases} |0\rangle & |x\rangle \neq |x\rangle^* \\ |1\rangle & |x\rangle = |x\rangle^* \end{cases}$$

Sin embargo, esta forma de plantear el problema no es válida pues no es una operación reversible, pues dado un resultado $|0\rangle$, se desconoce cual fue la correspondiente entrada $|x\rangle$.

El oráculo que se utiliza en el algoritmo de Grover marca el elemento o elementos que se buscan incorporándolos una fase negativa, es decir, el Oráculo O actúa de la siguiente forma, que sí es reversible:

$$O|x\rangle = \begin{cases} |x\rangle & x \neq x^* \\ -|x\rangle & x = x^* \end{cases}$$

Este operador tiene una representación matricial en forma de matriz diagonal con valor -1 en la entrada correspondiente al elemento marcado.

Generalizando y teniendo en cuenta que $(-1)^0 = 1$, $O|x\rangle = (-1)^{f(x)}|x\rangle$ donde:

$$f(x) = \begin{cases} 0 & O|x\rangle = (-1)^0|x\rangle = |x\rangle \\ 1 & O|x\rangle = (-1)^1|x\rangle = -|x\rangle \end{cases}$$

Por tanto, la versión cuántica del problema es ahora: dado un operador O encontrar el estado $|x\rangle$ tal que:

$$O|x\rangle = \begin{cases} |x\rangle & x \neq x^* \\ -|x\rangle & x = x^* \end{cases}$$

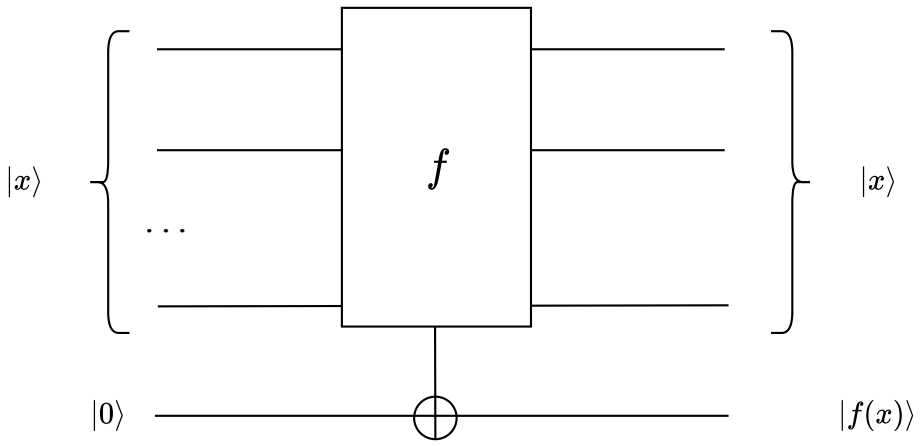

Figura 6: Versión reversible del oráculo utilizado en el algoritmo de Grover. Elaboración propia.

No se quiere comprobar cada valor de $|x\rangle$ individualmente pues en ese caso no se proporcionaría ventaja alguna frente a la estrategia clásica.

La computación cuántica permite que el estado del sistema no sea un estado clásico sino una superposición de estados clásicos de forma que al aplicar un operador a la superposición se pueda trabajar con todos ellos.

Como se ha visto anteriormente, se puede crear la superposición uniforme aplicando puertas de Hadamard a todos los cúbits. Es decir:

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

Por tanto, lo primero que deberá hacer el algoritmo es crear una superposición uniforme de estados $|s\rangle$ y aplicar sobre ella el operador correspondiente al oráculo:

$$O|s\rangle = \frac{1}{\sqrt{2^n}} O \sum_{x=0}^{2^n-1} |x\rangle = \frac{1}{\sqrt{2^n}} \left[\left(\sum_{x \neq x^*} |x\rangle \right) - |x^*\rangle \right]$$

El oráculo deja por tanto marcado con fase negativa el elemento buscado, sin embargo esa fase no es observable, si se realiza una medida la probabilidad de observar el elemento buscado es la misma que la de cualquier otro, pues en la superposición uniforme las amplitudes son las mismas para todos ellos. La probabilidad de observar

$|x^*\rangle$ sería:

$$P(|x^*\rangle) = \left[\frac{-1}{\sqrt{2^n}}\right]^2 = \frac{1}{2^n}$$

La forma de amplificar la amplitud de este elemento es el objetivo de la subrutina de amplificación de amplitud. Esta subrutina amplifica la amplitud del elemento marcado a costa de las amplitudes del resto de elementos, es decir reduciendo sus amplitudes de forma que el proceso de medida permitirá observar el elemento deseado con gran probabilidad.

La interpretación geométrica del algoritmo es muy elegante ya que transforma un problema en un espacio vectorial de 2^n dimensiones en otro de tan solo 2 dimensiones utilizando reflexiones.

Los dos estados que se necesitan considerar son $|x^*\rangle$ y la superposición uniforme $|s\rangle$. Estos dos vectores $|x^*\rangle, |s\rangle$ generan un subespacio de dimensión dos en el espacio vectorial $\mathbb{C}^{2^n}$. Forman una base pero $\langle x^*|s\rangle = \frac{1}{\sqrt{2^n}}$, es decir, no son del todo perpendiculares ya que $|x^*\rangle$ forma a su vez parte de la superposición, sin embargo, se puede considerar otro estado $|s'\rangle$ que sea ortogonal a $|x^*\rangle$ y que se obtiene simplemente retirándolo de la superposición $|s\rangle$.

$$|s'\rangle = |s\rangle - |x^*\rangle$$

Y por tanto:

$$\langle x^*|s'\rangle = 0$$

Siendo: $|s'\rangle = \frac{1}{\sqrt{2^n-1}} \sum_{x\neq x^*} |x\rangle$

Y ahora sí, $|x^*\rangle, |s'\rangle$ forman una base ortonormal.

La subrutina de amplificación de la amplitud, por tanto, toma como estado de partida la superposición uniforme descrita anteriormente:

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

Geométricamente, la evolución del algoritmo queda representada en la siguiente figura:

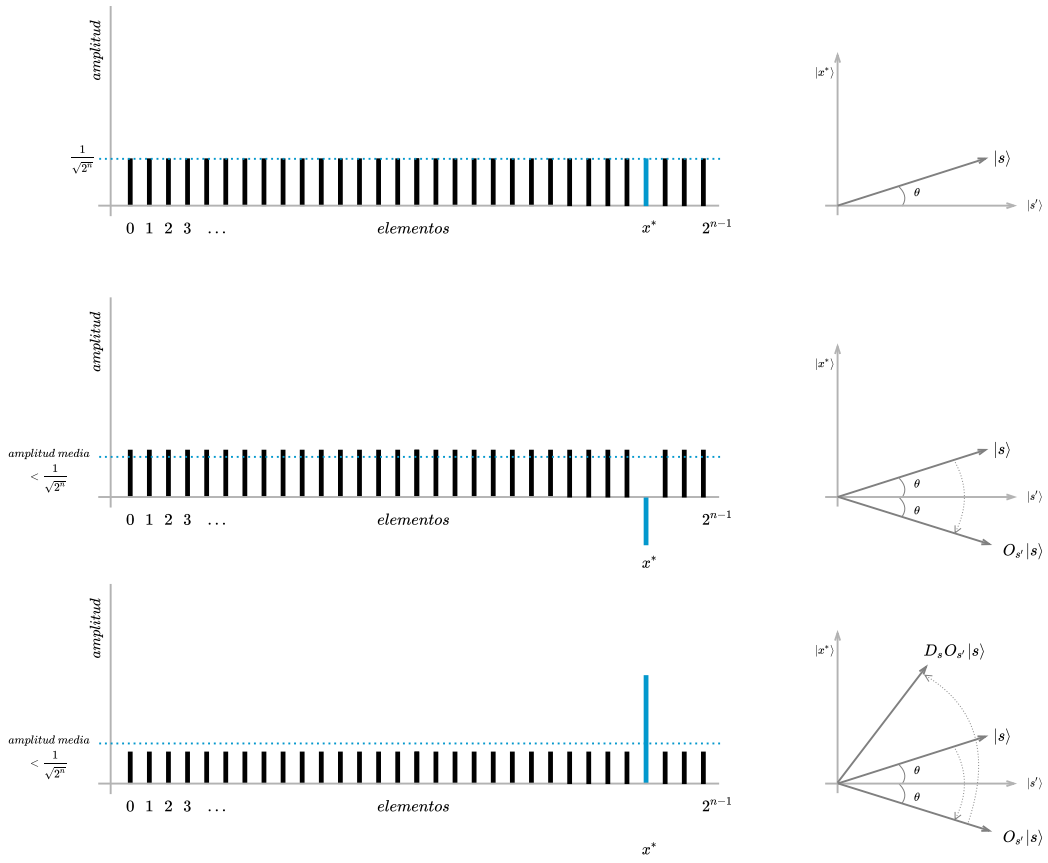

Figura 7: Superposición uniforme de partida en el algoritmo de Grover. Elaboración propia.

De Algebra Lineal se tiene, que la reflexión de un vector $|v\rangle$ sobre otro vector $|w\rangle$ tiene la forma:

$$R_w|v\rangle = (2|w\rangle\langle w| - I)|v\rangle$$

Es decir, el operador de rotación sobre el vector $|w\rangle$ es el siguiente: $R_w = 2|w\rangle\langle w| - I$

A continuación se aplica el oráculo a la superposición uniforme:

$$O_{s'}|s\rangle = \frac{1}{\sqrt{2^n}} O \sum_{x=0}^{2^n-1} |x\rangle = \frac{1}{\sqrt{2^n}} \left[\left(\sum_{x \neq x^*} |x\rangle \right) - |x^*\rangle \right]$$

Geométricamente, corresponde a una reflexión ($O_{s'}$) del estado $|s\rangle$ sobre $|s'\rangle$ lo cual implica que la amplitud del estado $|x^*\rangle$ se hace negativa, al ser justamente $|s'\rangle$ ortogonal a $|x^*\rangle$ como se ve en la figura anterior y por tanto, la amplitud media disminuye ligeramente, como también se puede observar en la figura.

A continuación se realiza una nueva reflexión, ahora con respecto a $|s\rangle$, denominada

operador de difusión de Grover, es decir, $D_s = 2|s\rangle\langle s| - I$ y que transforma el estado nuevamente a: $D_s O_{s'} |s\rangle$

Por lo tanto, la transformación $D_s O_{s'}$ rota el estado inicial $|s\rangle$ aproximándolo al estado $|x^*\rangle$.

La acción de la reflexión sobre $|s\rangle$ en el diagrama de amplitudes se puede interpretar como una reflexión con respecto a la media de las amplitudes. Dado que la amplitud media se había reducido por efecto de la primera reflexión $O_{s'}$, esta transformación amplifica la amplitud negativa de $|x^*\rangle$, mientras que disminuye las otras amplitudes.

Si se repite este proceso una segunda vez, $D_s O_{s'} D_s O_{s'} |s\rangle$ el estado se irá aproximando al estado buscado $|x^*\rangle$.

Después de repetir el proceso t pasos el estado se encontrará en:

$$|\psi_t\rangle = (D_s O_{s'})^t |s\rangle$$

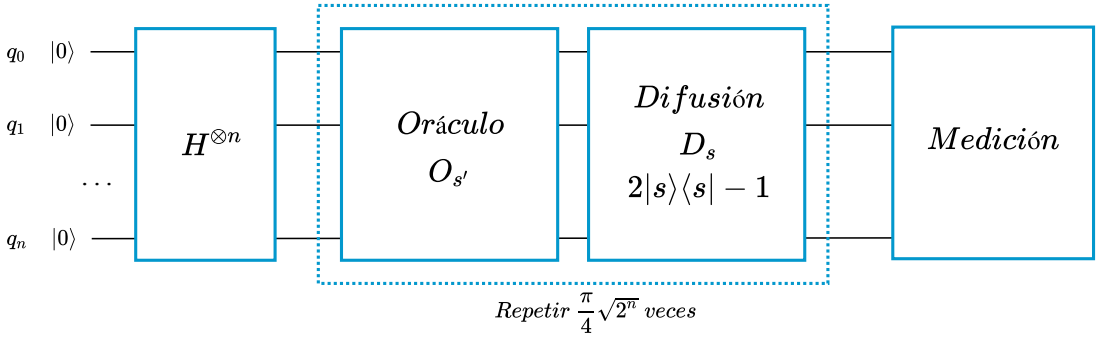

Figura 8: Esquema del algoritmo de Grover. Elaboración propia.

El número de veces que se debe repetir la secuencia de las dos rotaciones $D_s O_{s'}$ es de aproximadamente $\sqrt{2^n}$, es decir la raíz cuadrada del número de elementos en la base de datos. Si se aplicarán más iteraciones la consecuencia sería que alejaría al estado de $|x^*\rangle$ por lo que es fundamental aplicar el número de iteraciones correcto.

$$t \longrightarrow \frac{\pi\sqrt{2^n}}{4}$$

En el caso de que exista un número m de múltiples soluciones, se puede demostrar que aproximadamente será suficiente con $\sqrt{\frac{2^n}{m}}$.

La siguiente figura muestra una implementación del alagoritmo de Grover para en-

contrar el elemento marcado (11) entre la lista de cuatro elementos utilizando dos cúbits.

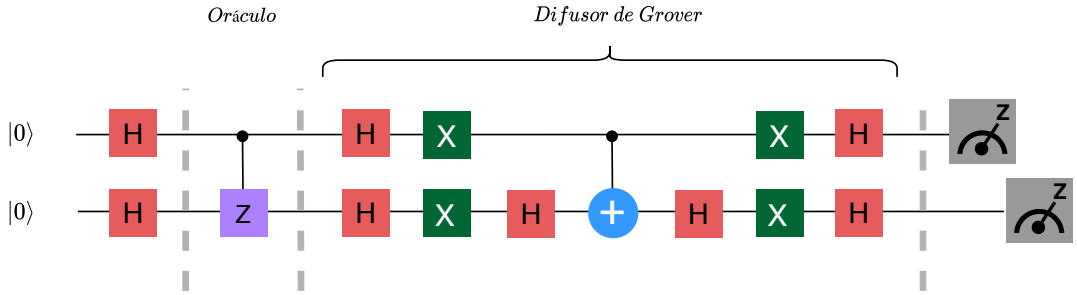

Figura 9: Algoritmo de Grover de busqueda del elemento 11 con dos cúbits. Elaboración propia.

10.5 Entornos de desarrollo en la nube

Los servicios de computación cuántica en la nube ofrecen acceso a procesadores cuánticos, simuladores y otros servicios a través de Internet.

Existen varias plataformas disponibles. En 2016, IBM desplegó en la nube el entorno de desarrollo Quantum Experience que permitía crear y ejecutar circuitos cuánticos simples mediante su librería QISKit en Python. En 2017, Rigetti Computing ofrecieron también acceso a sus procesadores a través del a nube mediante su biblioteca pyQuil también con Python.

Desde entonces han surgido otras plataformas que están acelerado la adopción por parte de los usuarios y el desarrollo de esta nueva industria. En el entrono académico se pueden utilizar la computación cuántica para facilitar la enseñanza de la mecánica cuántica en general, así como de este nuevo paradigma de computación. Para el mundo de la investigación, permite la aplicación de los recursos de computación cuántica aplicados a proyectos científicos. Incluso en el desarrollo de juegos o en el desarrollo artístico, tiene cabida el uso de la computación cuántica y el despliegue en la nube esta favoreciendo su expansión.

A continuación se describen brevemente algunos de los más importantes desarrollos

en este campo.

QISKit de IBM. IBM Quantum proporciona acceso a procesadores cuánticos basados en circuitos superconductores, así como a simuladores de computación de alto rendimiento. Se puede acceder a estos mediante programación utilizando el entorno de desarrollo Qiskit basado en Python, o mediante una interfaz gráfica interactiva, ambos basados en el estándar OpenQASM.

Xanadu Quantum Cloud de Xanadu, proporciona servicios en la nube de acceso a computadoras cuánticas fotónicas.

Forest de Rigetti Computing, proporciona herramientas para computación cuántica con un lenguaje de programación y algoritmos de ejemplo.

Azure Quantum. LIQUi|>de Microsoft, proporciona acceso a simuladores cuanticos utilizando el lenguaje de programación F#. Tiene como objetivo permitir la experimentación con algoritmos cuánticos antes de que las computadoras cuánticas físicas estén disponibles para su uso.

Quantum Playground de Google, que cuenta con un simulador con una interfaz simple y un lenguaje de secuencias de comandos y visualización de estado cuántico en 3D.

Quantum Inspire de Qutech es la primera plataforma en Europa que proporciona acceso a un emuloador y dos procesadores de computación cuántica en la nube, uno de cinco cúbits basado en circuitos superconductores y otro de dos cúbits basado en el spin del electrón.

Amazon Braket es un servicio en la nube que proporciona un entorno de desarrollo para diseñar algoritmos cuánticos y ejecutarlos en procesadores cuánticos simulados y reales ijmplementados con diferentes tecnologías.

Forge de QC Ware, proporciona acceso a procesadores de D-Wave, así como a simuladores de Google e IBM.

10.6 Referencias bibliográficas

Nielsen and Chuang (2011) Quantum Computation and Quantum Information

Aaronson (2013), Quantum Computing Since Democritus

Eric R. Johnston, Nic Harrigan y Mercedes Gimeno-Segovia (2019), Programming Quantum Computers

Eleanor Rieffel and Wolfgang Polak (2011), Quantum Computing

Robert Sutor (2019), Dancing with cúbits