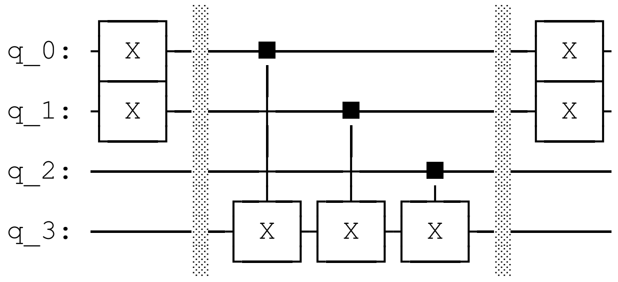
In [1]:
```
from qiskit import *
from qiskit.visualization import plot_histogram
IBMQ.load_account()
import numpy as np
%matplotlib inline
```

In [2]:
```
n=3
const=QuantumCircuit(n+1)
output = np.random.randint(2)
if output==1:
    const.x(n)
const.draw()
```

Out[2]:
```
q_0:

q_1:

q_2:

q_3:
```

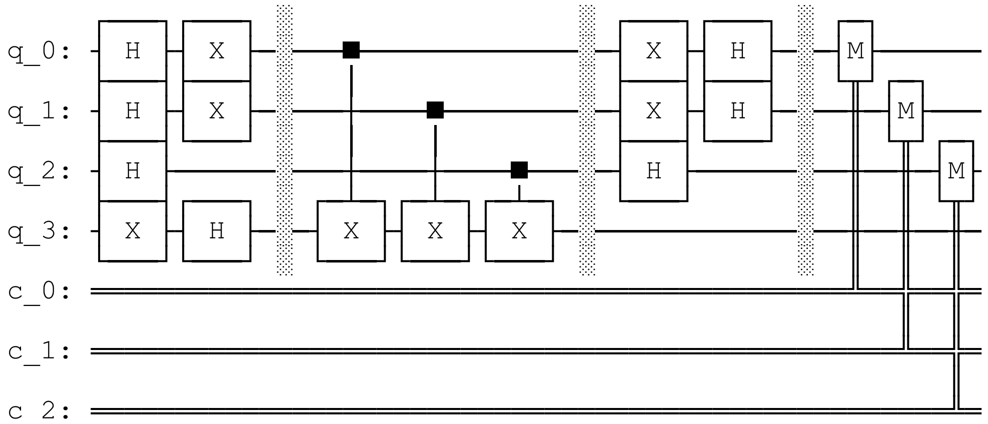
In [3]:
```
balance=QuantumCircuit(n+1)
b_start='110'

for qubit in range(len(b_start)):
    if b_start[qubit]=='1':
        balance.x(qubit)

balance.barrier()
for qubit in range(n):
    balance.cx(qubit,n)
balance.barrier()

for qubit in range(len(b_start)):
    if b_start[qubit]=='1':
        balance.x(qubit)

balance.draw()
```

Out[3]:



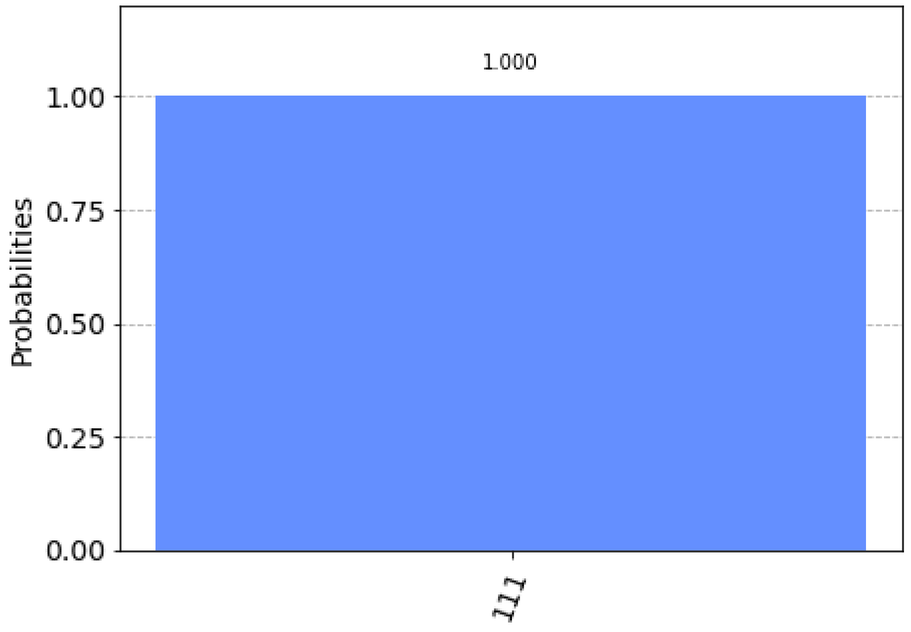
In [4]:
```
DJ_circuit= QuantumCircuit(n+1,n)
for qubit in range(n):
    DJ_circuit.h(qubit)

DJ_circuit.x(n)
DJ_circuit.h(n)

DJ_circuit +=balance
for qubit in range(n):
    DJ_circuit.h(qubit)
DJ_circuit.barrier()


for i in range(n):
    DJ_circuit.measure(i, i)


DJ_circuit.draw()
```

Out[4]:



In [5]:
```
backend = BasicAer.get_backend('qasm_simulator')
shots = 1024
results = execute(DJ_circuit, backend=backend, shots=shots).result()
answer = results.get_counts()

plot_histogram(answer)
```

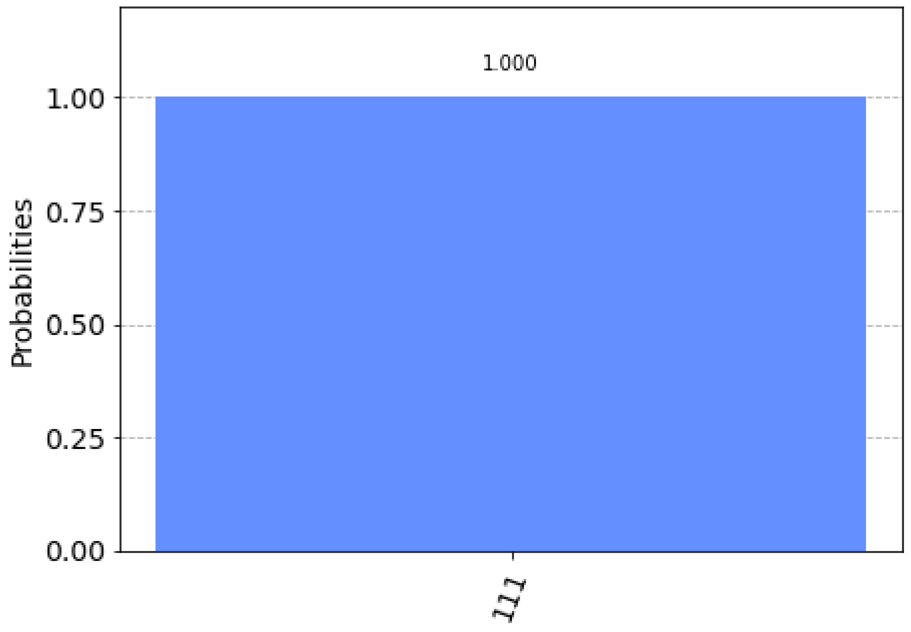
Out[5]:



In [6]:
```
provider= IBMQ.get_provider('ibm-q')
qcomp=provider.get_backend('ibmq_santiago')
job= execute(DJ_circuit,backend =backend,shots=1024)
from qiskit.tools.monitor import job_monitor
job_monitor(job)
result = job.result()
plot_histogram(result.get_counts(DJ_circuit))
```

```
Job Status: job has successfully run
```

Out[6]:



```
In [ ]:
```