MOOC: Computación y Criptografía Cuánticas

Lección 3: Algoritmos Cuánticos

Soluciones a los ejercicios propuestos

Ejercicio 3.2.1. Pon un ejemplo de función periódica y 2 a 1 definida en Z_2^3.

Un elemento de Z_2^3 es una cadena de 3 bits que podemos denotar abc, donde a, b y c pueden tomar los valores 0 y 1.

Un ejemplo de función periódica y 2 a 1 puede ser la función $f : Z_2^3 \to Z_2^3$ dada por $f(abc) = 0bc$, es decir f deja igual todas las cadenas de 3 bits que empiezan por 0 y cambia el 1 por un 0 en las cadenas de 3 bits que empiezan por 1. La imagen de f está formada por las cuatro cadenas de 3 bits que empiezan por 0.

Obviamente f es una función 2 a 1 puesto que $f(0bc) = f(1bc)$ y es una función periódica de periodo 100 pues $f((abc) \oplus (100)) = 0bc = f(abc)$.

Ejercicio 3.2.2 Calcula la probabilidad de que al aplicar $n-1$ veces el algoritmo cuántico que resuelve el problema de Simon se obtenga un sistema de rango $n-1$.

Sea P la probabilidad de que con $n-1$ evaluaciones del algoritmo se obtenga un sistema lineal homogéneo de rango máximo. La ecuación $T \cdot k = 0$ tiene exactamente 2^{n-1} soluciones en Z_2^n (número de vectores de un subespacio de dimensión $n-1$), todas con la misma probabilidad de ser el resultado del algoritmo.

Para que la primera ecuación sea linealmente independiente, basta que el vector k sea no nulo, luego será independiente con probabilidad $\dfrac{2^{n-1}-1}{2^{n-1}}$. Si ya tenemos j ecuaciones independientes, la $(j+1)$-ésima ecuación será independiente si el vector k no está en el subespacio generado por los anteriores, lo que ocurre con probabilidad $\dfrac{2^{n-1}-2^j}{2^{n-1}}$.

Por tanto, la probabilidad de que al aplicar $n-1$ veces el algoritmo se obtengan $n-1$ ecuaciones linealmente independientes es:

$$P = \frac{2^{n-1}-2^0}{2^{n-1}} \frac{2^{n-1}-2^1}{2^{n-1}} \cdots \frac{2^{n-1}-2^{n-2}}{2^{n-1}} = \prod_{j=1}^{n-1} \left(1 - \frac{1}{2^j}\right)$$

El producto anterior decrece cuando n se hace grande.

Sea $l = \displaystyle\prod_{j=1}^{\infty} \left(1 - \frac{1}{2^j}\right)$, tomando logaritmos y desarrollando en series de potencias queda

$$\log(l) = \sum_{j=1}^{\infty} \log\left(1 - \frac{1}{2^j}\right) = -\sum_{j=1}^{\infty}\sum_{k=1}^{\infty} \frac{1}{k}\left(\frac{1}{2}\right)^{jk} = -\sum_{m=1}^{\infty} b_m \left(\frac{1}{2}\right)^m$$

donde b_m es la suma de los inversos de los divisores de m. Puesto que m tiene como mucho m divisores, el primero de ellos es 1 y los demás serán mayores o iguales que 2, la suma de sus inversos es $b_m \leq 1 + \dfrac{m-1}{2}$. Por lo tanto

$$-\log(l) \leq \sum_{m=1}^{\infty} \left(1 + \frac{m-1}{2}\right)\left(\frac{1}{2}\right)^m = \frac{3}{2} \qquad \Longrightarrow \qquad P \geq l \geq e^{-3/2} > \frac{1}{5}$$

En consecuencia, aplicando el algoritmo $n-1$ veces la probabilidad de obtener un sistema de rango $n-1$ es mayor que $1/5$.

Ejercicio 3.3.1 Determina el número óptimo de iteraciones del algoritmo de Grover para encontrar la solución en un problema de búsqueda no estructurada con 10000 datos y solución única.

Comprueba que la probabilidad de acierto no mejora al aumentar el número de iteraciones por encima del número óptimo.

El número optimo de iteraciones es la parte entera de $\frac{\pi}{4}\sqrt{N}$, que es 78, y vamos a calcular en este caso la probabilidad de acierto, que será $|b_{78}|^2$.

Sabemos que que $b_0 = m_0 = \frac{1}{100}$ y para $k \geq 0$ se verifica:

$$\begin{pmatrix} m_{k+1} \\ b_{k+1} \end{pmatrix} = \begin{pmatrix} \frac{N-2}{N} & -\frac{2}{N} \\ \frac{2N-2}{N} & \frac{N-2}{N} \end{pmatrix} \begin{pmatrix} m_k \\ b_k \end{pmatrix}$$

Usamos un software matemático y definimos la sucesión recursiva de las amplitudes. Por ejemplo, en Maple esto podría hacerse con el procecimiento:

```
>amplitudes:= proc(k, N)
local Mat, vampl, j;
vampl:=Vector([1/sqrt(N),1/sqrt(N)]);
Mat:=Matrix([[(N-2)/N, -2/N],[(2*N-2)/N, (N-2)/N]]);
for j to k do
vampl:=Mat.vampl;
od;
vampl
end:
```

Invocando el procedimiento con la instrucción `amplitudes(78,10000)` se obtiene que

$$m_{78} = 0.000007701973854, \qquad b_{78} = 0.9999997034$$

Es decir, la probabilidad de acierto es $0.9999997034^2 = 0.9999994068$.

Sin embargo con 100 iteraciones se obtiene que

$$m_{100} = -0.04252704378 \qquad b_{100} = 0.9050763173$$

y la probabildiad de acierto es bastante menor.

Ejercicio 3.4.1 Determina la QFT de los cuatro estados de la base de H_2 y la matriz asociada a la transformación unitaria F_2.

De acuerdo con la definición de la QFT es

$$F_2|j\rangle = \frac{1}{\sqrt{4}} \sum_{k=0}^{3} e^{\frac{2i\pi jk}{4}} |k\rangle$$

Para $j = 0$ se tiene que $e^{\frac{i\pi jk}{2}} = 1$ para todo k y por lo tanto

$$F_2|0\rangle = \frac{1}{2}(|0\rangle + |1\rangle + |2\rangle + |3\rangle).$$

Para $j = 1$, se tiene

$$F_2|1\rangle = \frac{1}{2}(e^0|0\rangle + e^{i\pi/2}|1\rangle + e^{i\pi}|2\rangle + e^{3i\pi/2}|3\rangle) = \frac{1}{2}(|0\rangle + i|1\rangle - |2\rangle - i|3\rangle)$$

Análogamente para $j = 2$ se obtiene

$$F_2|2\rangle = \frac{1}{2}(e^0|0\rangle + e^{i\pi}|1\rangle + e^{2i\pi}|2\rangle + e^{3i\pi}|3\rangle) = \frac{1}{2}(|0\rangle - |1\rangle + |2\rangle - |3\rangle).$$

Y finalmente, para $j = 3$, es

$$F_2|3\rangle = \frac{1}{2}(e^0|0\rangle + e^{3i\pi/2}|1\rangle + e^{3i\pi}|2\rangle + e^{9i\pi/2}|3\rangle) = \frac{1}{2}(|0\rangle - i|1\rangle - |2\rangle + i|3\rangle).$$

De esta forma, la matriz asociada a la transformación F_2 es

$$\frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix}$$

Ejercicio 3.4.2 Demuestra que si $f = f_0, f_1, \ldots, f_{Q-1}$ es periódica de periodo T, divisor de Q, su transformada cuántica de Fourier $F_n(f_0, f_1, \ldots, f_{Q-1}) = (\hat{f}_0, \hat{f}_1, \cdots, \hat{f}_{Q-1})$ verifica que:

$$F_n\left(\sum_{j=0}^{Q-1} f_j|j\rangle\right) = \sum_{k=0}^{T-1} \hat{f}_{wk}|wk\rangle$$

Sabemos que, de acuerdo con la definición de la QFT, es $\hat{f}_k = \frac{1}{\sqrt{Q}} \sum_{j=0}^{Q-1} \sigma^{jk} f_j$, con $\sigma = e^{\frac{i2\pi}{Q}}$.

Sea $w = Q/T$. Si k no es múltiplo de w, teniendo en cuenta que $f_j = f_{j+T}$ para todo j, esta suma la podemos desarrollar del siguiente modo:

$$\hat{f}_k = \frac{1}{\sqrt{Q}}\left(\sigma^0 f_0 + \sigma^k f_1 + \cdots + \sigma^{Tk} f_T + \sigma^{(T+1)k} f_{T+1} + \cdots + \sigma^{2Tk} f_{2T} + \cdots\right)$$

$$= \frac{1}{\sqrt{Q}}\left(\left(\sigma^0 + \sigma^{Tk} + \cdots + \sigma^{(w-1)Tk}\right) f_0 + \left(\sigma^k + \sigma^{(T+1)k} + \cdots\right) f_1 + \cdots\right)$$

Sumando las progresiones geométricas, y teniendo en cuenta que $\sigma^{wTk} = e^{\frac{i2\pi Qk}{Q}} = e^{i2\pi k} = 1$, resulta:

$$\hat{f}_k = \frac{1}{\sqrt{Q}}\left(\frac{1-\sigma^{wTk}}{1-\sigma^{Tk}} f_0 + \sigma\frac{1-\sigma^{wTk}}{1-\sigma^{Tk}} f_1 + \cdots\right) = 0$$

Observemos que el desarrollo anterior no es válido cuando k es múltiplo de w. Finalmente, aplicando la linealidad de F_n y el hecho de que $\hat{f}_k = 0$ si k no es múltiplo de w, se obtiene de manera inmediata la expresión

$$F_n\left(\sum_{j=0}^{Q-1} f_j|j\rangle\right) = \sum_{k=0}^{T-1} \hat{f}_{wk}|wk\rangle$$

Ejercicio 3.5.1 Uno de los mejores algoritmos clásicos de factorización es el algoritmo de Pollard y Strassen de 1976, que tiene complejidad $O(2^{n/4}n^2)$, siendo n el número de dígitos. Si se quiere factorizar un número con $n = 200$ dígitos, calcula el tiempo necesario para realizar $2^{n/4}n^2$ operaciones, en una máquina que realice 10^9 operaciones por segundo.

Teniendo en cuenta que el algoritmo de Shor tiene complejidad $O(\log^4(N)\log\log(N))$, con $N = 2^n$, compara el tiempo anterior con el que se tardaría en realizar $n^4\log(n)$ operaciones.

Poniendo $n = 200$, se tiene que $2^{n/4}n^2 = 45035996273704960000$ y estas operaciones se realizarían en $45035996273704960000 \cdot 10^{-9} = 4.503599627 \cdot 10^{10}$ segundos, dividiendo por 3600, por 24 y por 365 se tiene una estimación de 1428 años.

Por otra parte, con el algoritmo de Shor y la misma velocidad de cálculo se tendría $n^4\log(n) \approx 8.477307786 \cdot 10^9$ operaciones, que se realizarían en 8.5 segundos.