Quantum Computing with Qiskit
Qubits, Quantum Gate and Measurement

• A quantum computation is a collection of three elements
  ① A quantum register or a set of quantum register.
  ② A unitary matrix, which is used to execute a given quantum algorithm.
  ③ Measurement to extract information we need.

• Quantum circuit model
  • Universal quantum computer
Quantum Properties

• Superposition

The linear combination of two or more state vectors is another state vector in the same Hilbert space and describes another state of the system.

• Entanglement

Two systems are in a special case of quantum mechanical superposition called *entanglement* if the measurement of one system is correlated with the state of the other system in a way that is stronger than correlations in the classical world.

Source: Hidary, Quantum Computing: An Applied Approach
Quantum Circuit

Source: https://qiskit.org/textbook/ch-states/introduction.html
Quantum Gates

• Quantum Operators
  • In gate-based quantum computers, these operators are used to evolve the state.
    • Unitary and reversible.
    • Single qubit gate: rotation in block sphere.

• There exist universal quantum gates set.

Single Qubit Gates

\[ X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}; \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}; \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \]

\[ H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}; \quad S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}; \quad T = \begin{bmatrix} 1 & 0 \\ 0 & \exp(i\pi/4) \end{bmatrix} \]

\[ u3(\theta, \phi, \lambda) = \begin{pmatrix} \cos(\theta/2) & -e^{i\lambda}\sin(\theta/2) \\ e^{i\phi}\sin(\theta/2) & e^{i\lambda} + i\phi\cos(\theta/2) \end{pmatrix}. \quad u2(\phi, \lambda) = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -e^{i\lambda} \\ e^{i\phi} & e^{i(\phi + \lambda)} \end{pmatrix}. \quad u1(\lambda) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\lambda} \end{pmatrix}. \]
Quantum Gates in Qiskit

• Very Important!
  • The LSB (Least Significant Bit) is from right to left in qiskit.

\[ |q_{n-1}, \ldots, q_1, q_0 \rangle = |q_{n-1}\rangle \otimes |q_{n-2}\rangle \otimes \cdots |q_1\rangle \otimes |q_0\rangle \]


Also, this show the difference between entangle and separate state
Qiskit Circuit Example

```
from qiskit import QuantumCircuit

def create_quantum_circuit()
    q_circ = QuantumCircuit(4, 4)
    q_circ.x([0,1]) # x gate on q0 & q1
    q_circ.barrier()
    q_circ.cx(1,2) # control not gate (control,target)
    q_circ.cx(0,2)
    q_circ.ccx(0,1,3) # control control not gate (control1,control2,target)
    q_circ.barrier()
    q_circ.measure([2, 3], [0, 1]) # measurement (target_qubitlist,classical_bitlist)

    q_circ.draw(output='mpl', plot_barriers=True) # draw circuit you can also use: print(circuit_name)
```

Out[1]:
Quantum Computing Stack

# Qiskit Overview

<table>
<thead>
<tr>
<th>Institution</th>
<th>IBM</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Release</td>
<td>0.1 on March 7, 2017</td>
</tr>
<tr>
<td>Open Source</td>
<td>Yes</td>
</tr>
<tr>
<td>License</td>
<td>Apache-2.0</td>
</tr>
<tr>
<td>HomePage</td>
<td><a href="https://qiskit.org/">https://qiskit.org/</a></td>
</tr>
<tr>
<td>Github</td>
<td><a href="https://github.com/Qiskit">https://github.com/Qiskit</a></td>
</tr>
<tr>
<td>Documentation</td>
<td><a href="https://qiskit.org/documentation/">https://qiskit.org/documentation/</a></td>
</tr>
<tr>
<td>OS</td>
<td>Mac, Windows, Linux</td>
</tr>
<tr>
<td>Language</td>
<td>Python</td>
</tr>
<tr>
<td>Quantum Language</td>
<td>OpenQASM</td>
</tr>
</tbody>
</table>

## Version Information

<table>
<thead>
<tr>
<th>Qiskit Software</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qiskit</td>
<td>0.17.0</td>
</tr>
<tr>
<td>Terra</td>
<td>0.12.0</td>
</tr>
<tr>
<td>Aer</td>
<td>0.4.1</td>
</tr>
<tr>
<td>Ignis</td>
<td>0.2.0</td>
</tr>
<tr>
<td>Aqua</td>
<td>0.6.5</td>
</tr>
<tr>
<td>IBM Q Provider</td>
<td>0.6.0</td>
</tr>
</tbody>
</table>
The Qiskit Elements

Terra, the ‘earth’ element, is the foundation on which the rest of the software lies.

Aera, the ‘air’ element, permeates all Qiskit elements. For accelerating development via simulators, emulators and debuggers.

Aqua, the ‘water’ element, is the element of life. For building algorithms and applications.

Ignis, the ‘fire’ element, is dedicated to fighting noise and errors and to forging a new path.

Source: https://qiskit.org/documentation/the_elements.html
## Different Simulator

<table>
<thead>
<tr>
<th>Simulator Backends</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QasmSimulator ([configuration, provider])</td>
<td>Noisy quantum circuit simulator backend.</td>
</tr>
<tr>
<td>StatevectorSimulator ([configuration, provider])</td>
<td>Ideal quantum circuit statevector simulator</td>
</tr>
<tr>
<td>UnitarySimulator ([configuration, provider])</td>
<td>Ideal quantum circuit unitary simulator.</td>
</tr>
<tr>
<td>PulseSimulator ([configuration, provider])</td>
<td>Pulse schedule simulator backend.</td>
</tr>
</tbody>
</table>
Qiskit Code Example

```python
In [1]: from qiskit import QuantumCircuit
q_bell = QuantumCircuit(2, 2)
q_bell.barrier()
q_bell.h(0)
q_bell.cx(0, 1)
q_bell.barrier()
q_bell.measure([0, 1], [0, 1])
q_bell.draw(output='mpl', plot_barriers=True)
```

```
In [2]: from qiskit import Aer, execute
from qiskit.visualization import plot_histogram
backend = Aer.get_backend('qasm_simulator')
job_sim = execute(q_bell, backend, shots=10000)
sim_result = job_sim.result()
print(sim_result.get_counts(q_bell))
plot_histogram(sim_result.get_counts(q_bell))
```

```
Out[2]:
{’11’: 50254, ’00’: 49746}
```
from qiskit import QuantumCircuit
q_bell = QuantumCircuit(2, 2)
q_bell.barrier()
q_bell.h(0)
q_bell.cx(0, 1)
q_bell.barrier()
q_bell.draw(output='mpl', plot_barriers=True)

from qiskit import Aer, execute
unitary_backend=Aer.get_backend('unitary_simulator')
result = execute(q_bell,unitary_backend).result()
print(result.get_unitary())

[[ 0.70710678+0.00000000e+00j  0.70710678-8.65956056e-17j]
 0.00000000e+00j  0.00000000e+00j]
 0.00000000e+00j  0.00000000e+00j]
 0.70710678+0.00000000e+00j -0.70710678+8.65956056e-17j]
 0.00000000e+00j  0.00000000e+00j]
 0.70710678+0.00000000e+00j  0.70710678+8.65956056e-17j]
 0.00000000e+00j  0.00000000e+00j]

from qiskit import Aer, execute
statevector_backend=Aer.get_backend('statevector_simulator')
result = execute(q_bell,statevector_backend).result()
print(result.get_statevector())

[0.70710678+0.j  0.  +0.j  0.  +0.j  0.70710678+0.j]
Real QC

Histogram

Probabilities

State

00  54.08%
01  2.78%
10  2.72%
11  40.44%
Installation

• Anaconda (highly recommended for learning)
  • pip install qiskit

• IBM online notebook

• This is CS course.

Reference: https://qiskit.org/documentation/install.html#install
The Deutsch Algorithm

• The first to demonstrate quantum over classical computing.

Deutsch Problem:
Given a black box that implement some Boolean function \( f: \{0,1\} \rightarrow \{0,1\} \).
We are promised that the function is either constant or balanced.

<table>
<thead>
<tr>
<th>( x )</th>
<th>( f_0 )</th>
<th>( f_1 )</th>
<th>( f_x )</th>
<th>( f_{\overline{x}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

require two function evaluations to figure out the answer.
The Deutsch Algorithm

- We need only one query on a quantum computer!

$$|\psi_0\rangle = |0\rangle \otimes |1\rangle$$

$$|\psi_1\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

$$|\psi_2\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle)$$

$$f$$ is Boolean function.

$$|\psi_2\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|f(x)\rangle - |f(x)\rangle)$$
Phase Kick Back

• Useful trick in many quantum algorithms.

Consider Quantum Black box function $f$

$$f(x) = 0 : \quad \frac{|0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle}{\sqrt{2}} = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

$$f(x) = 1 : \quad \frac{|0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle}{\sqrt{2}} = \frac{|1\rangle - |0\rangle}{\sqrt{2}} = -\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

$$\left|0 \oplus f(x)\right\rangle - \left|1 \oplus f(x)\right\rangle = (-1)^f(x)\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

$$U_f \left(|0\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\right) = |x\rangle \otimes \frac{1}{\sqrt{2}}\left(|0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle\right)$$

$$U_f \left(|x\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\right) = (-1)^f(x) \left(|x\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\right)$$

$$U_f \left((a_0|0\rangle + a_1|1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\right) = \left((-1)^f(0)a_0|0\rangle + (-1)^f(1)a_1|1\rangle\right) \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$
The Deutsch Algorithm

\[ |\psi_0\rangle = |0\rangle \otimes |1\rangle \]

\[ |\psi_1\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \]

\[ |\psi_2\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|f(x)\rangle - |f(x)\rangle) \]

\[ U_f \left( (c_0|0\rangle + c_1|1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \right) = (-1)^{f(0)}c_0|0\rangle + (-1)^{f(1)}c_1|1\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \]

\[ |\psi_2\rangle = \begin{cases} 
  \pm \left[ \frac{|0\rangle + |1\rangle}{\sqrt{2}} \right] \left[ \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right] & \text{if } f(0) = f(1) \\
  \pm \left[ \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right] \left[ \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right] & \text{if } f(0) \neq f(1).
\end{cases} \]

\[ |\psi_3\rangle = \begin{cases} 
  \pm |0\rangle \left[ \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right] & \text{if } f(0) = f(1) \\
  \pm |1\rangle \left[ \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right] & \text{if } f(0) \neq f(1).
\end{cases} \]
Qiskit Example

<table>
<thead>
<tr>
<th>x</th>
<th>$f_0$</th>
<th>$f_1$</th>
<th>$f_x$</th>
<th>$f_{\bar{x}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

constant output = 0
balanced output = 1

$f_0 \rightarrow I$
$f_1 \rightarrow -I$
$f_x \rightarrow Z$
$f_{\bar{x}} \rightarrow -Z$

```
from qiskit import QuantumCircuit
def_x(QC)
QC.x(1)  # psi0
QC.h([0,1])  # psi1
QC.cx([0,1])  # psi2
QC.x(1)  # psi3
QC.measure([0],[0])
def_x.draw(output='mpl')  # draw your circuit
```

```
from qiskit import Aer, execute
from qiskit.visualization import plot_histogram
backend = Aer.get_backend('qasm_simulator')
demo_circ = execute(q_demo, backend, memory=False)
result = demo_circ.result()

#memory = result.get_memory(q_demo)
print(result.get_counts(q_demo))
plot_histogram(result.get_counts(q_demo))
```
Extension: The Deutsch-Jozsa algorithm

- This time the function $f$ is a function from $n$ bits string to a bit.

<table>
<thead>
<tr>
<th>The Deutsch–Jozsa Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> A black-box for computing an unknown function $f : {0, 1}^n \rightarrow {0, 1}$.</td>
</tr>
<tr>
<td><strong>Promise:</strong> $f$ is either a constant or a balanced function.</td>
</tr>
<tr>
<td><strong>Problem:</strong> Determine whether $f$ is constant or balanced by making queries to $f$.</td>
</tr>
</tbody>
</table>
The Deutsch-Jozsa algorithm

\[ |\psi_0\rangle = |0\rangle^\otimes n \otimes |1\rangle \]

\[ |\psi_1\rangle = \frac{1}{\sqrt{2^n}} \sum_x |x\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \]

\[ |\psi_2\rangle = \frac{1}{\sqrt{2^n}} \sum_x (-1)^{f(x)} |x\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \]

\[ |\psi_3\rangle = \frac{1}{\sqrt{2^n}} \sum_y \sum_x (-1)^{f(x)+y\cdot x} |y\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \]
Algorithm: Deutsch–Jozsa

Inputs: (1) A black box $U_f$ which performs the transformation $|x\rangle|y\rangle \rightarrow |x\rangle|y \oplus f(x)\rangle$, for $x \in \{0, \ldots, 2^n - 1\}$ and $f(x) \in \{0, 1\}$. It is promised that $f(x)$ is either constant for all values of $x$, or else $f(x)$ is balanced, that is, equal to 1 for exactly half of all the possible $x$, and 0 for the other half.

Outputs: 0 if and only if $f$ is constant.

Runtime: One evaluation of $U_f$. Always succeeds.

Procedure:

1. $|0\rangle^\otimes n |1\rangle$ initialize state
2. $\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \left[ \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right]$ create superposition using Hadamard gates
3. $\sum_{x} (-1)^{f(x)} |x\rangle \left[ \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right]$ calculate function $f$ using $U_f$
4. $\sum_{z} \sum_{x} (-1)^{x \cdot z + f(x)} |z\rangle \left[ \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right]$ perform Hadamard transform
5. $z$ measure to obtain final output $z$
Qiskit Example
Improvement?

- The worst case in classical take us $\frac{2^n}{2} + 1$
- In quantum we need only one query ..... But, if there exist error ?

- That’s why we have Bernstein-Vazirani, Simon’s algorithm.
Where are we now?
NISQ Quantum Computing

- Noisy Intermediate-Scale Quantum

- Circuit depth is important.

Depth = 6 in this case
Reference

• Nielsen & Chuang : Quantum Computation and Quantum Information

• Hidary : Quantum Computing: An Applied Approach

• Qiskit Textbook : https://qiskit.org/textbook/preface.html