Grover's Algorithm

August 10, 2021

1 Grover´s Algorithm for two qubits

Grover's Search Algorithm is a quantum algorithm that searches through unordered data. The algorithm, however, like all quantum, does not output the location with 100% precision but gives a very high probability of finding the element in a certain position. It's proven that Grover's Search is $\sqrt{X}$ faster than classical computer, where X is the number of elements in the database.

[1]:
```
#import libraries

from qiskit import QuantumCircuit, execute, Aer, IBMQ, transpile
from qiskit.visualization import plot_histogram
from qiskit_textbook.tools import array_to_latex
from qiskit.providers.ibmq import least_busy
from qiskit.aqua.algorithms import Grover
from qiskit.aqua.components.oracles import LogicalExpressionOracle,␣
↪TruthTableOracle
from qiskit.utils import QuantumInstance

print("Libraries imported successfully.")
```

```
Libraries imported successfully.
```

```
/opt/conda/lib/python3.8/site-packages/qiskit/aqua/__init__.py:86:
DeprecationWarning: The package qiskit.aqua is deprecated. It was
moved/refactored to qiskit-terra For more information see
<https://github.com/Qiskit/qiskit-aqua/blob/main/README.md#migration-guide>
  warn_package('aqua', 'qiskit-terra')
```

1.1 Creating an equal superposition

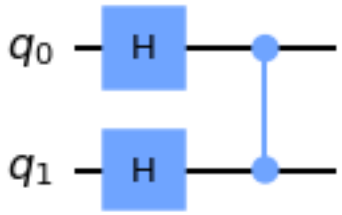
[2]:
```
# Define a quantum circuit with 2 qubits and create an equal superposition

qc = QuantumCircuit(2)
qc.h(0)
qc.h(1)
```

[2]:
```
<qiskit.circuit.instructionset.InstructionSet at 0x7f13e089acd0>
```

1.2 Applying the Gate

[3]:
```
# Apply the CZ Gate

qc.cz(0,1)
qc.draw()
```

[3]:



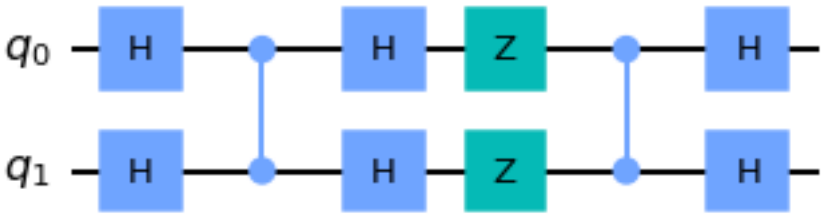
[4]:
```
# Check to make sure we got the right statevector

svsim = Aer.get_backend('statevector_simulator') # Tell it which simulator you␣
↪want to use
job = execute(qc,svsim) # Put in the name of your quantum circuit where it says␣
↪qc
result = job.result()
state = result.get_statevector()
array_to_latex(state, pretext="\\text{Statevector} = ")
```

$$\text{Statevector} = \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ -\frac{1}{2} \end{bmatrix}$$

1.3 Applying Grover's diffusion operator

[5]:
```
# BLOCK 4 - Applying the diffusion operator

qc.h(0)
qc.h(1)
qc.z(0)
qc.z(1)
qc.cz(0,1)
qc.h(0)
qc.h(1)
qc.draw()
```

[5]:



This circuit can be looped many-many times if there are a lot of items to search through to get a higher precision.

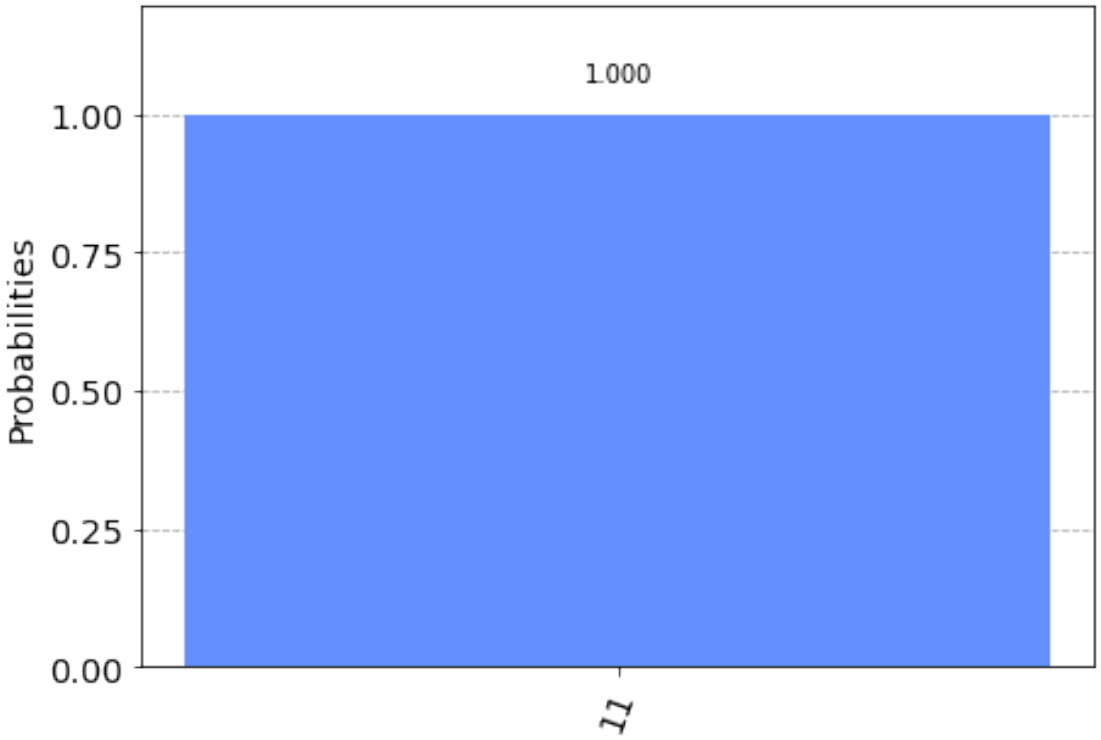
[6]:
```
# Find the statevector to make sure all the gates are implemented correctly

svsim = Aer.get_backend('statevector_simulator') # Tell it which simulator you␣
↪want to use
job = execute(qc,svsim) # Put in the name of your quantum circuit where it says␣
↪qc
result = job.result()
state = result.get_statevector()
array_to_latex(state, pretext="\\text{Statevector} = ")
```

$$\text{Statevector} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

1.4 Make a measurement

[7]:
```
# Use the QASM simulator to find the output

qc.measure_all()
svsim = Aer.get_backend('qasm_simulator') # Change statevector to qasm
job = execute(qc,svsim,shots=100) # add shots - tell it how many times to run
result = job.result()
counts = result.get_counts(qc)
plot_histogram(counts)
```

[7]:



1.5 Running Grover's Algorithm for many items

[8]:
```
# Define the list of items

items = '0000000010000000'
```

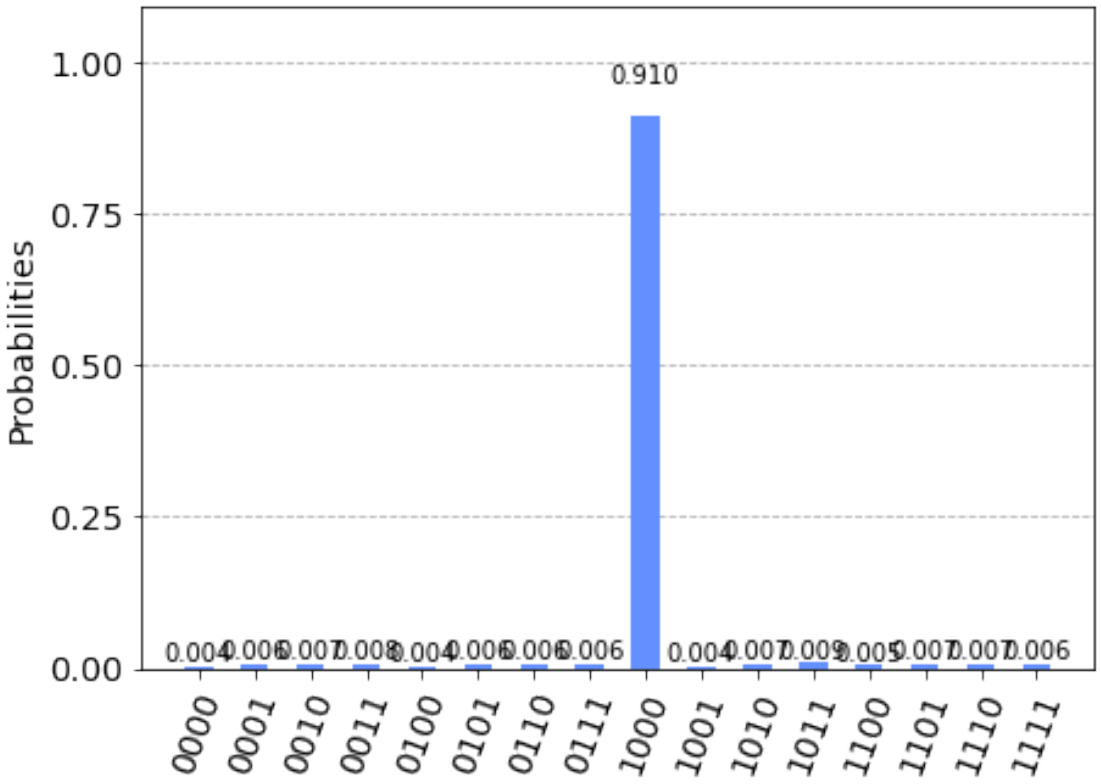
[9]:
```
# Create the oracle (built-in function for Qiskit)

oracle = TruthTableOracle(items)
```

[10]:
```
# Call the Grover function and visualize result

grover = Grover(oracle, iterations = 2)

result = grover.run(QuantumInstance(Aer.get_backend('qasm_simulator'),␣
↪shots=1024))
plot_histogram(result['measurement'])
```

```
/opt/conda/lib/python3.8/site-
packages/qiskit/aqua/algorithms/amplitude_amplifiers/grover.py:215:
DeprecationWarning: The package qiskit.aqua.algorithms.amplitude_amplifiers is
deprecated. It was moved/refactored to qiskit.algorithms.amplitude_amplifiers
(pip install qiskit-terra). For more information see
```

```
<https://github.com/Qiskit/qiskit-aqua/blob/main/README.md#migration-guide>
  warn_package('aqua.algorithms.amplitude_amplifiers',
```

[10]:



[11]:
```
The histogram above shows us the "1" state we wanted to find.

Brought up by Veronika Kitsul
```