Bernstein-Vazirani Algorithm

1. Initialization

In [1]:

```
import matplotlib.pyplot as plt
import numpy as np
from qiskit import IBMQ, Aer
from qiskit.providers.ibmq import least_busy
from qiskit import QuantumCircuit, ClassicalRegister, QuantumRegister, transpile, as

from qiskit.visualization import plot_histogram
```

2. Setup

In [2]:

```
n = 3 # number of qubits used to represent s
s = '011'   # the hidden binary string
```

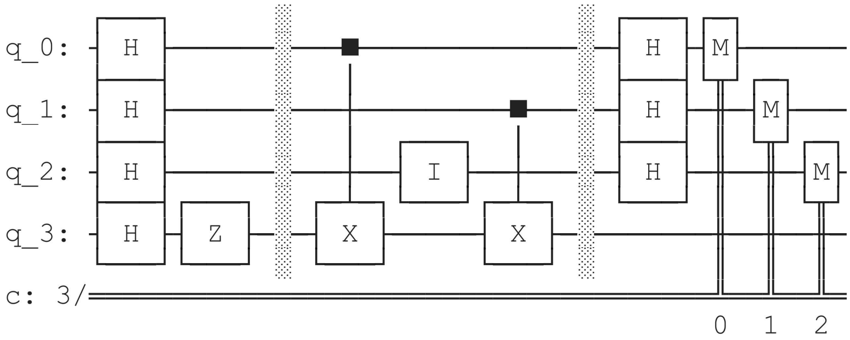
3. Algorithm

In [3]:

```
# We need a circuit with n qubits, plus one auxiliary qubit
# Also need n classical bits to write the output to
bv_circuit = QuantumCircuit(n+1, n)

# put auxiliary in state |->
bv_circuit.h(n)
bv_circuit.z(n)

# Apply Hadamard gates before querying the oracle
for i in range(n):
    bv_circuit.h(i)

bv_circuit.barrier()

# Apply the inner-product oracle
s = s[::-1] # reverse s to fit qiskit's qubit ordering
for q in range(n):
    if s[q] == '0':
        bv_circuit.i(q)
    else:
        bv_circuit.cx(q, n)

bv_circuit.barrier()

#Apply Hadamard gates after querying the oracle
for i in range(n):
    bv_circuit.h(i)

# Measurement
for i in range(n):
    bv_circuit.measure(i, i)

bv_circuit.draw()
```

… …

Out[3]:

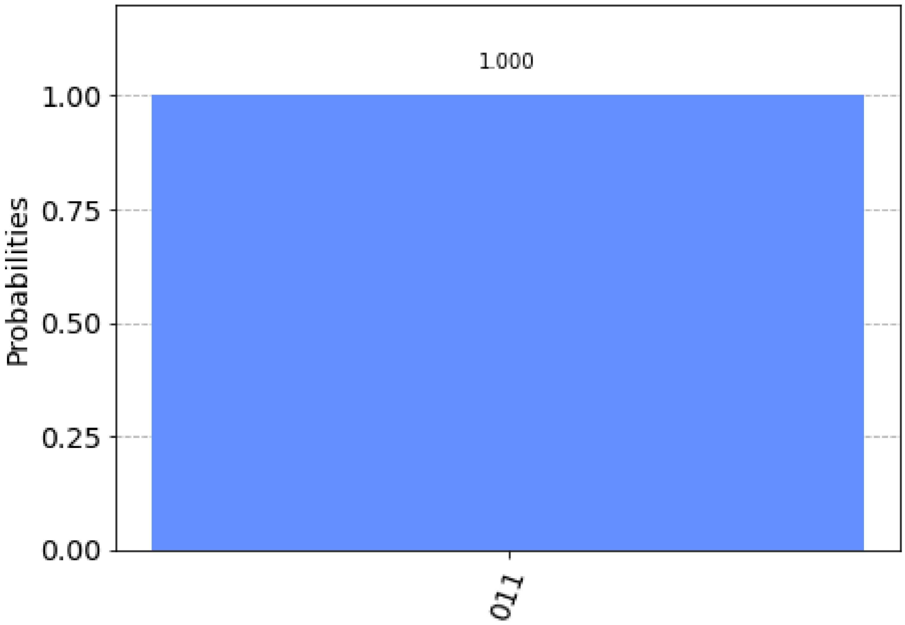


4. Simulator

In [4]:

```
# local simulator

sim = Aer.get_backend('aer_simulator')
shots = 1024
qobj = assemble(bv_circuit, sim)
results = sim.run(qobj).result()
answer = results.get_counts()

plot_histogram(answer)
```

Out[4]:



5. Real Device

In [5]:

```
IBMQ.load_account()
provider = IBMQ.get_provider(hub='ibm-q')
backend = least_busy(provider.backends(filters=lambda x: x.configuration().n_qubits
                                    not x.configuration().simulator and x.status().op
print("least busy backend: ", backend)
```

```
least busy backend:  ibmq_quito
```

In [6]:

```
# Run our circuit on the least busy backend. Monitor the execution of the job in the
from qiskit.tools.monitor import job_monitor

shots = 1024
```

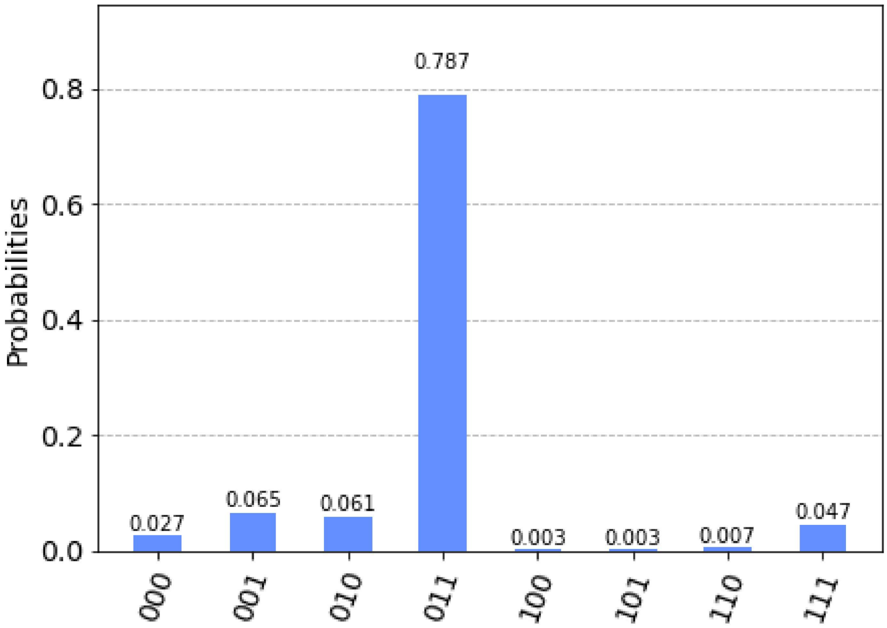
```
transpiled_bv_circuit = transpile(bv_circuit, backend)
job = backend.run(transpiled_bv_circuit, shots=shots)

job_monitor(job, interval=2)
```

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

In [7]:

```
# Get the results of the computation
results = job.result()
answer = results.get_counts()

plot_histogram(answer)
```

Out[7]:



Most of the results are 011. The other results are due to errors in the quantum computation.

In [ ]: