Universidad Intercontinental de la Empresa

Asignatura

Computación Cuántica y Natural

Actividad Práctica

Unidad I

Introducción a la Computación Cuántica y Natural

Sesión 02

La moneda perfecta

Profesor: Yago González Rozas

1 Objetivo de la actividad

El objetivo de la presente actividad es explicar el funcionamiento de la puerta de Hadamard, una de las puertas claves de la computación cuántica, y cómo se puede usar con PyQuil.

2 Resultados de aprendizaje relacionados

1. Conocer los fundamentos de los sistemas cuánticos y sus aplicaciones (RA01)
2. Identificar las posibles aplicaciones futuras de la computación cuántica y natural en las empresas (RA06)
3. Utilizar herramientas de software en el ámbito de la asignatura (RA07)

3 Descripción de la actividad

La puerta de Hadamard, o puerta H, es una puerta de cúbit único (aunque en lecciones posteriores veremos que puede verse amplificada para n-cúbits mediante operaciones matriciales) fundamental en la computación cuántica de cara a garantizar la generación de programas realmente cuánticos. Su importancia está basada en su capacidad para generar estados de superposición, es decir, partiendo de un estado |0> en una puerta cuántica, podremos pasar a un estado |0> y |1>, y viceversa con el estado |1>. A nivel formulación matricial, la puerta de Hadamard se representa como:

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

Si aplicamos la puerta de Hadamard sobre |0>:

$$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} |0\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle)$$

Si aplicamos la puerta de Hadamard sobre |1>:

$$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} |1\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix} = \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle)$$

El objetivo de la presente práctica es construir un sistema de moneda perfecta cuántica en tres fases:

1. Construimos el sistema simple, que arte de un solo cúbit sobre el que aplica la puerta de Hadamard y se realiza una medida sobre ella. Si repetimos la iteración en diferentes ejecuciones, el sistema debería devolver cara / cruz en una función aleatoria. Necesitaremos / queremos adaptar varias partes del programa base de las S01:
    a. El sistema solo debe ejecutarse una vez, por lo que el número de iteraciones de ejecución debería ser una. ¿Es posible hacerlo de otra forma?
    b. Nos interesa que la estructura lógica del circuito se identifique de la forma más simple posible, por lo que usaremos funciones para realizar la declaración de la memoria del programa y la realización de las medidas. En el primer caso, usaremos la función declare para construir un almacenamiento clásico de registros, y en el segundo usaremos la función measure para realizar la medida del cúbit y almacenar el resultado en el registro clásico.
    c. El sistema debe imprimir si es cara (valor 0) y cruz (valor 1) en función del resultado, que debe imprimirse también por pantalla.

2. Supongamos ahora que tenemos una competición entre dos usuarios A (gana cara) y B (gana cruz). En ella, van a lanzar una moneda al aire 50 veces, y nos interesa saber cuál de ellos gana. ¿Cómo debemos modificar el programa anterior para añadir el número de iteraciones e indicar quién gana? ¿Cómo iteramos sobre el resultado? ¿Qué pasa si en vez de modificar el número de iteraciones ejecuto múltiples veces el programa (qvm.run)?
3. Ahora, vamos a extender el programa, participando los mismos dos competidores, pero con cuatro monedas distintas, lanzando las monedas 50 veces. Nos interesa conocer las medidas de todas las monedas para declarar al ganador. Para ello usaremos dos técnicas distintas, primero modificando el registro para almacenar los cuatro valores por separado, y posteriormente usando la función measure_all. ¿Cómo modificarías el programa en ambos casos?