Universidad Complutense de Madrid
Facultad de Informática
Curso 2021 – 2022

Computación Cuántica
Markel Álvarez Martínez

Hoja de problemas 2

1. El algoritmo utilizado para determinar cual es el periodo de una función de n bits es el algoritmo de Shor. Este busca la factorización de un valor dado un $N > 0$ asumiendo que es un valor semi-primo con factores desconocidos.

   Aquellos números enteros menores que N que a su vez son coprimos con N forman un grupo finito que se multiplica con el módulo de N. Existe un entero a que debe tener un orden finito r siendo el menor entero positivo tal que $a^r \equiv 1 \mod (N)$, por lo tanto, $N|(a^r - 1)$. Supongamos que podemos obtener r, y es par. Entonces $a^r - 1 = (a^{\frac{r}{2}} - 1)(a^{\frac{r}{2}} + 1) \equiv 0 \mod (N) \rightarrow N|(a^{\frac{r}{2}} - 1)(a^{\frac{r}{2}} + 1)$.

   r es el número entero positivo más pequeño tal que a $a^r \equiv 1$, así que N no puede dividir a $(a^{\frac{r}{2}} - 1)$. Si N tampoco divide $(a^{\frac{r}{2}} + 1)$, entonces N debe tener un factor común no trivial con $(a^{\frac{r}{2}} - 1)$ y $(a^{\frac{r}{2}} + 1)$. Si N es el producto de dos primos, esta es la única factorización posible.

2. Primero debemos elegir un número aleatorio entre 2 y $N - 2$ $(2, 19)$, por ejemplo 15. El siguiente paso consiste en saber cuantos qubits vamos a utilizar, como en nuestro caso $N = 21$ $(2^4 < 21 < 2^5)$ vamos a utilizar 5 qubits como se puede ver en la figura 1 donde M hace referencia al número de qubits y x al número elegido aleatoriamente.

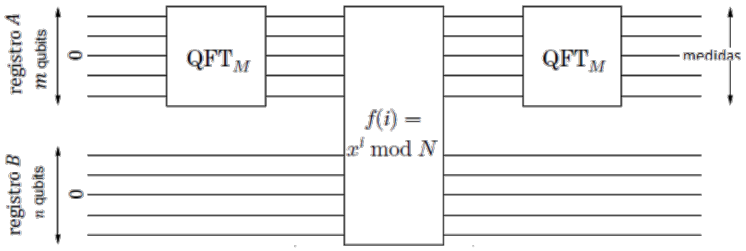

Figura 1: Algoritmo de Shor

   La primera QFT en el registro A produce una superposición igual de los qubits de A, es decir, el estado resultante $\left(\frac{1}{\sqrt{M}} \sum_{i=0}^{M} |i, 0\rangle\right)$. A este lo sigue un circuito de exponenciación modular que calcula la función $f(i) = x^i \mod (N))$ en el segundo registro con el estado resultante siendo $\left(\frac{1}{\sqrt{M}} \sum_{i=0}^{M} |i, f(i)\rangle\right)$. Antes de aplicar la siguiente QFT, hacemos una medición del registro B, si el valor medido es s, el

estado resultante será $\left(\frac{1}{\sqrt{\frac{M}{r}}}\sum_{i=0,f(i)=0}^{M}|i,s\rangle\right)$, r es el periodo de $f(i)$. El registro A es una superposición periódica con el período r. Por último, dado la propiedad de QFT el resultado es el estado $\frac{1}{\sqrt{r}}\sum_{i=0}^{r}|i\left(\frac{M}{r}\right),s\rangle$. La medida del registro A generará una múltiplo de $\frac{M}{r}$.

3. ▪ Algoritmo de Grover: se utiliza para resolver problemas de búsqueda. La búsqueda en una secuencia no ordenada de datos con N componentes en un tiempo $O(N^{\frac{1}{2}})$, y con una necesidad adicional de espacio de almacenamiento de $O(\log N)$. En una búsqueda normal de un dato, si tenemos una secuencia desordenada se debe realizar una inspección lineal, que necesita un tiempo $O(N)$, por lo que el algoritmo de Grover es una mejora bastante sustancial, evitando, además, la necesidad de la ordenación previa.

   Para el caso en el que tengamos un circuito con un *input* de 2 qubits en la que que queramos encontrar la cadena $|\omega\rangle = |11\rangle$ podemos usar un circuito parecido al de la figura 2 en el que podemos observar el oráculo y la difusión de este.

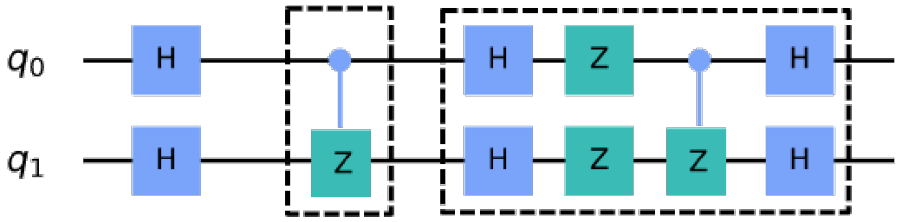

Figura 2: Oráculo y difusor

Si queremos encontrar la cadena $|\omega_1\rangle = |110\rangle$ y $|\omega_2\rangle = |111\rangle$ podemos usar el siguiente circuito de 3 qubits:

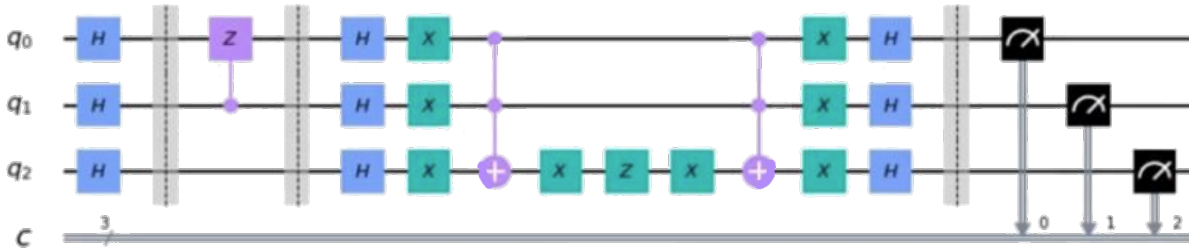

Figura 3: Algoritmo de Grover

Este algoritmo es interesante porque puede ser usado para el cálculo de la media y la mediana de un conjunto de números, y para resolver otros problemas de naturaleza análoga. También se puede utilizar para resolver algunos problemas de naturaleza NP-completa, por medio de inspecciones exhaustivas en un espacio de posibles soluciones.

- Algoritmo de Deutsch-Jozsa: se tiene una función (que puede considerarse como un oráculo o caja negra) $f(x_1, x_2, \ldots, x_n)$ que toma n bits de entrada $x_1, x_2, \ldots, x_n$ y devuelve un valor binario $f(x_1, x_2, \ldots, x_n) = 0$ ó 1. El objetivo es determinar si la función es constante (0 en todas las entradas o 1 en todas las entradas) o balanceada (devuelve 1 para la mitad de las entradas y 0 para la otra mitad).

  El problema es determinar cómo es la función (constante o balanceada) aplicando entradas a la caja negra y observando su salida. A modo de ejemplo, considérese la función $f(x) = x\,\%2$, esta función devuelve 1 si el argumento es impar y 0 si el argumento es par, por lo que se trata de una función balanceada. La función del algoritmo sería la de llegar a esta misma conclusión con el menor número posible de iteraciones, algo que en el caso clásico requeriría la evaluación repetida de la función hasta alcanzar dos resultados diferentes, y por tanto el número de iteraciones dependería del orden en el que se escogieran las variables de entrada.

  Este algoritmo es interesante porque es un ejemplo claro de problema que resulta sencillo de resolver para un ordenador cuántico pero muy complejo para un ordenador clásico, que en el peor de los casos necesita $2^{n-1} + 1$ iteraciones para determinar si la función es constante o balanceada.

- Algoritmo de Bernstein–Vazirani: permite conocer una cadena s binaria ($s = 0010110101001$) que está contenida en una función. Más concretamente, se sabe que dicha función toma la forma $f(x) = sx \mod (2)$, donde x es otra cadena y la multiplicación se entiende como producto binario.

  Este algoritmo funciona de manera similar al de Deutsch-Jozsa, pero en vez de tratar de distinguir entre clases de funciones, busca la cadena que caracteriza a la función dada. Supongamos

a modo de ejemplo que se participa en un juego consistente en encontrar un número oculto escrito en código binario. Con la versión clásica del algoritmo, la única manera de obtener la solución sería ir haciendo comprobaciones del número oculto bit a bit, lo cual requiere al menos N ejecuciones, siendo N el número de bits de s (complejidad $O(N)$). En el caso del algoritmo de Bernstein-Vazirani si se consigue codificar dicho número en la cadena s, una única ejecución del algoritmo bastaría para encontrar el número completo.

La importancia de este algoritmo radica en la superioridad que muestra frente a su equivalente clásico, pudiéndose encontrar la cadena buscada tras una única ejecución.

4.