Practical 10

Name: Heja Bibani
Student Number: 16301173

In [94]:
```
from qiskit import *
from qiskit.providers.aer import QasmSimulator
from qiskit.visualization import plot_histogram
import numpy as np
from math import sqrt
%matplotlib inline
```

Section 1

Suppose $V = [12345678]^T$ is a column vector. You should use Python code to complete the following:

1.1

Construct the matrix $2AA - II$ as defined in the lecture. It should be of the shape 8 × 8.

In [95]:
```
x = np.array([[1,1,1,1,1,1,1,1],[1,1,1,1,1,1,1,1], [1,1,1,1,1,1,1,1], [1,1,1,1,1,1,1,1] , [1,1,1,1,1,1,1,1], [1
x

y = np.array([[1,0,0,0,0,0, 0,0],[0,1,0,0,0,0, 0,0], [0,0,1,0,0,0,0,0], [0,0,0,1,0,0,0,0] , [0,0,0,0,1,0,0,0],
y
V = np.array([[1],[2], [3],[4],[5],[6],[7],[8]])
V

MATRIX = (2*(1/8)*x-y)
MATRIX
```

Out[95]:
```
array([[-0.75,  0.25,  0.25,  0.25,  0.25,  0.25,  0.25,  0.25],
       [ 0.25, -0.75,  0.25,  0.25,  0.25,  0.25,  0.25,  0.25],
       [ 0.25,  0.25, -0.75,  0.25,  0.25,  0.25,  0.25,  0.25],
       [ 0.25,  0.25,  0.25, -0.75,  0.25,  0.25,  0.25,  0.25],
       [ 0.25,  0.25,  0.25,  0.25, -0.75,  0.25,  0.25,  0.25],
       [ 0.25,  0.25,  0.25,  0.25,  0.25, -0.75,  0.25,  0.25],
       [ 0.25,  0.25,  0.25,  0.25,  0.25,  0.25, -0.75,  0.25],
       [ 0.25,  0.25,  0.25,  0.25,  0.25,  0.25,  0.25, -0.75]])
```

1.2

Calculate $V_1 = (2A - I) \times V$

In [96]:
```
V1 = np.dot(MATRIX,V)
V1
```

Out[96]:
```
array([[8.],
       [7.],
       [6.],
       [5.],
       [4.],
       [3.],
       [2.],
       [1.]])
```

1.3

Apply the formula $2a - v$ defined in the lecture to each entry in V to obtain a new vector $V2$.

The average of the vector denoted by a is given by the following:

$a = \frac{(1+2+3+4+5+6+7+8)}{8} = 4.5$

Multiplying by 2 this equals:

$2(4.5) = 9$

Then we must calculate $9 - v$ (which is each of the entries in V).

We can determine this by doing the following:

In [97]:
```
V2  = 9 - V
V2
```

Out[97]:
```
array([[8],
       [7],
       [6],
       [5],
       [4],
       [3],
       [2],
       [1]])
```

1.4

Compare V_1 and V_2. Do they contain the same entries?

In [98]:
```
V1 == V2
```

Out[98]:
```
array([[ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True]])
```

Yes, as expected they do contain the same entries.

Section 2

Suppose in a Grover's circuit, the probability for the measuring outcome to be '101' is 0.945, and four independent measurements are conducted on this circuit. Then, what is the probability for obtaining '101' at least twice among these four measurements? (NB: This task should be done by Python code.)

This follows binomial theorem, n = 4 and since we are looking for the value of atleast 2, we are going to look at all values k >=2. Thus, three terms need to be added.

Binomial theorem uses:

$poutcome = \binom{n}{k} p^k \times (1-p)^{n-k}$

Where n = 4, p = 0.945. We need to add the values k >= 2:

$(\binom{4}{2}(0.945)^2 \times (0.055)^2) + (\binom{4}{3}(0.945)^3 \times (0.055)^1) + (\binom{4}{4}(0.945)^4 \times (0.055)^0)$

In [99]:
```
from scipy.stats import binom

term1 = binom.pmf(k=2, n=4, p=0.945)
term2 = binom.pmf(k=3, n=4, p=0.945)
term3 = binom.pmf(k=4, n=4, p=0.945)

p_outcome = term1 + term2 + term3
p_outcome
```

Out[99]:
```
0.9993619518750001
```

Section 3

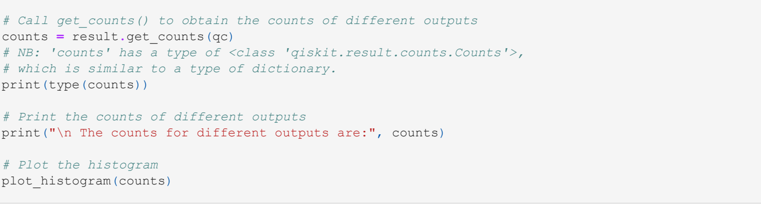
Suppose $f(x_1, x_0)$ is a 2-Boolean-variable function. It outputs 1 only when the input is $x_1 = 0$ and $x_0 = 0$, otherwise it outputs 0. In this task, you are asked to implement a Grover's circuit that reveals which input makes $f(x_1, x_0)$ output 1.

3.1

Roughly follow the notebook accompanying Lecture 10 to implement this circuit and conduct measurement. Run this circuit for at least 1000 shots and plot the result. In this situation the oracle must be different from F_1. We do this by implementing an x gate on both x_0 and x_1 and then restore this by implementing the x-gate again.

In [100…
```
# Construct a circuit with a 3-qubit input, and 2-bit measurement storage.
qc = QuantumCircuit(3, 2)

# Initialise q2 with |1>
initial_state = [0, 1]
qc.initialize(initial_state, 2)

# Apply H-gates to q0, q1 and q2
qc.h([0,1,2])
# Equivalent:
#qc.h(0)
#qc.h(1)
#qc.h(2)

# Insert the oracle F1
# And add two barriers to enclose this oracle.
qc.barrier()
qc.x(0)
qc.x(1)
qc.ccx(0,1,2)
qc.x(0)
qc.x(1)
qc.barrier()

# For 2A-I
# Apply H-gates to q0, q1
qc.h([0,1])
# Apply X-gates to q0, q1
qc.x([0,1])
# Apply CZ gate with q0 as control and q1 as target
qc.cz(0,1)
# Apply X-gates to q0, q1 again
qc.x([0,1])
# Apply H-gates to q0, q1 again
qc.h([0,1])

# Measure q0, q1
qc.barrier()
qc.measure([0,1], [0,1])

# Draw the circuit
qc.draw()
```

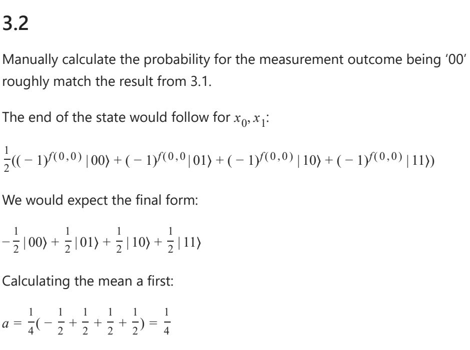
Out[100…



In [101…
```
simulator = QasmSimulator()
# Run the circuit with certain number of shots.
# The 'shots' below means the number of times to run this circuit.
# Its default value is 1024, so if it is omitted, you'll run the circuit 1024 times.
job = execute(qc, backend=simulator, shots=1000)

# Call result() to obtain the result
result = job.result()

# Call get_counts() to obtain the counts of different outputs
counts = result.get_counts(qc)
# NB: 'counts' has a type of <class 'qiskit.result.counts.Counts'>,
# which is similar to a type of dictionary.
print(type(counts))

# Print the counts of different outputs
print("\n The counts for different outputs are:", counts)

# Plot the histogram
plot_histogram(counts)
```

```
<class 'qiskit.result.counts.Counts'>

 The counts for different outputs are: {'00': 1000}
```

Out[101…



As expected $|00\rangle$ has been measured with 100% certainity.

3.2

Manually calculate the probability for the measurement outcome being '00' given the above circuit. The probability obtained should roughly match the result from 3.1.

The end of the state would follow for x_0, x_1:

$\frac{1}{2}((-1)^{f(0,0)}|00\rangle + (-1)^{f(0,0}|01\rangle + (-1)^{f(0,0)}|10\rangle + (-1)^{f(0,0)}|11\rangle)$

We would expect the final form:

$-\frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle$

Calculating the mean a first:

$a = \frac{1}{4}(-\frac{1}{2} + \frac{1}{2} + \frac{1}{2} + \frac{1}{2}) = \frac{1}{4}$

Flipping by the formula $2a - v$: $\frac{1}{2}$ becomes 0, $-\frac{1}{2}$ becomes 1.

Therefore after measurment we would expect $|00\rangle$ with 100% certainity.

Section 4

Suppose $f(x_2, x_1, x_0)$ is a 3-Boolean-variable function. It outputs 1 only when the input is $x_2 = 0, x_1 = 0$ and $x_0 = 0$, otherwise it outputs 0. In this task, you are asked to implement a Grover's circuit that reveals which input makes $f(x_2, x_1, x_0)$ output 1.

4.1

Roughly follow the notebook accompanying Lecture 10 to implement this circuit and conduct measurement. Run this circuit for at least 1000 shots and plot the result.

4. Constructing the Grover circuit with F_6 as oracle

As derived in the Lecture 10, the oracle F_6 can be achieved by an X-gate, an MCT gate, and an X-gate again. The $2A - I$ can be implemented by $H^{\otimes 3} \times X^{\otimes 3} \times CCZ \times X^{\otimes 3} \times H^{\otimes 3}$.

Moreover, we need to do the amplitude amplification twice to obtain a high-probability outcome when $n = 3$. The number of amplitude amplifications to perform (denoted by k) is calculated by the following formula:

$$k = \text{floor}\,(\frac{\pi}{4}\sqrt{2^n})$$

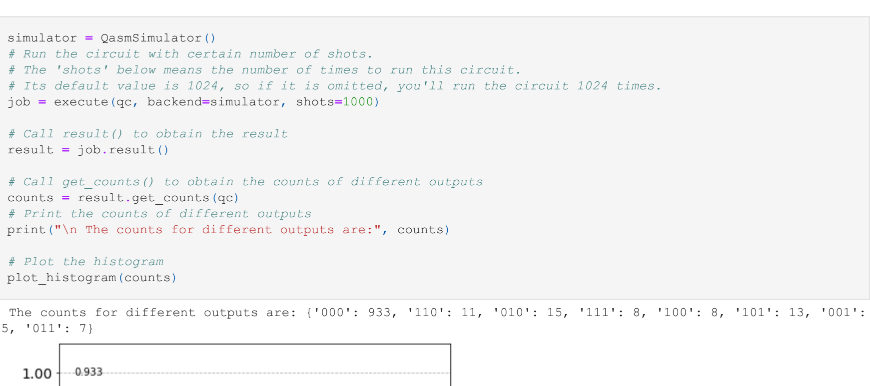
Where k = 2.

The oracle must also be different, we do this by implementing an x gate on both x_0, x_1 and x_2 and then restore this by implementing the x-gate again.

In [102…
```
# Needed by the construction of the CCZ gate
from qiskit.circuit.library import MCMT
import math

# Construct a circuit with a 4-qubit input, and 3-bit measurement storage.
qc = QuantumCircuit(4, 3)

# Apply H-gates to q0, q1, and q2
qc.h([0, 1, 2])

# Calculate the number of amplitude amplifications
k = math.floor(math.pi*math.sqrt(2**3)/4)
print("k=", k)

# Repeat k times
for i in range(k):
    # Initialise q3 with |1>
    initial_state = [0, 1]
    qc.initialize(initial_state, 3)
    # Apply H-gate to q3
    qc.h(3)

    # Insert the oracle F6
    # And add two barriers to enclose this oracle.
    qc.barrier()
    qc.x(0)
    qc.x(1)
    qc.x(2)
    qc.mct([0, 1, 2], 3)
    qc.x(0)
    qc.x(1)
    qc.x(2)
    qc.barrier()

    # Apply the gates for 2A-I to q0, q1 and q2
    # Apply H and X gates first
    qc.h([0, 1, 2])
    qc.x([0, 1, 2])
    # Realise the CCZ gate via MCMT circuit
    qc = qc.compose(MCMT('z', 2, 1))
    # Apply X and H gates again
    qc.x([0, 1, 2])
    qc.h([0, 1, 2])

    qc.barrier()

# Measure q0, q1, q2
qc.measure([0,1,2], [0,1,2])
# Draw the circuit
qc.draw()
```

```
k= 2
```

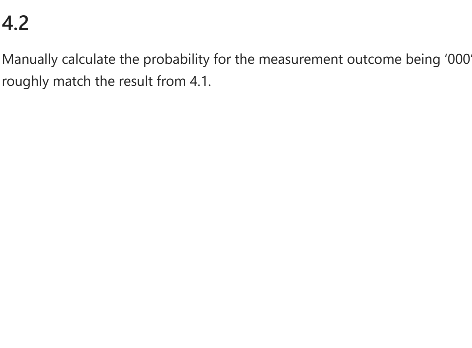
Out[102…



In [103…
```
simulator = QasmSimulator()
# Run the circuit with certain number of shots.
# The 'shots' below means the number of times to run this circuit.
# Its default value is 1024, so if it is omitted, you'll run the circuit 1024 times.
job = execute(qc, backend=simulator, shots=1000)

# Call result() to obtain the result
result = job.result()

# Call get_counts() to obtain the counts of different outputs
counts = result.get_counts(qc)
# Print the counts of different outputs
print("\n The counts for different outputs are:", counts)

# Plot the histogram
plot_histogram(counts)
```

```
 The counts for different outputs are: {'000': 933, '110': 11, '010': 15, '111': 8, '100': 8, '101': 13, '001':
5, '011': 7}
```

Out[103…



Thus as expected $|000\rangle$ will be seen with roughly 0.945 probability.

4.2

Manually calculate the probability for the measurement outcome being '000' given the above circuit. The probability obtained should roughly match the result from 4.1.

The end of the state would follow for x_0, x_1, x_2:

$$\frac{1}{2\sqrt{2}}(-|000\rangle + |001\rangle + |010\rangle + |011\rangle + |100\rangle + |101\rangle + |110\rangle + |111\rangle)$$

Let's apply flipping about mean to this state vector.

Calculating the mean 'a' first: $a = \frac{1}{8} \times \frac{1}{2\sqrt{2}} \times (7-1) = \frac{3}{8\sqrt{2}}$

Flipping by the formula $2a - v$: $\frac{1}{2\sqrt{2}}$ becomes $-\frac{1}{4\sqrt{2}}$; $-\frac{1}{2\sqrt{2}}$ becomes $\frac{5}{4\sqrt{2}}$

Measuring now, we'll get $|000\rangle$ with probability $(\frac{5}{4\sqrt{2}})^2 = 0.78$. This is not ideal, so let's go through flipping sign and flipping about mean for a second time.

With flipping the sign, we'll only flip the sign for $|000\rangle$, thus the state vector becomes:

$$\frac{1}{4\sqrt{2}}(-5|000\rangle + |001\rangle + |010\rangle + |011\rangle + |100\rangle + |101\rangle + |110\rangle + |111\rangle)$$

Let's apply flipping about mean to this state vector.

Calculating the mean 'a' first: $a = \frac{1}{8} \times \frac{1}{4\sqrt{2}} \times (7-5) = \frac{1}{16\sqrt{2}}$

Flipping by the formula $2a - v$: $\frac{1}{4\sqrt{2}}$ becomes $-\frac{1}{8\sqrt{2}}$; $-\frac{5}{4\sqrt{2}}$ becomes $\frac{11}{8\sqrt{2}}$.

Measuring now, we'll get $|000\rangle$ with probability $(\frac{11}{8\sqrt{2}})^2 = 0.945$. This matches the result above.