grover

May 28, 2022

[ ]:
```
#initialization
import matplotlib.pyplot as plt
import numpy as np
import math

# importing Qiskit
from qiskit import IBMQ, Aer, assemble, transpile
from qiskit import QuantumCircuit, ClassicalRegister, QuantumRegister
from qiskit.providers.ibmq import least_busy

# import basic plot tools
from qiskit.visualization import plot_histogram
from qiskit_textbook.tools import vector2latex
```

[ ]:
```
n = 4
```

[ ]:
```
def apply_oracle(qc):
    # Oracle
    qc.h([2,3])
    qc.ccx(0,1,2)
    qc.h(2)
    qc.x(2)
    qc.x([1,3])
    qc.ccx(0,2,3)
    qc.x(2)
    qc.h(3)
    qc.x([1,3])
    qc.h(2)
    qc.mct([0,1,3],2)
    qc.x([1,3])
    qc.h(2)
    qc.x(2)

    return qc
```

[ ]:
```
def diffuser(nqubits):
    qc = QuantumCircuit(nqubits)
    # Apply transformation |s> -> |00..0> (H-gates)
```

```
    for qubit in range(nqubits):
        qc.h(qubit)
    # Apply transformation |00..0> -> |11..1> (X-gates)
    for qubit in range(nqubits):
        qc.x(qubit)
    # Do multi-controlled-Z gate
    qc.h(nqubits-1)
    qc.mct(list(range(nqubits-1)), nqubits-1)  # multi-controlled-toffoli
    qc.h(nqubits-1)
    # Apply transformation |11..1> -> |00..0>
    for qubit in range(nqubits):
        qc.x(qubit)
    # Apply transformation |00..0> -> |s>
    for qubit in range(nqubits):
        qc.h(qubit)
    # We will return the diffuser as a gate
    U_s = qc.to_gate()
    U_s.name = "U$_s$"
    return U_s
```

[ ]:
```
def initialize_s(qc, qubits):
    """Apply a H-gate to 'qubits' in qc"""
    for q in qubits:
        qc.h(q)
    return qc
```

[ ]:
```
grover_circuit = QuantumCircuit(n)
all_qubits = list(range(n))
grover_circuit = initialize_s(grover_circuit,all_qubits)
grover_circuit = apply_oracle(grover_circuit)
grover_circuit.append(diffuser(n), all_qubits)
grover_circuit.measure_all()
grover_circuit.draw()
```

[ ]:
```
                                                                     »
   q_0:  H                             0          M »
                                                                »
   q_1:  H         X   X              X     1              »
                                          U$_s$          »
   q_2:  H   H   X   H   X      X   H   X   H   X   2             »
                                                       »
   q_3:  H   H   X         X   H   X      X     3             »
                                                           »
meas: 4/                                            »
                                                                            0 »
«
«   q_0:
```

```
«
«   q_1:  M
«
«   q_2:    M
«
«   q_3:      M
«
«meas: 4/
«          1  2  3
```

[ ]:
```python
aer_sim = Aer.get_backend('aer_simulator')
transpiled_grover_circuit = transpile(grover_circuit, aer_sim)
qobj = assemble(transpiled_grover_circuit)
results = aer_sim.run(qobj).result()
counts = results.get_counts()
print(counts)
plot_histogram(counts)
```

```
{'0110': 3, '0111': 2, '0010': 3, '1001': 4, '0000': 5, '0101': 5, '0001': 3,
'0011': 2, '1101': 7, '1111': 3, '0100': 3, '1100': 175, '1000': 190, '1110':
197, '1010': 208, '1011': 214}
```

[ ]: