In [22]:

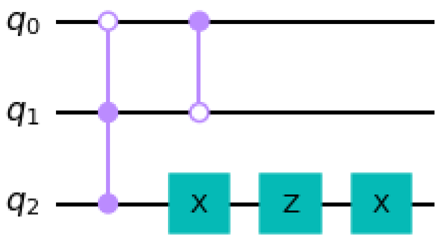
```
# Grover's Algorithm
# CPSC 4110
import numpy as np
from qiskit import Aer
from qiskit.visualization import plot_histogram
from qiskit.utils import QuantumInstance
from qiskit.algorithms import Grover, AmplificationProblem
from qiskit.circuit.library import PhaseOracle

with open('3sat.dimacs', 'r') as f:
    dimacs = f.read()
print(dimacs)

oracle = PhaseOracle.from_dimacs_file('3sat.dimacs')
oracle.draw()
```

```
c example DIMACS-CNF 3-SAT
p cnf 3 3
1 2 -3 0
-1 -2 -3 0
-1 2 3 0
```

Out[22]:



In [26]:

```
class Verifier():
    """Create an object that can be used to check whether
    an assignment satisfies a DIMACS file.
        Args:
            dimacs_file (str): path to the DIMACS file
    """
    def __init__(self, dimacs_file):
        with open(dimacs_file, 'r') as f:
            self.dimacs = f.read()

    def is_correct(self, guess):
        """Verifies a SAT solution against this object's
        DIMACS file.
            Args:
                guess (str): Assignment to be verified.
                             Must be string of 1s and 0s.
            Returns:
                bool: True if `guess` satisfies the
                           problem. False otherwise.
        """
        # Convert characters to bools & reverse
        guess = [bool(int(x)) for x in guess][::-1]
        for line in self.dimacs.split('\n'):
            line = line.strip(' 0')
            clause_eval = False
            for literal in line.split(' '):
                if literal in ['p', 'c']:
                    # line is not a clause
                    clause_eval = True
                    break
                if '-' in literal:
                    literal = literal.strip('-')
                    lit_eval = not guess[int(literal)-1]
                else:
                    lit_eval = guess[int(literal)-1]
                clause_eval |= lit_eval
            if clause_eval is False:
                return False
        return True

v = Verifier('3sat.dimacs')
v.is_correct('011')
```

Out[26]:

```
True
```

In [27]:

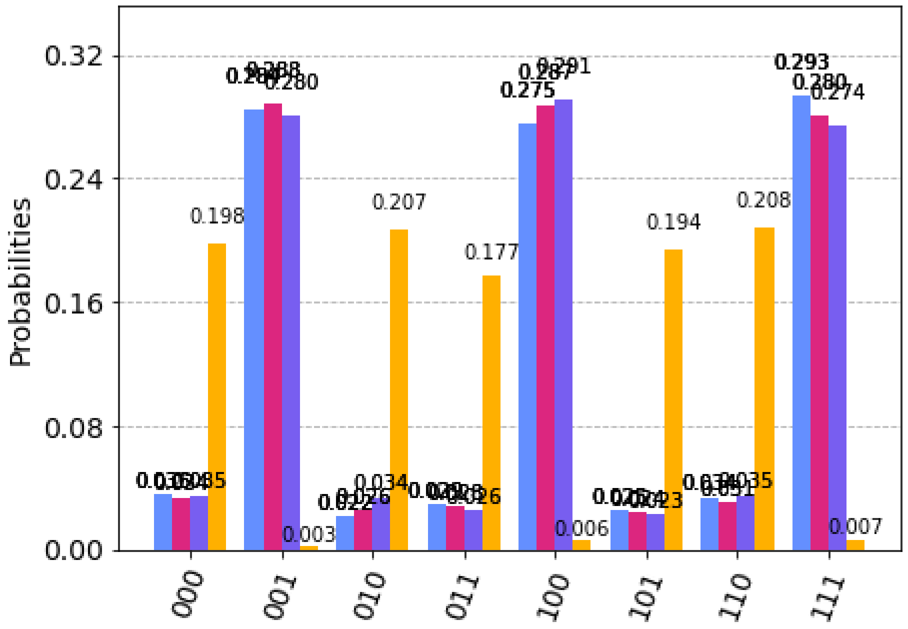
```
# Configure backend
backend = Aer.get_backend('aer_simulator')
quantum_instance = QuantumInstance(backend, shots=1024)

# Create a new problem from the phase oracle and the
# verification function
problem = AmplificationProblem(oracle=oracle, is_good_state=v.is_correct)

# Use Grover's algorithm to solve the problem
grover = Grover(quantum_instance=quantum_instance)
result = grover.amplify(problem)
result.top_measurement

plot_histogram(result.circuit_results)
```

Out[27]:



In [25]:

```
# Load our saved IBMQ accounts and get the ibmq_16_melbourne backend
from qiskit.test.mock import FakeMelbourne
melbourne = FakeMelbourne()

from qiskit.compiler import transpile

# transpile the circuit for ibmq_16_melbourne
qc = grover.construct_circuit(problem, max(result.iterations))
qc.measure_all()
grover_compiled = transpile(qc, backend=melbourne, optimization_level=3)

print('gates = ', grover_compiled.count_ops())
print('depth = ', grover_compiled.depth())
```

```
gates =  OrderedDict([('u3', 57), ('cx', 44), ('u2', 26), ('u1', 6), ('measur
e', 3), ('barrier', 1)])
depth =  89
```

In [ ]:

```

```