Q Battleship

QuantumTarget

Primera Escuela de Computación Cuántica

Vista general

1. Reglas de juego

2. Algoritmos cuánticos

Reglas de juego

- Es una partida de battleship de un jugador vs otro jugador.
- Se eligen los barcos poniendo las coordenadas de un arreglo de 4x4.
- Se jugará por turnos eligiendo las coordenadas ingresadas.
- Si se le agrega una Q junto a la coordenada se da la opción de movimiénto cuántico.
- Gana el jugador que logre derribar los 4 barcos enemigos primero.

Movimiento cuántico

El movimiento cuántico permite tener un turno extra

Advertencia

También puede hacer que pierdas el turno.

Coordenadas del arreglo

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Cuadro: Las coordenadas se colocan como números del 1 al 16

Algoritmos cuánticos

Primero, se colca el qubit 0 en superposición de los estados $|0\rangle$ y $|1\rangle$ para luego entrelazarlo con los otros $n-1$ qubits. Así, se obtienen 2^n cadenas de 0s y 1s que luego son convertidos a decimal para obtener un arreglo desordenado de números entre 1 y 2^n. Este algoritmo se usa para escoger la casilla a la que se atacará después de haber obtenido un 1 en el movimiento cuántico.

```
def Qtarget_n(n):\\
    device_backend = FakeYorktown()
    sim_yorktown = AerSimulator.from_backend(device_backend)
    N_QUBITS = n
    circ = QuantumCircuit(N_QUBITS, N_QUBITS)
    circ.h(0)
    for idx in range(N_QUBITS - 1):
        circ.cx(idx, idx + 1)
    circ.measure_all()
```

```
tcirc = transpile(circ, sim_yorktown)
result_noise = sim_yorktown.run(tcirc, shots=1024).result()
counts_noise = result_noise.get_counts(0)
lcounts=list(counts_noise.keys())
for i in range(len(lcounts)):
    lcounts[i]=str(bin2dec(lcounts[i].replace(" 0000",""))+1)
return(lcounts)
```

Algoritmos cuánticos

Se utiliza el estado de 3 qubits, conocido como estado GHZ. Que tiene probabilidad igual de estar en dos estados $|000\rangle$ y $|111\rangle$. A través de este se puede obtener una cadena solo de 0s o solo de 1s, así, se puede obtener un algoritmo que de un 1 o un 0. Este se usa para el movimiento cuántico y decide si se obtiene un turno extra o se pierde uno.

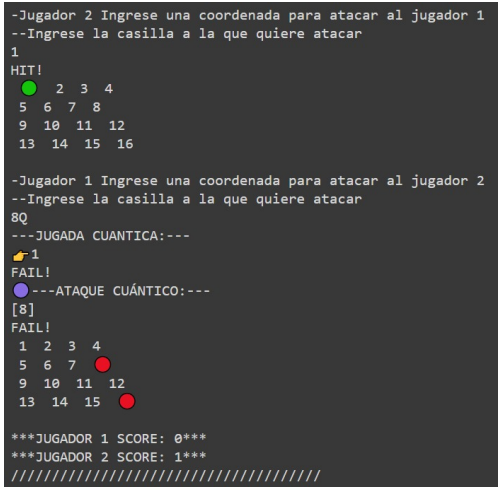
```
def Qtarget_():
    circ_ghz = QuantumCircuit(3,3)
    circ_ghz.h(0)
    circ_ghz.cx(0, 1)
    circ_ghz.cx(0, 2)
    circ_ghz.barrier()
    circ_ghz.measure([0,1,2], [0,1,2])

    circ_ghz.draw('mpl')
```

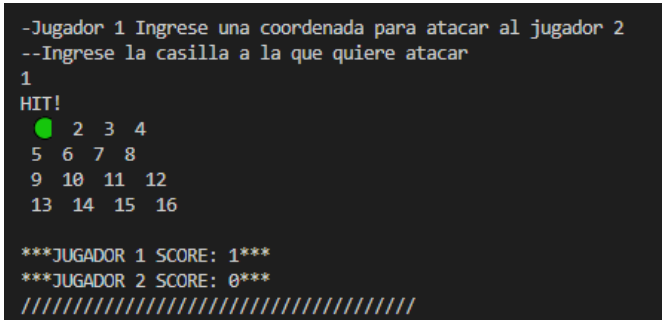
```
simulator = Aer.get_backend('aer_simulator')

job = execute(circ_ghz, simulator, shots=1)
counts = job.result().get_counts(circ_ghz)
r=23
if '1' in list(counts.keys())[0]:
    r=1
else:
    r=0
return r
```

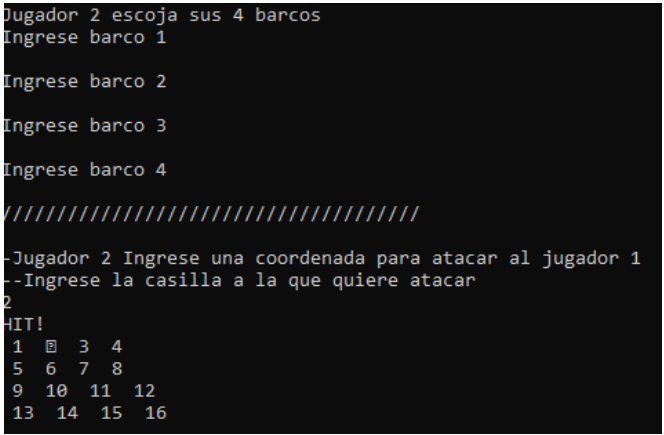
Q-Battleship en acción



Q-Battleship en VSC



Q-Battleship en consola



Game Over