Resources: IBM and qiskit

- https://qiskit.org/
- https://qiskit.org/aqua
- https://quantumexperience.ng.bluemix.net/qx/editor
Resources: cirq and Forest

Cirq
- https://github.com/quantumlib/Cirq

Forest
- https://github.com/rigetti/pyquil
- pyquil.readthedocs.io/en/latest

see

https://quantum-computing.ibm.com/support
Example: The Deutsch-Jozsa algorithm

- One of the first algorithms to demonstrate usefulness of QC

Problem: given a function $f$ from $\{0, 1\}$ to itself determine whether $f$ is a constant function.

- The function is constant when $f(x) \equiv 0 \forall x$ or $f(x) \equiv 1 \forall x$ ($\forall = \text{for all}$). It is balanced otherwise.

- Here are all possible 2-bit functions:
  - Constant: $f_0$, $f_1$, balanced: $f_x$, $f_{\overline{x}}$

<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>

- Normally we need 2 evaluations to solve the problem [one eval. = querying one qubit]

- Can do it with one - with quantum computing

- For $n$ bit functions would classically need $n$ evals. QC: one
The Deutsch-Jozsa algorithm

- First: \( f \) is not injective - so cannot tell \( x \) from \( f(x) \). It is not reversible. Make it reversible with a trick

- Define 'Oracle':
  \[
  U_f(|x\rangle|y\rangle) := |x\rangle|y \oplus f(x)\rangle
  \]
  * Note: \( \oplus \) == addition mod 2 == XOR

**D1** Show that \( U_f \circ U_f = I \) (where: \( \circ = \) composition)

- From above exercise we see that \( U_f \) is now reversible (even though \( f \) may not be)

- Consider \( U_f \) as a function of the 2 qubits \( x \) and \( y \)

**D2** Show that when \( f = f_0 \) then \( U_f \) is the identity

**D3** Show: when \( f = f_1 \) then \( U_f \) does an XOR on the 2nd qubit
When \( f = f_x \) then \( U_f \) does the CNOT operation:

**Case \( f = f_x \)**

Control=\( x \), Target=\( y \)

\[
U_f (|x\rangle|y\rangle) =
\begin{align*}
|00\rangle & \rightarrow |00\rangle \\
|01\rangle & \rightarrow |11\rangle \\
|10\rangle & \rightarrow |10\rangle \\
|11\rangle & \rightarrow |11\rangle
\end{align*}
\]

When \( f = f\overline{x} \) then \( U_f \) does the operation:

**Case \( f = f\overline{x} \)**

\[
U_f (|x\rangle|y\rangle) =
\begin{align*}
|00\rangle & \rightarrow |00\rangle \\
|01\rangle & \rightarrow |10\rangle \\
|10\rangle & \rightarrow |10\rangle \\
|11\rangle & \rightarrow |11\rangle
\end{align*}
\]

Note: all second bits are flipped from case \( f_x \) above - therefore:

- This is a CNOT operation followed by a NOT (X) on 2nd qubit.

Show that for a given \( f \), \( U_f \) (a 2 qubit operator) is linear and that it is unitary. What is its matrix representation for each of the 4 functions \( f_0, f_1, f_x, f\overline{x} \)?

- Deutsch-Jozsa algorithm based on exploiting superposed states

- Take second qubit as \( |\overline{\text{\_}}\rangle = \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) \) and apply oracle.
\[ U_f |x\rangle |\rangle = U_f |x\rangle \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) \]

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

\[ = |x\rangle \frac{1}{\sqrt{2}} (|f(x)\rangle - |\bar{f}(x)\rangle) \]

\[ = (-1)^{f(x)} |x\rangle |\rangle \]

**Known as the phase kick-back trick** – value of the function reflected in phase.

**Q:** If we observe the first qubit on output: to what operation is the oracle equivalent for \( f_0, f_1, f_x, f_{\bar{x}} \)?

**A:**

<table>
<thead>
<tr>
<th>( f_0 )</th>
<th>( f_1 )</th>
<th>( f_x )</th>
<th>( f_{\bar{x}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( I )</td>
<td>( -I )</td>
<td>( Z )</td>
<td>( -Z )</td>
</tr>
</tbody>
</table>
One more transform: Exploit the relation $HZH = X$. Apply $H$ to $x$ before and after $U_f$. Let $x = |0\rangle$ (top qubit).

- If $f$ is either $f_0$ or $f_1$ we observe $\pm |0\rangle$
- If $f$ is either $f_x$ or $f_\bar{x}$ we observe a $\pm |1\rangle$

**Done!**

**DIAGRAM**

```
H  U_f  H
\ -> \ -> \\
X  H  
```

- Note: The actual final state has the form (prove it)

$$\psi = \pm |f(0) \oplus f(1)\rangle \left[ \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right]$$
Determine the states $\psi_0, \ldots, \psi_3$ after each ‘stage’

**Partial Solution:**

1. $|\psi_0\rangle = |01\rangle$

2. $|\psi_1\rangle = \left[ \frac{|0\rangle + |1\rangle}{\sqrt{2}} \right] \left[ \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right]$. Write as $|x\rangle|\rightarrow\rangle$

3. $|\psi_2\rangle = U_f(|x\rangle, |\rightarrow\rangle) = (-1)^f(x)|x\rangle|\rightarrow\rangle$  
   $= \frac{(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle}{\sqrt{2}}|\rightarrow\rangle$

   If $f(0) = f(1) \rightarrow$ same sign $\psi_2 = \pm |+\rangle|\rightarrow\rangle$

   Otherwise $\psi_2 = \pm |\rightarrow\rangle|\rightarrow\rangle$

4. Apply $H$ to 1st qubit of $\psi_2$:
   
   If $f(0) = f(1) \rightarrow \psi_3 = \pm |H+\rangle|\rightarrow\rangle = \pm |0\rangle|\rightarrow\rangle$

   Otherwise $\psi_3 = \pm |H-\rangle|\rightarrow\rangle = \pm |1\rangle|\rightarrow\rangle$
Quantum parallelism

- In effect the DJ algorithm is able to evaluate $f(0)$ and $f(1)$ at the same time.

- Assume same context: $f : \{0, 1\} \rightarrow \{0, 1\}$. Same oracle $U$.

Consider the circuit to the right. Show that the output is

\[
\frac{|0, f(0)\rangle + |1, f(1)\rangle}{\sqrt{2}}
\]

- In effect $|\Psi\rangle$ carries information about both $f(0)$ and $f(1)$!

- The above circuit is same as:
Generalization to $n + 1$ gates. Function $f$ is now from $\{0, 1\}^n$ to $\{0, 1\}$.

Recall the notation seen earlier: at top we have $n$ qubit at state $|0\rangle$ - each followed by Hadamard.

Output state is now:

$$\frac{1}{\sqrt{2^n}} \sum_x |x\rangle |f(x)\rangle$$

**Example:** When $n = 2$ – state $x$ input to $U_f$ is

$$x = \frac{1}{2} [ |00\rangle + |01\rangle + |10\rangle + |11\rangle ]$$

Output:

$$\frac{1}{2} [ |00, f(00)\rangle + |01, f(01)\rangle + |10, f(10)\rangle + |11, f(11)\rangle ]$$
Cirq codes

Resources:

- See [https://github.com/quantumlib/cirq](https://github.com/quantumlib/cirq)
- Also: the Cirq workshop bootcamp repository (google search it)
- Cirq provides a toolkit (a ‘framework’) for simulating quantum algorithms.
- Written in python. Implements all the gates we have seen and more.
- The following illustration shows a simple example
import cirq
q0 = cirq.NamedQubit("q0")
q1 = cirq.NamedQubit("q1")
q2 = cirq.NamedQubit("q2")
ops = [cirq.X(q0), cirq.H(q1), cirq.CNOT(q1, q2), cirq.X(q1),
       cirq.CZ(q0, q1)]
circuit = cirq.Circuit(*ops)
print(circuit)

Output:
A longer example showing many of the gates

```python
import cirq
import numpy as np
q0, q1, q2 = cirq.LineQubit.range(3)
ops = [cirq.X(q0),
       cirq.Y(q1),
       cirq.Z(q2),
       cirq.CZ(q0, q1),
       cirq.CNOT(q1, q2),
       cirq.H(q0),
       cirq.T(q1),
       cirq.S(q2),
       cirq.CCZ(q0, q1, q2),
       cirq.SWAP(q0, q1),
       cirq.CSWAP(q0, q1, q2),
       cirq.CCX(q0, q1, q2),
       cirq.ISWAP(q0, q1),
       cirq.Rx(0.5 * np.pi)(q0),
       cirq.Ry(0.5 * np.pi)(q1),
       cirq.Rz(0.5 * np.pi)(q2),
       (cirq.X ** 0.5)(q0)]
print(cirq.Circuit(*ops))
print(cirq.unitary(cirq.CNOT))
print(cirq.unitary(cirq.CZ))
```
Output:

A few commands to loot at:

- `cirq.X(q0)` : gate X at q0.
- `cirq.LineQubit.range(p)` : create a line of qubits .. or
- `cirq.GridQubit.range(p,q)` create a grid of qubits ..
- `print(cirq.Circuit(*ops))` prints circuit
Quantum Fourier Transform

- QFT is at the core of the Shor algorithm
- Main idea of QFT: Exploit product decomposition. Recall:

\[ x = [x_0, x_1, \cdots, x_{N-1}]^T \] is transformed to \( y \) with:

\[
y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2i\pi jk/N}
\]

Therefore:

\[
|j\rangle \longrightarrow \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{2i\pi jk/N} |k\rangle \quad (\ast)
\]

- Suppose that \( N = 2^n \). Write any \( k \) in its binary representation:

\[
k = k_1 2^{n-1} + k_2 2^{n-2} + \cdots + k_n 2^0 = \sum_{l=1}^{n} k_l 2^{n-l}
\]
Drop the scaling term $\frac{1}{\sqrt{N}}$ in (*) and set that $N = 2^n$. Then:

$$
\sum_{k=0}^{2^n-1} e^{2i\pi jk/2^n} |k\rangle = \sum_{k=0}^{2^n-1} e^{2i\pi j \sum_{l=1}^{n} k_l 2^{-l}} |k_1...k_n\rangle
$$

$$
= \sum_{k_1=0}^{1} \sum_{k_2=0}^{1} \cdots \sum_{k_n=0}^{1} \otimes_{l=1}^{n} e^{2i\pi j k_l 2^{-l}} |k_l\rangle
$$

$$
= \otimes_{l=1}^{n} \left[ \sum_{k_l=0}^{1} e^{2i\pi j k_l 2^{-l}} |k_l\rangle \right]
$$

$$
= \otimes_{l=1}^{n} \left[ |0\rangle + e^{2i\pi j 2^{-l}} |1\rangle \right]
$$
Write \( j = \sum_{m=1}^{n} j_m 2^{n-m} \). Since \( e^{2i\pi \times \text{integer}} = 1 \) then

\[
e^{2i\pi j 2^{-l}} = e^{2i\pi \sum_{m=1}^{n} j_m 2^{n-m} 2^{-l}} = e^{2i\pi \sum_{m=1}^{n} j_m 2^{n-l-m}} \]
\[
= e^{2i\pi \sum_{m=n-l+1}^{n} j_m 2^{n-l-m}} \]
\[
= e^{2i\pi \cdot j_{n-l+1} j_{n-l+2} \ldots j_n} \]

In the end:

\[
\frac{1}{2^{n/2}} \sum_{k=0}^{2^{n-1}} e^{2i\pi j k / 2^n} |k\rangle =
\]
\[
\frac{\left(|0\rangle + e^{2i\pi 0 \cdot j_n} |1\rangle\right) \left(|0\rangle + e^{2i\pi 0 \cdot j_{n-1} j_n} |1\rangle\right) \ldots \left(|0\rangle + e^{2i\pi 0 \cdot j_1 j_2 \ldots j_n} |1\rangle\right)}{2^{n/2}}
\]

Let \( R_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{2i\pi / 2^k} \end{pmatrix} \)
Here is a diagram for a 4-cubit QFT
Concluding notes

L. K. Glover

Will quantum computers ever grow into their software? How long will it take them to blossom into the powerful calculating engines that theory predicts they could be? I would not dare to guess, but I advise all would-be forecasters to remember these words, from a discussion of the Electronic Numerical Integrator and Calculator (ENIAC) in the March 1949 issue of Popular Mechanics:

Where a calculator on the ENIAC is equipped with 18,000 vacuum tubes and weighs 30 tons, computers in the future may have only 1,000 vacuum tubes and weigh only 1.5 tons.