SimonsAlgorithm

March 17, 2021

[1]:
```
%matplotlib inline
# Importing standard Qiskit libraries
from qiskit import QuantumCircuit, execute, Aer, IBMQ
from qiskit.compiler import transpile, assemble
from qiskit.visualization import *

# Loading your IBM Q account(s)
provider = IBMQ.load_account()
```

```
C:\Users\lenovo\anaconda3\lib\site-
packages\qiskit\providers\ibmq\ibmqfactory.py:192: UserWarning: Timestamps in
IBMQ backend properties, jobs, and job results are all now in local time instead
of UTC.
  warnings.warn('Timestamps in IBMQ backend properties, jobs, and job results '
```

0.1 Defining the Circuit

Simon's algorithm is a demonstration of how quantum computers outperform classical computers. The essential theory is that we are given a black box function and we want to know if it is injective or not. Now on classical computers we would have to keep feeding the function unique input until it displays a collision such that $ f(x_1) = f(x_2) $ when $x\_1 \neq x\_2$. If this never happens then the function is 1-1.

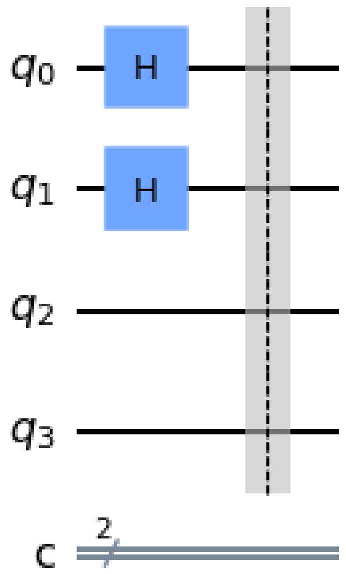
But on a quantum computer, we can make use of quantum propeties just like in Berstein-Vazirani. The implementation is as follows :

[15]:
```
qSimon = QuantumCircuit(4, 2)
qSimon.h(0)
qSimon.h(1)
qSimon.barrier()
```

[15]:
```
<qiskit.circuit.instructionset.InstructionSet at 0x1ffe39dd670>
```

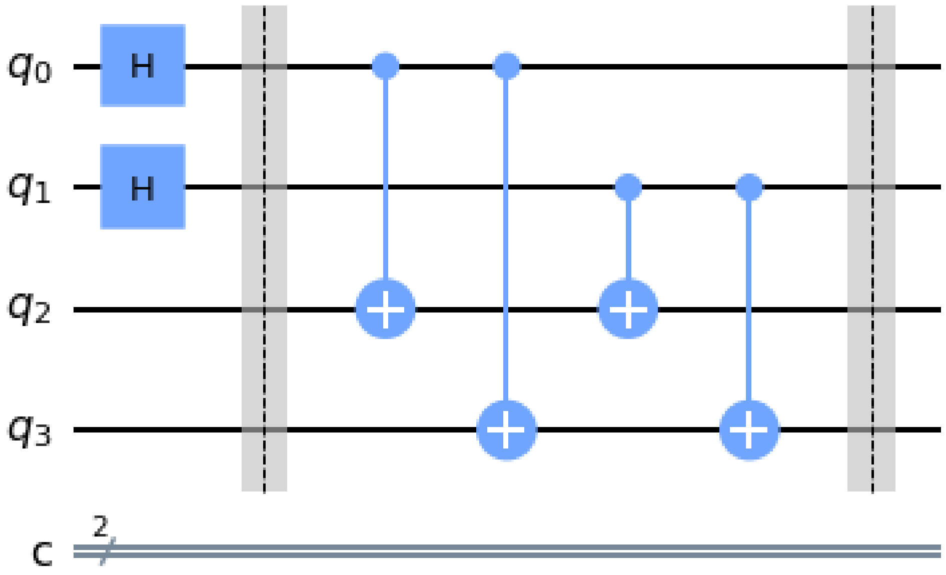
[16]:
```
qSimon.draw('mpl')
```

[16]:



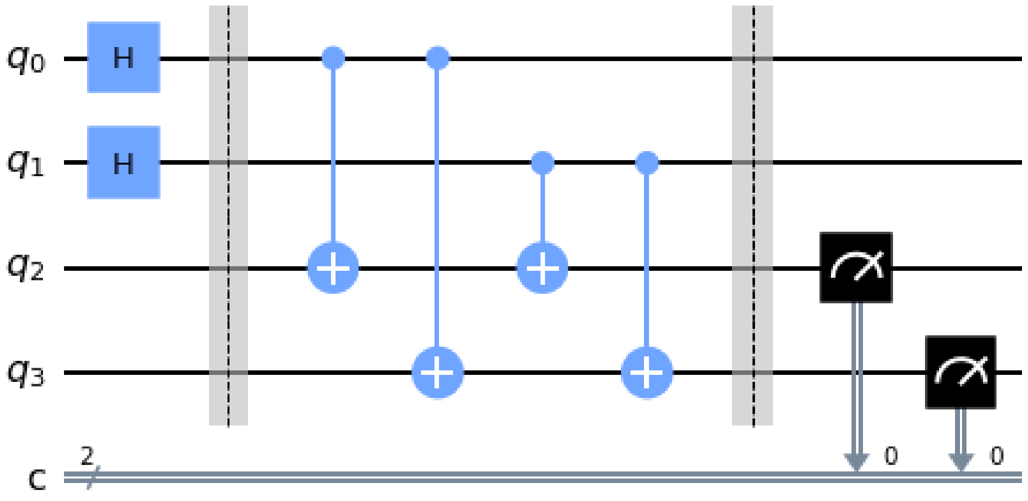
```
[8]: qSimon.cx(0,2)
     qSimon.cx(0,3)
     qSimon.cx(1,2)
     qSimon.cx(1,3)
     qSimon.barrier()
     qSimon.draw('mpl')
```

[8]:



[9]:
```
qSimon.measure(2,0)
qSimon.measure(3,0)
qSimon.draw('mpl')
```

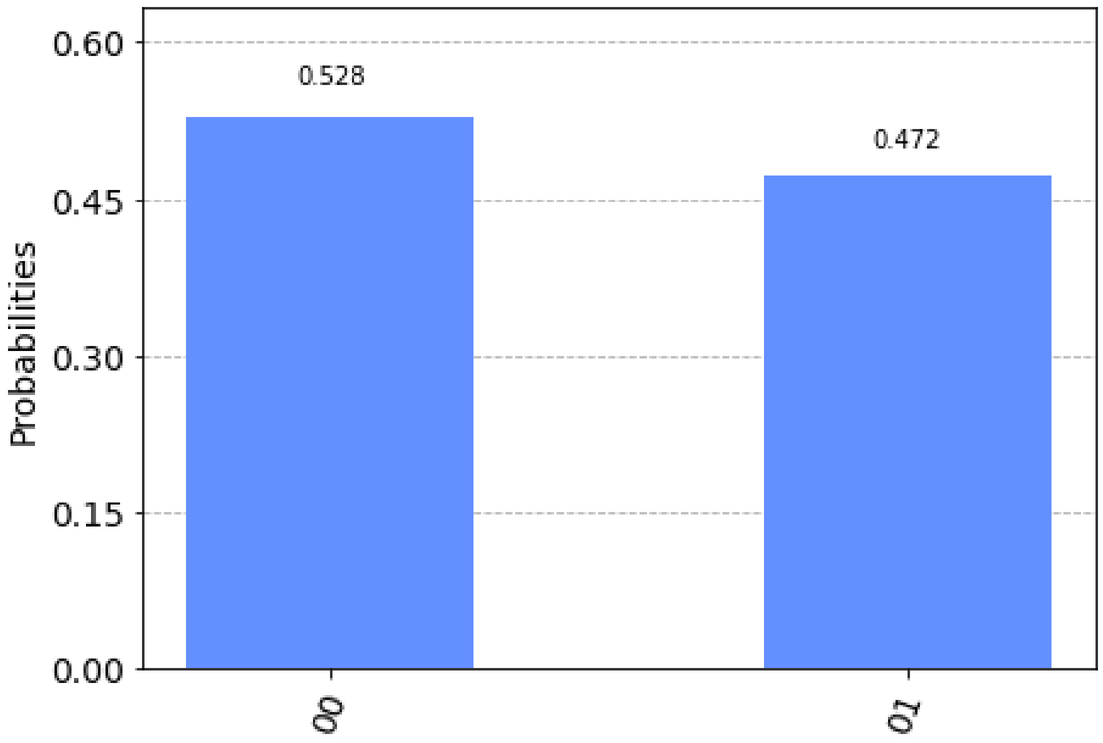
[9]:



0.2 Running it on a local quantum simulator

[10]:
```
backend = Aer.get_backend('qasm_simulator')
result = execute(qSimon, backend = backend, shots = 1024).result()
counts = result.get_counts()
plot_histogram(counts)
```

[10]:



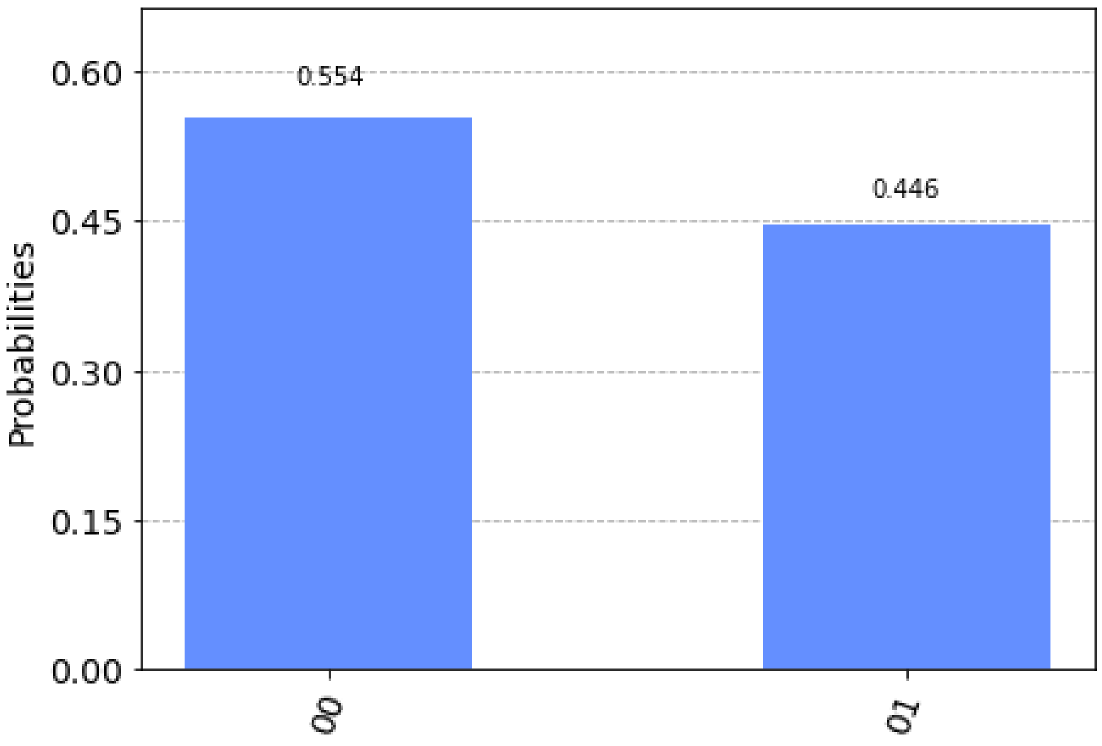
0.3 Running it on a IBM Q quantum computer

[11]:
```
from qiskit.tools.monitor import job_monitor
qProvider = IBMQ.get_provider();
qComp = qProvider.get_backend('ibmq_santiago')
job = execute(qSimon, backend = qComp, shots = 1024)
job_monitor(job)
```

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

[12]:
```
counts = job.result().get_counts()
plot_histogram(counts)
```

[12]:



0.3.1 References

- Implementing Simon's Algorithm
- Simon's Algorithm