CSCI Laboratory Nine

Satisfiability and Grover's Algorithm

Introduction

When using the Grover's algorithm implementation provided by Qiskit with the satisfiability problem I got some results I didn't expect. To further explore these results, I decided to go back to our own implementation of Grover's algorithm, which we have more control over. In this laboratory we will repeat the results of my experiments.

Logical Expression Oracle

I didn't want to build my own oracle for logical expressions, so I decided to continue using the one from Qiskit. We will also continue to use the same logical expression. To start with we need some import statements.

```
import numpy as np
from qiskit.visualization import plot_histogram
from qiskit.aqua.components.oracles import LogicalExpressionOracle, TruthTableOracle
from qiskit import Aer, QuantumCircuit, ClassicalRegister, QuantumRegister, execute
```

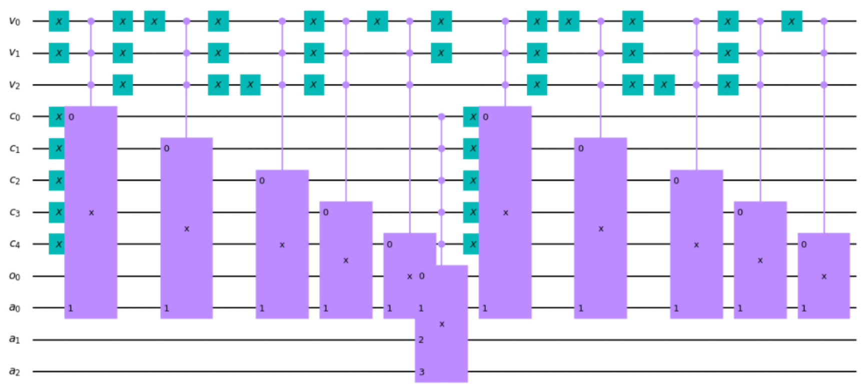
Next, we have the code for constructing the oracle:

```
input_3sat = '''
c example DIMACS-CNF 3-SAT
p cnf 3 5
-1 -2 -3 0
1 -2 3 0
1 2 -3 0
1 -2 -3 0
-1 2 3 0
'''

oracle = LogicalExpressionOracle(input_3sat)
oracle.construct_circuit()
orc = oracle.circuit
orc.draw('mpl')
```

There isn't very much new here. The main thing that's different is we call construct_circuit() to build the circuit for the logical expression. The circuit attribute will now contain the constructed circuit that we can retrieve using the circuit attribute.

When we print the circuit, we get the following, which is quite a complicated circuit:

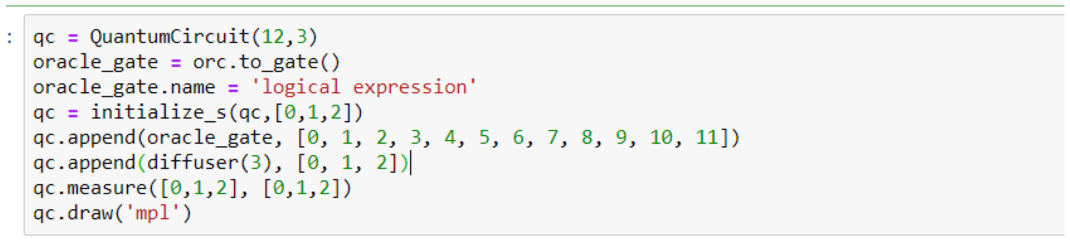


There is an optimization parameter that we could have used to make the circuit slightly smaller, but it doesn't make a lot of difference. Note that this circuit uses 12 qubits.

Grover's Algorithm

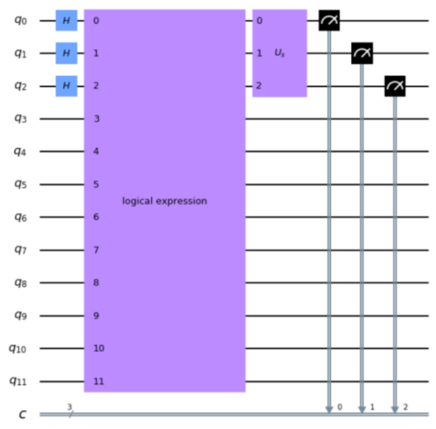
Now we are ready to use our version of Grover's algorithm. Go back to laboratory five and copy the code that you had for initialize_s and diffuser, they will be the starting point for the next part of the laboratory.

We will use the following Python code to combine our oracle with our code for Grover's algorithm:

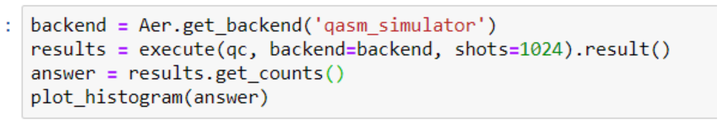


```
qc = QuantumCircuit(12,3)
oracle_gate = orc.to_gate()
oracle_gate.name = 'logical expression'
qc = initialize_s(qc,[0,1,2])
qc.append(oracle_gate, [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11])
qc.append(diffuser(3), [0, 1, 2])
qc.measure([0,1,2], [0,1,2])
qc.draw('mpl')
```

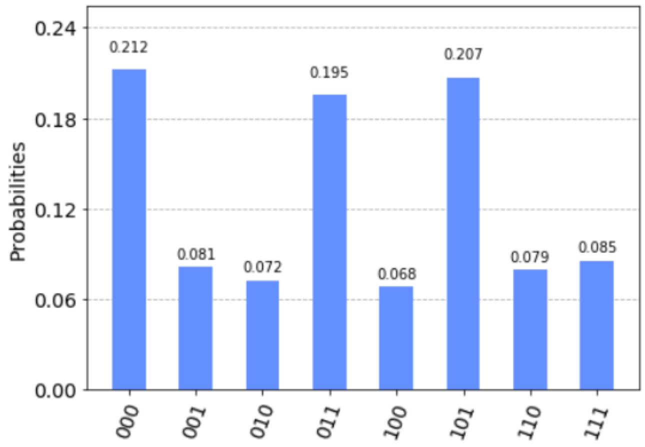
I've converted the oracle circuit into a gate so we can draw the circuit. I initially experimented with two clauses instead of the five in the current logical expression. This required fewer gates and Qiskit could draw the entire circuit, but with 5 clauses we seem to have exceeded its limit. This is what we get when we draw the circuit:



Now we can run the circuit and observe the results:



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



These results are basically the same as what we had with the Qiskit implementation of Grover's algorithm, so our code seems to be correct.

Laboratory Exercise

Now for the interesting part of the experiment. We have only done one iteration of Grover's algorithm. The results that we've got look reasonable. But what happens when we do more iterations? For your part of the laboratory try two and three iterations of Grover's algorithm. Remember increasing the number of iterations is just a matter of cutting and pasting two lines of code. Cut and paste the histograms that you produce into the report for this laboratory. What do you observe from this experiment? Submit your report as a PDF or PNG file.