Quantum computing Homework - 4

Adrian Kowalski

January 2026

Homework 4

Reference book:

Reference book: Chapter 13 of Roberto Loredo. Learn Quantum Computing with Python and IBM Quantum Experience A hands-on introduction to quantum computing and writing your own quantum programs with Python. Here is the source code used throughout this book:
https://github.com/PacktPublishing/Learn-Quantum-Computing-with-Python-and-IBM-Quantum-Experience/tree/master/Chapter13

Contents

1 Implementation of Deutsch's algorithm for a constant function.

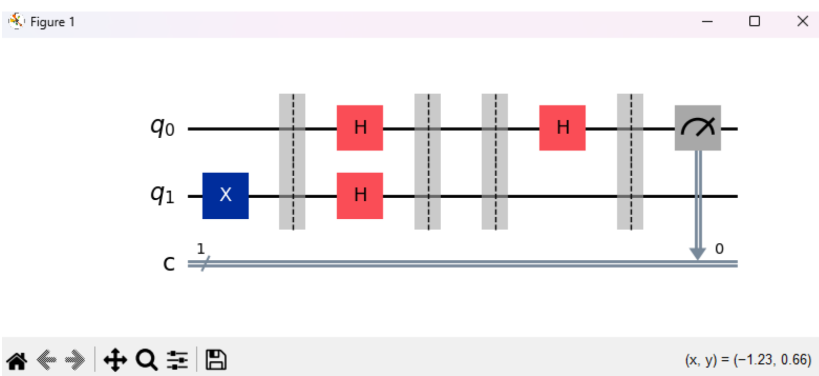
1.1 Composinig the circuit:

1.1.1 Circuit in Qiskit code:

Qiskit

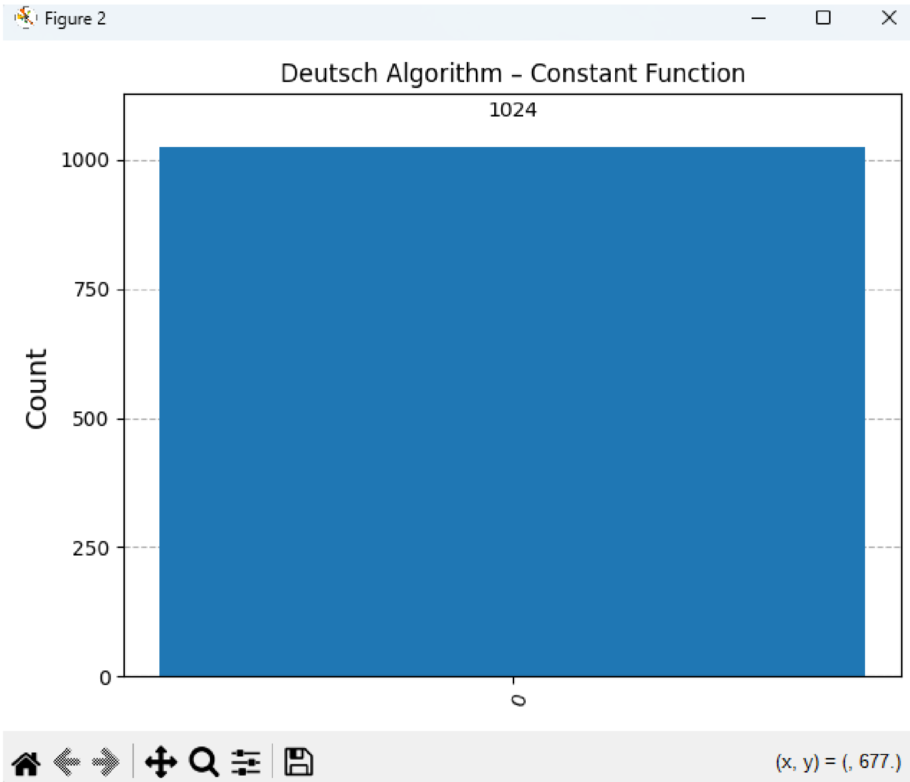
```
import...

qc = QuantumCircuit(2, 1)
qc.x(1)
qc.barrier()
qc.h(0)
qc.h(1)
qc.barrier()
#qc.cx(0, 1)
qc.barrier()
qc.h(0)
qc.barrier()
qc.measure(0, 0)
qc.draw(output='mpl')

sim = AerSimulator()
job = sim.run(qc, shots=1024)
result = job.result()
plot_histogram(counts,title='Deutsch Algorithm
    Constant Function')
plt.show()
```

1.1.2 Circuit in gate form:



1.2 Results of the simulation:

1.2.1 Histogram:



1.2.2 Measurements:

Qiskit

```
counts = result.get_counts()
print("Counts:", counts)
```

Counts: '0': 1024

2 Which algorithm would you use to determine whether an n-bit string is balanced?

To determine whether a function is balanced or constant, we could use the Deutsch-Jozsa algorithm, which is a more general expression of the normal Deutsch algorithm.
The algorithm considers a function:

$$f\{0,1\}^n \rightarrow \{0,1\}$$

This method is better then classical since we would need $2^{n-1}+1$ queries to determine whether the results are constant or balanced in worst case.

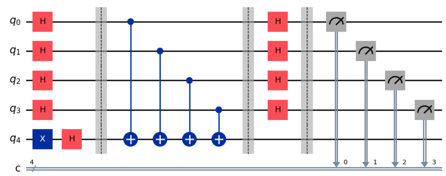
2.1 Exemplary circuit:

2.1.1 Circuit in Qiskit code:

Qiskit

```
import...

n = 4
qc = QuantumCircuit(n + 1, n)
for i in range(n):
    qc.h(i)
qc.x(n)
qc.h(n)
qc.barrier()
for i in range(n):
    qc.cx(i, n)
qc.barrier()
for i in range(n):
    qc.h(i)
qc.barrier()
qc.measure(range(n), range(n))
qc.draw(output='mpl')
```

2.1.2 Circuit in gate form:



3 Would phase kickback work if the ancilla qubit was not set to the state $|1\rangle$?

Phase kickback requires the ancilla qubit to be prepared in the state $|-\rangle$, which is obtained by applying a Hadamard gate to $|1\rangle$; if the ancilla is not prepared in this state, the oracle does not imprint the function value as a phase on the input register, and phase kickback does not occur.

3.1 Ancilla set to $|1\rangle$

$$|1\rangle \xrightarrow{H} |-\rangle$$

$$|-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

and so:

$$U_f|x, y\rangle = |x, y \oplus f(x)\rangle$$

$$\Downarrow$$

$$U_f|x\rangle|-\rangle = (-1)^{f(x)}|x\rangle|-\rangle$$

3.2 Ancila set to $|0\rangle$

$$|0\rangle \xrightarrow{H} |+\rangle$$

$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$$

and so:

$$U_f|x\rangle|+\rangle = |x\rangle|+\rangle$$

As we can see the Oracle doesn't imprint the functions and we don't get phase kickback, thus we cannot set the ancilla qubit to state $|0\rangle$.

4 Implementing the Bernstein-Vazirani algorithm to find the state $|11010\rangle$.

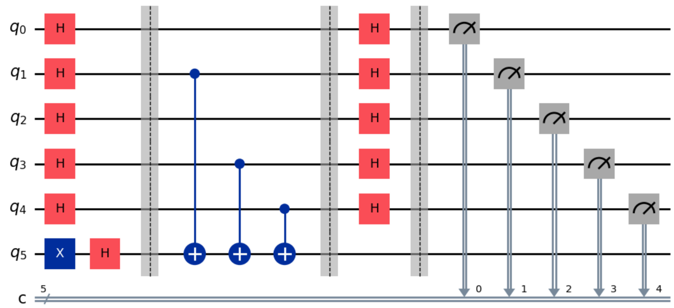
4.1 Composinig the circuit:

4.1.1 Circuit in Qiskit code:

Qiskit

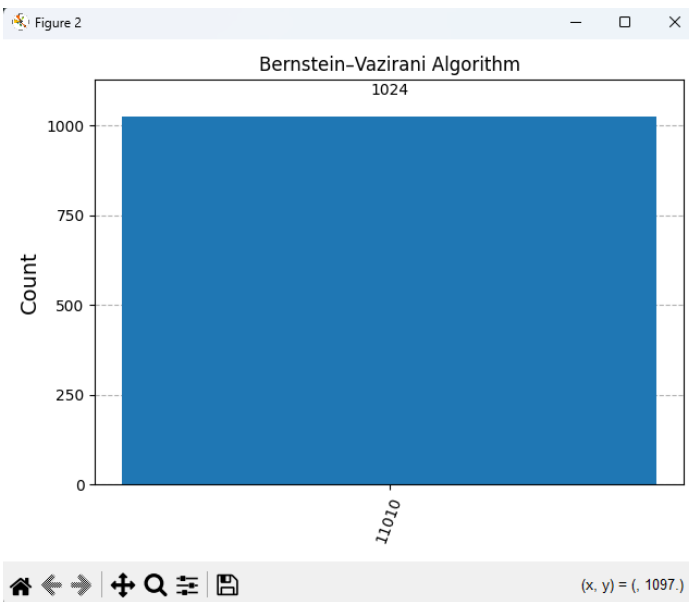
```
import...

ssh = "11010"
n = len(ssh)
qc = QuantumCircuit(n + 1, n)
for i in range(n):
    qc.h(i)
qc.x(n)
qc.h(n)
qc.barrier()
print("Secret before reverse:", ssh)
ssh = ssh[::-1]
print("Secret after reverse: ", ssh)
for i, bit in enumerate(ssh):
    if bit == "1":
        qc.cx(i, n)
qc.barrier()
for i in range(n):
    qc.h(i)
qc.barrier()
qc.measure(range(n), range(n))
qc.draw(output="mpl")

sim = AerSimulator()
result = sim.run(qc, shots=1024).result()
plot_histogram(counts, title="BV Algorithm")
plt.show()
```

4.1.2 Circuit in gate form:



4.2 Results of the simulation:

4.2.1 Histogram:



4.2.2 Measurements:

Qiskit

```
counts = result.get_counts()
print("Measurement results:", counts)
```

Measurement results: '11010': 1024

5 What would happen if you set the ancilla qubit in either of the algorithms by first placing a Hadamard gate, followed by an X gate? Explain the reason for the results.

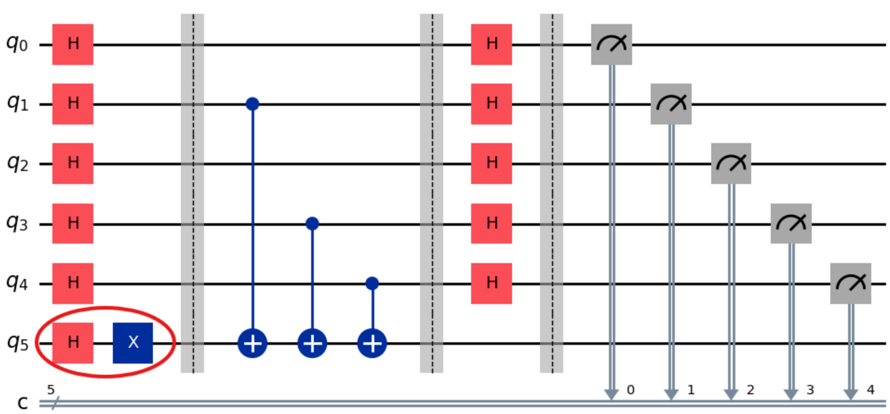
5.1 Composinig the circuit:

5.1.1 Modified BV algorithm:

Qiskit

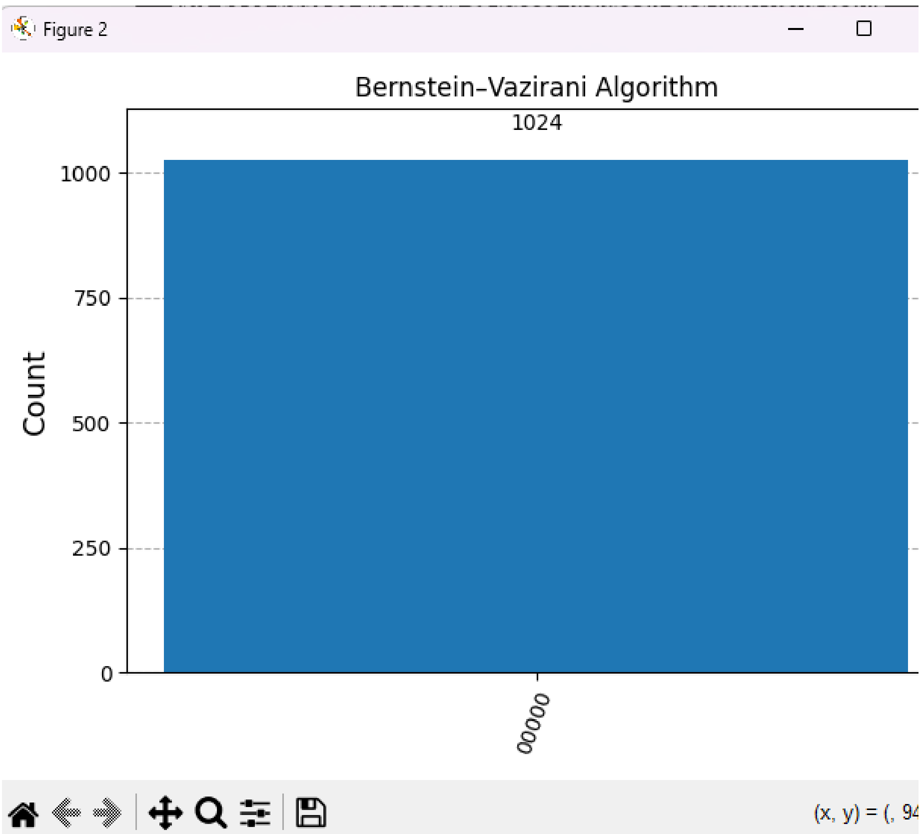
```
.
.
.
for i in range(n):
    qc.h(i)
qc.h(n)
qc.x(n)
qc.barrier()
.
.
.
```

5.1.2 Circuit in gate form:



5.2 Results of the simulation:

5.2.1 Histogram:



5.2.2 Measurements:

Qiskit

```
counts = result.get_counts()
print("Measurement results:", counts))
```

Measurement results: '00000': 1024

5.3 Explaining Results

If we apply Hadamard first, then X to the ancilla qubit, the state ends up as $|+\rangle$ and phase kickback does not occur, so both Deutsch–Jozsa and Bernstein–Vazirani fail.
Start with 0 apply H gate:

$$H|0\rangle = |+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$$

Apply X gate:

$$\Downarrow X$$

$$= \frac{|1\rangle + |0\rangle}{\sqrt{2}} = |+\rangle$$

we can see X only flips basis states, not phases.
Once again Oracle does nothing:

$$U_f|x\rangle|+\rangle = |x\rangle|+\rangle$$

And because of this the algorithm is reduced to just two Hadamrds gates which explains why the return will always be zeros.

6 Program and create an automated oracle generator for the Bernstein-Vazirani algorithm that randomly generates the secret state.

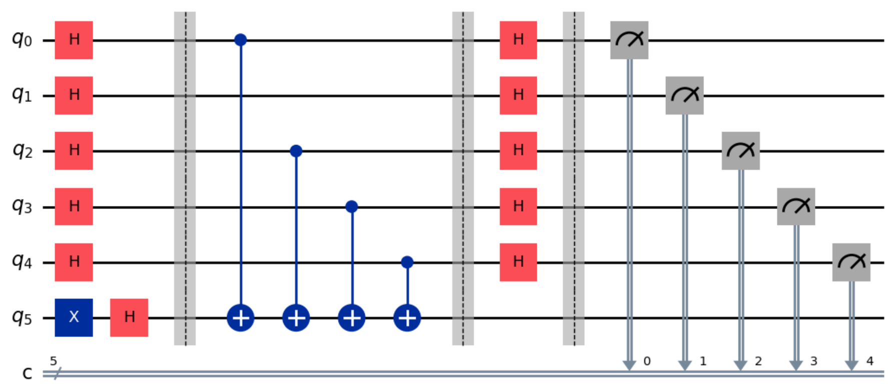
6.1 Composinig the circuit:

6.1.1 Modified BV algorithm:

Qiskit

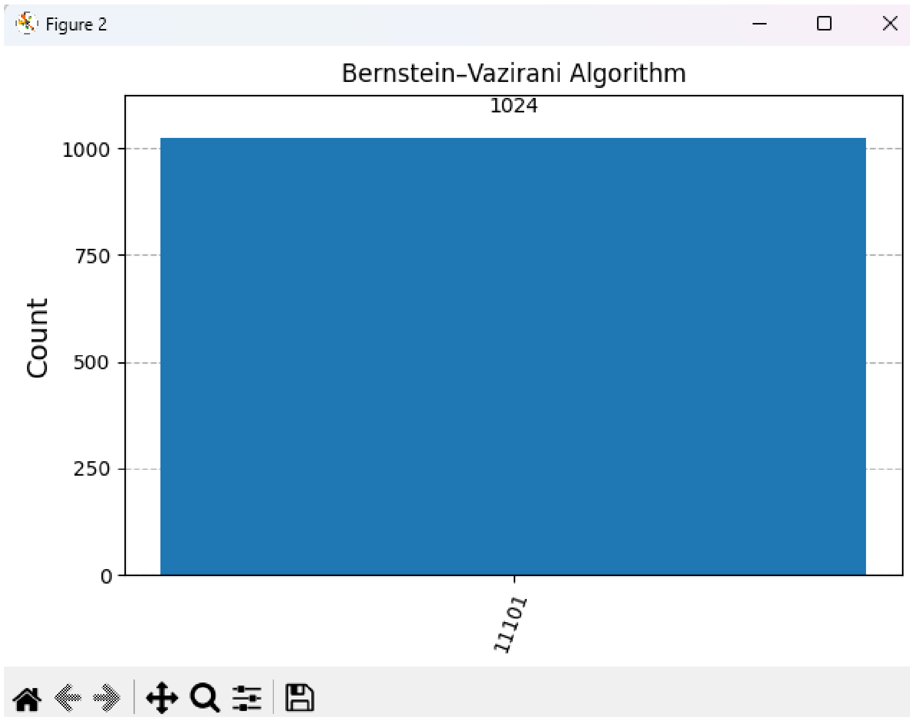
```
import random
.
.

n = 5 # len of oracle
ssh = ''.join(random.choice(['0','1']) for i in range(n))
qc = QuantumCircuit(n + 1, n)
.
.
```

6.1.2 Circuit in gate form:



6.2 Results of the simulation:

6.2.1 Histogram:



6.2.2 Measurements:

Qiskit

```
counts = result.get_counts()
print("Measurement results:", counts)
```

Measurement results: '11101': 1024

6.3 Can you determine the value by just running the circuit and reviewing the results?

Yes by running Bernstein-Vazirani circuit we can determine the secret sate of the oracle.
As said before the algorithm exploits phase kickback and quantum interference to encode oracle state onto register.