Implementación del algoritmo de Grover

Rivano Antonella, Gallo García Camila, Meroi Franco

June 2025

Introducción

El algoritmo de Grover es uno de los ejemplos más representativos del potencial de la computación cuántica frente a los enfoques clásicos. Se trata de un algoritmo del tipo "caja negra" (*black-box*), pensado para encontrar un elemento marcado dentro de un conjunto desordenado de N objetos, con una ventaja cuadrática en la cantidad de consultas necesarias respecto a cualquier estrategia clásica.

En el enfoque clásico, una búsqueda no estructurada requiere, en promedio, $O(N)$ evaluaciones para dar con el dato buscado, ya que no hay ninguna estructura que se pueda aprovechar para acelerar el proceso. En cambio, el algoritmo de Grover reduce ese número a $O(\sqrt{N})$ consultas a un oráculo, alcanzando una mejora cuadrática que, además, se sabe que es la mejor posible en este tipo de problema dentro del marco de la computación cuántica[1].

El origen de esta ventaja está en la utilización de la coherencia cuántica como un recurso. El procedimiento [2] consiste en aplicar repetidamente una combinación de operaciones: primero, un oráculo que cambia el signo de la amplitud de los elementos buscados; luego, el operador de difusión , que refleja el estado respecto del valor medio de las amplitudes.

En este trabajo se implementará este algoritmo en un espacio de Hilbert de 4 qubits para encontrar dos posibles elementos.

Algoritmo

En esta sección se cubriran los fundamentos teóricos del algoritmo de Grover y cómo implementarlo.

Como se destacó en la introducción, el algoritmo tiene dos partes esenciales. La primera parte consiste en el oráculo, que conoce la forma de la solución y va a marcar los elementos buscados, y la segunda parte es el difusor, que va a rotar el estado del circuito para aproximarlo al estado buscado. Estas dos partes se describen a continuación.

El Oráculo

En primer lugar, se quiere un operador unitario de n qubits que otorgue las siguiente salida para dos elementos $|x_0\rangle, |y_0\rangle$:

$$\mathcal{O}|x\rangle = \begin{cases} |x\rangle & \text{si } |x\rangle \neq |x_0\rangle \, ó \, |y_0\rangle \\ -|x\rangle & \text{si } |x\rangle = |x_0\rangle \, ó \, |y_0\rangle \, . \end{cases}$$

Para implementar el oráculo y buscar los estados seleccionados, es necesario aplicar compuertas X a los estados input, pero sólo a los qubits necesarios con tal de transformar el estado $|x_{input}\rangle$ en el estado $|11\ldots1\rangle$ si $|x_{input}\rangle$ era $|x_0\rangle$ ó $|y_0\rangle$. Luego, se utiliza una compuerta multicontrolada Z (MCZ), que introduce una fase de -1 únicamente si el estado del sistema coincide con $|11\ldots1\rangle$. Esta operación equivale a aplicar una fase de -1 exclusivamente sobre el estado $|x_0\rangle$. Finalmente, se revierten las compuertas X aplicadas al inicio, restaurando el sistema a su estado original salvo la fase cambiada. Para seleccionar varios elementos a la vez, en el mismo oráculo, basta con aplicar las compuertas necesarias de manera secuencial. Esto se puede ver el la figura 1.

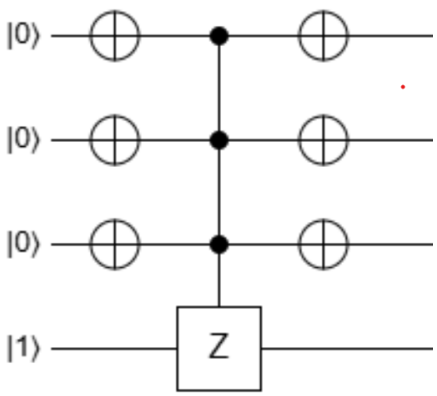

Figura 1: Ilustración del oráculo actuando sobre el estado $|0001\rangle$. Esta operación agregará una fase únicamente para este estado.

Operador difusor

En esta parte del algoritmo lo que se busca hacer es una reflexión alrededor del estado $|s\rangle$, donde $|s\rangle$ es el estado de superposición obtenido previamente. La reflexión que se realiza está dada por la siguiente operación:

$$\mathcal{D} = 2\,|s\rangle\,\langle s| - \mathbf{I},$$

Para construir el operador $\mathcal{D}$, se aprovechó que este actúa no trivialmente sólo sobre el estado $|0\rangle^{\otimes n}$. En consecuencia, se aplicaron compuertas H a todos los qubits, transformando el estado $|s\rangle$ al estado $|0\rangle^{\otimes n}$. Luego, se aplicaron compuertas X, para mapear el estado $|0\rangle^{\otimes n}$ a $|1\rangle^{\otimes n}$. A continuación, se aplicó una compuerta multicontrolada Z para cambiar la fase del estado. Se volvió a aplicar las compuertas X y H para recuperar el estado original. Este circuito se puede ver en la figura 2.

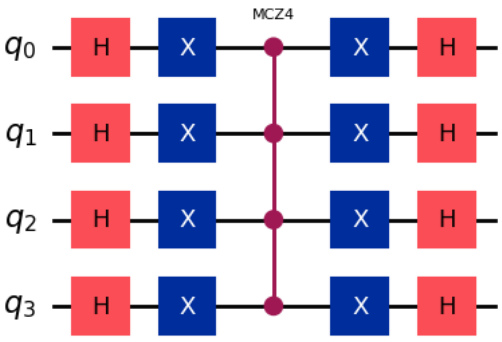

Figura 2: Esquematización del circuito de la compuerta difusión aplicada al $|q_1 q_2 q_3 q_4\rangle$

Pseudocódigo

A continuación se dará el pseudocódigo del algoritmo para encontrar 2 datos $|x_0\rangle, |y_0\rangle \in \{0,1\}^n$.

1. Como input: $|\psi_0\rangle = |0\rangle^{\otimes n}$
2. Se obtiene el estado en superposición mediante Hadamard

$$|\psi\rangle = H^{\otimes n}|0\rangle = \frac{1}{\sqrt{N}} \sum_{x\in\{0,1\}^n} |x\rangle$$

3. Se aplica el oráculo $\mathcal{O}$
4. Se aplica el operador difusor $\mathcal{D}$
5. Se repite la aplicación del oráculo y el difusor una cantidad k veces. Siendo $k = \frac{\pi}{4}\sqrt{N} - 1$.

Intuición geométrica del algoritmo y velocidad de búsqueda

Para visualizar cómo es el funcionamiento del algoritmo[3], se dividirá el espacio de Hilbert $\mathcal{H}$ en dos subespacios ortogonales, el espacio de vectores solución $\{|x_{0i}\rangle\}_{i=1}^{M}$ y el resto $\{|x_j\rangle\}_{j=1}^{N-M}$ con M la cantidad de estados a buscar y N la cantidad de estados finales posibles, y se definen los estados totalmente coherentes en estos subespacios como:

$$|\alpha\rangle = \frac{1}{\sqrt{N-M}} \sum_{j}^{N-M} |x_j\rangle$$

$$|\beta\rangle = \frac{1}{\sqrt{M}} \sum_{i}^{M} |x_{0i}\rangle$$

Luego de la primera aplicación de las compuertas de Hadamard, se puede demostrar que

$$|\psi\rangle = \sqrt{\frac{N+M}{N}}|\alpha\rangle + \sqrt{\frac{N}{M}}|\beta\rangle.$$

Por otro lado, tanto el oráculo como el operador inversión son rotaciones. Definiendo $cos(\frac{\theta}{2}) = \sqrt{\frac{N-M}{N}}$, se observa que

$$(\mathcal{IO})^k|\psi\rangle = cos(\frac{2k+1}{2}\theta)|\alpha\rangle + sin(\frac{2k+1}{2}\theta)|\beta\rangle$$

Por lo tanto, la aplicación seguida de estos dos operador provoca una rotación entre los estados que son solución y los que no. Puede observarse que la velocidad de búsqueda está asociada al ángulo θ, que crece con $\sqrt{N}$.

Por último, si se están buscando dos datos para un espacio de Hilbert de 4 qubits, $N = 16$, $M = 2$, el número de aplicaciones que hay que realizar este algoritmo es

$$R = \frac{\pi}{4}\sqrt{\frac{N}{M}} \approx 2{,}2.$$

Implementación

Para un sistema de $n = 4$ qubits en donde se querían encontrar los estados $|1110\rangle$ y $|1101\rangle$, siguiendo el pseudocódigo mencionado en la sección Algoritmo y usando la librería qiskit en python, se obtuvo el siguiente circuito.

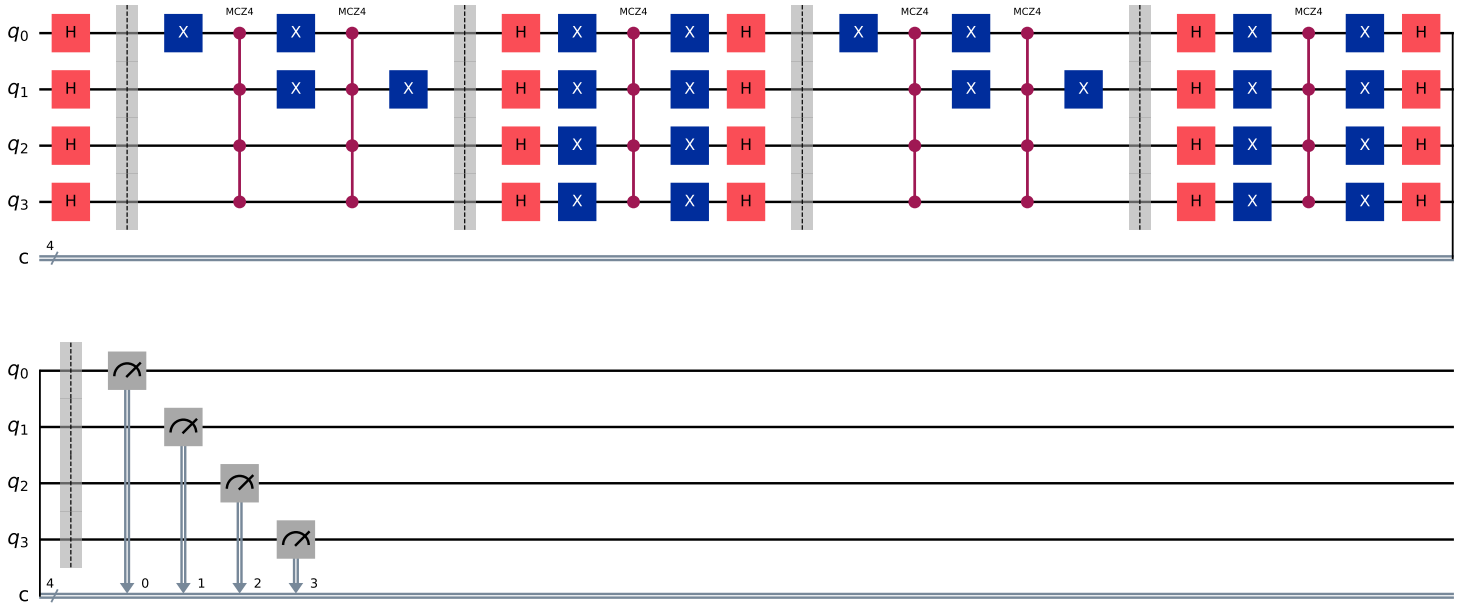

Figura 3: Circuito de la implementación del algoritmo de Grover. Al inicio se ven las compuertas de Hadamard aplicadas a todos los qubits. Luego una doble iteración del oráculo que marca dos elementos y el difusor que reflexiona los estados.

Se eligió un sistema de 4 qubits y un oráculo que marque dos elementos para tener un circuito más interesante a implementar. Luego de correr el algoritmo para distintos valores k de iteraciones, más allá de saber que el valor óptimo era $k = 2$, se obtuvieron los histogramas de las figuras 4 y 5. Dos aspectos a mencionar sobre las decisiones que tomamos a la hora de implementar el oráculo se vinculan con la cantidad de elementos que decidimos buscar y la cantidad de qubits que utilizamos para hacerlo. Por un lado, decidimos realizar una búsqueda de 2 elementos en lugar de únicamente 1 para aportar creatividad a la implementación (con respecto a la implementación tradicional). Por otro lado, una vez decidido realizaríamos una búsqueda de dos elementos, analizamos cuál era la cantidad de qubits más adecuada para este contexto. De haber elegido utilizar únicamente un qubit, el problema hubiera sido reducido a una trivialidad en la que la probabilidad de encontrar cualquiera de los dos elementos sería $\frac{1}{2}$. El próximo caso a analizar fue el de utilizar 2 qubits. De haberlo hecho, se hubiera condenado el trabajo a una trivialidad similar a la mencionada en el caso de utilizar 1 qubit: en total se tienen 4 estados distintos posibles y, estando en busca de 2 elementos, se trata de un problema de búsqueda en la mitad del espacio. Estas conclusiones nos llevaron a intentar con 3 qubits. Sin embargo, esta decisión tampoco nos llevó a buen puerto o más bien, nos hizo desconfiar de la precisión, quizás poco realista, de los resultados. En la iteración óptima para 3 qubits, la segunda

iteración, el algoritmos encontraba ambos elementos con probabilidad 1, y la probabilidad de encontrar cualquiera de los otros era exactamente 0. Desconfiando de este resultado fue que decidimos implementar 4 qubits y así obtuvimos los sistemas aquí adjuntos, y los resultados de la iteración óptima (en este caso también la segunda) fueron más realistas, como puede observarse en los histogramas: si bien los elementos buscados tienen una probabilidad más alta de ser encontrados que los demás, todos aparecen con una probabilidad $>$ (estricta) a 0.

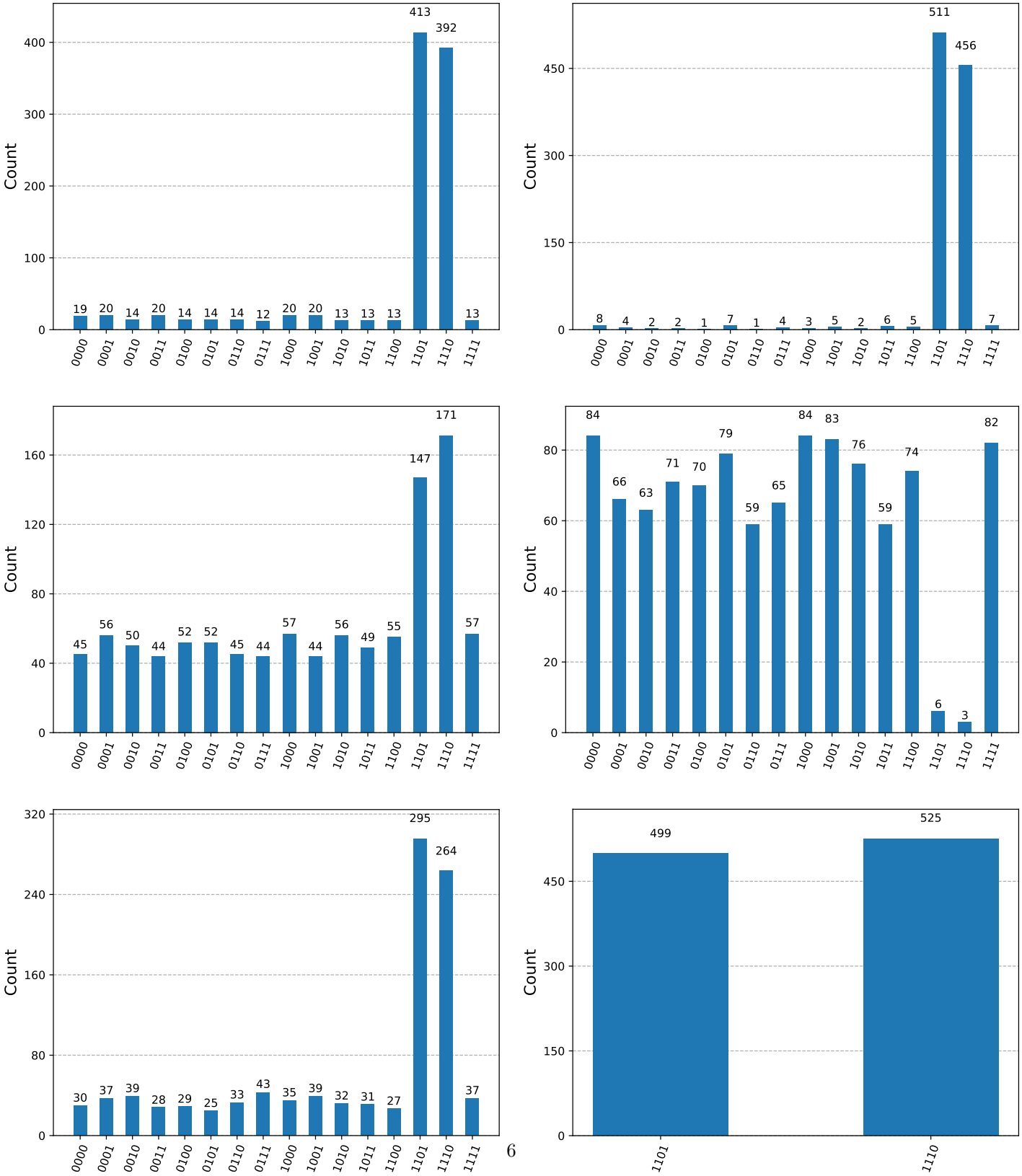

Figura 4: De izquierda a derecha están los histogramas para $k = 1, 2, 3, 4, 5$ y 6 iteraciones por circuito.

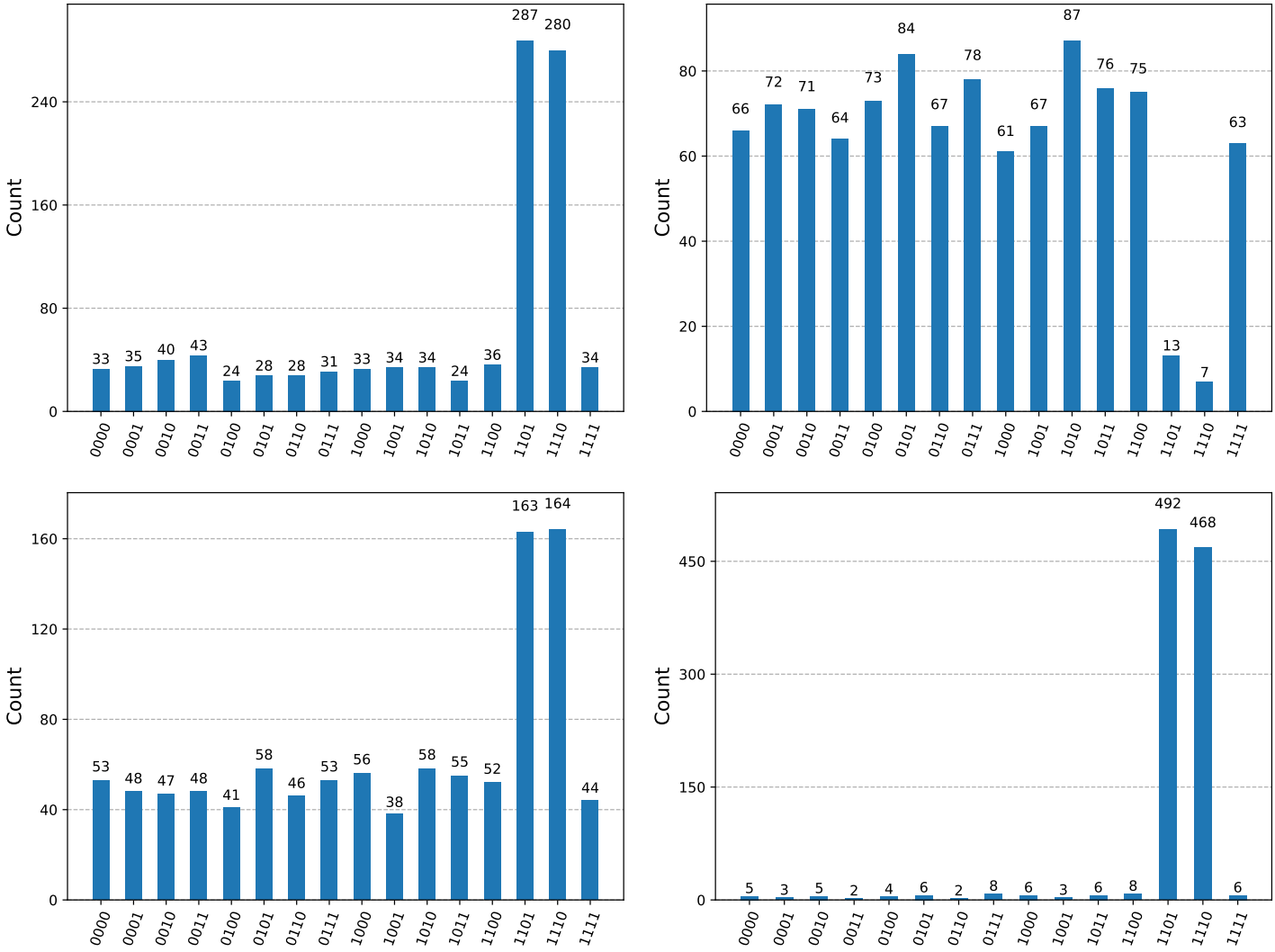

Figura 5: De izquierda a derecha se pueden ver los histogramas para $k = 7, 8, 9$ y 10.

Se puede observar cómo a medida que aumentamos las reflexiones y nos pasamos del valor óptimo, la probabilidad de encontrar los estados marcados disminuye respecto a la de encontrar otros estados.

Referencias

[1] Bennet, C et al., "The strengths and weaknesses of quantum computation", doi:10.1137/s0097539796300933, 1997

[2] Grover Lov K, .ᴬ fast quantum mechanical algorithm for database search", doi:10.1145/237814.237866, 1996.

[3] Nielsen Chuang, "Quantum Computation and Quantum Information", doi:10.1017/CBO9780511976667, 2010.