CSCI 4140 Laboratory Five

Grover's Algorithm

Introduction

The video lecture for Grover's algorithm presented two examples where only one iteration of the algorithm was required to find the item. In this laboratory we will examine the case where more than one iteration is required. In addition, we will use some more sophisticated oracles. You may need to update your Qiskit textbook code in order to access the oracle. Instructions for doing this are below.

Upgrade Qiskit Textbook Code

I found that I had to upgrade my Qiskit textbook code in order to get the Grover problem oracle. You can do this in the following way:

python -m pip install --upgrade git+https://github.com/qiskit-community/qiskit-textbook.git#subdirectory=qiskit-textbook-src

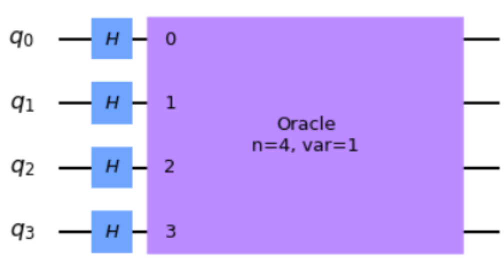
Again, you may need to do this as administrator.

Grover Test Oracle

The Qiskit textbook provides a function that produces oracles that can be used with Grover's algorithm. This function provides both the circuit and the solution for the oracle. This function can be used in the following way:

```
from qiskit_textbook.problems import grover_problem_oracle
## Example Usage
n = 4
oracle = grover_problem_oracle(n, variant=1, print_solutions = True)  # 0th variant of oracle, with n qubits
qc = QuantumCircuit(n)
qc = initialize_s(qc, [0,1,2, 3])
qc.append(oracle, [0,1,2,3])
qc.draw('mpl')
```

```
Solutions:
|0111>
|1001>
```



Let's put this version of the oracle into Grover's algorithm and see how it behaves. Since we are using 4 qubits and there are two solutions, we only need one iteration of the algorithm. From the lecture of Grover's algorithm, we have the following two functions that are used to implement the algorithm:

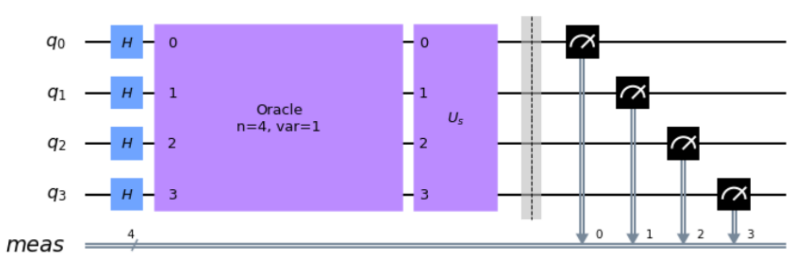
```
def initialize_s(qc, qubits):
    """Apply a H-gate to 'qubits' in qc"""
    for q in qubits:
        qc.h(q)
    return qc

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
```

With these functions we can put together the following code to find the solution:

```
from qiskit_textbook.problems import grover_problem_oracle
## Example Usage
n = 4
oracle = grover_problem_oracle(n, variant=1, print_solutions = True)  # 0th variant of oracle, with n qubits
qc = QuantumCircuit(n)
qc = initialize_s(qc, [0,1,2, 3])
qc.append(oracle, [0,1,2,3])
qc.append(diffuser(n), [0,1,2, 3])
qc.measure_all()
qc.draw('mpl')
```

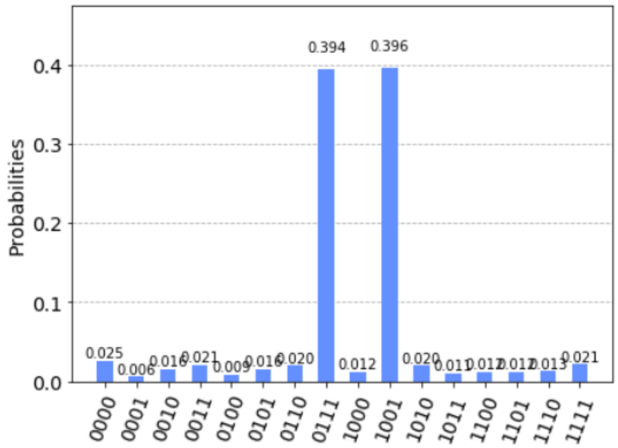
When we run this code we get the following result:

```
Solutions:
|0111>
|1001>
```



The next step is to get a backend and simulate the circuit. We will use the qasm simulator since it gives the probabilities of all the potential solution. The code and results are:

```
backend = Aer.get_backend('qasm_simulator')
results = execute(qc, backend=backend, shots=1024).result()
answer = results.get_counts()
plot_histogram(answer)
```

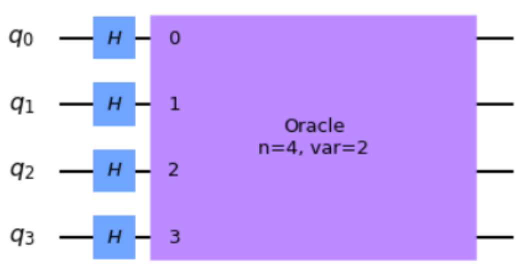


Try this code and make sure that you get the correct results. We will be reusing some of this code in the rest of the lab.

Multiple Iterations

We are still just doing one iteration of the algorithm. We need to have an oracle that requires multiple iterations. By changing the variant to 2, we get the following oracle

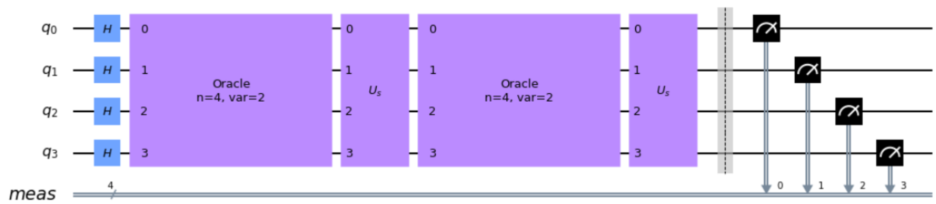
```
Solutions:
|0011>
```



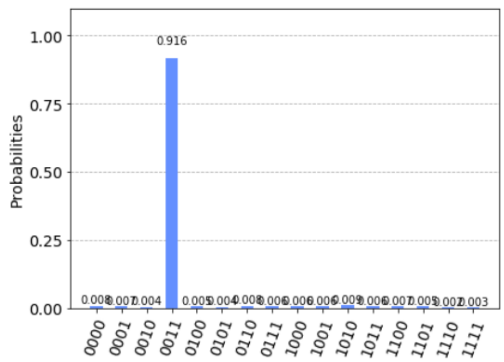
Since this is 4 qubits and one solution this will require two iterations of Grover's algorithm. This requires a slight change to our code where the oracle and diffuser are repeated. This gives us the following:

```python
from qiskit_textbook.problems import grover_problem_oracle
## Example Usage
n = 4
oracle = grover_problem_oracle(n, variant=2, print_solutions = True)  # 0th variant of oracle, with n qubits
qc = QuantumCircuit(n)
qc = initialize_s(qc, [0,1,2, 3])
qc.append(oracle, [0,1,2,3])
qc.append(diffuser(n), [0,1,2, 3])
qc.append(oracle, [0,1,2,3])
qc.append(diffuser(n), [0,1,2, 3])
qc.measure_all()
qc.draw('mpl')
```

```
Solutions:
|0011>
```



When we simulate this circuit we get the following:



Run this code and check that you get the correct result.

Laboratory Activity

The first activity is to use variant=0. You will see that this variant has 1 solution so you will need to use 2 iterations of the algorithm. Cut and Paste the circuit and your plot and add them to your report.

For the second activity create an oracle with 5 qubits and variant 2. This oracle has one solution with 5 qubits. We have $\sqrt{5} \approx 2.24$, so 2 iterations probably won't work. Give this oracle a try with 2 iterations to see if that is enough. If not, go to 3 iterations. Again, cut and paste the circuit and plot into your report.

Submit the report through Canvas in the form of a PDF or PNG.