

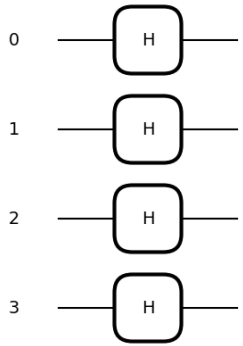
THE UNIVERSITY of EDINBURGH
informatics

Introduction to Quantum Computing

Pennylane: Oracles and Black-box Algorithms

Mina Doosti

Exercise 1: Uniform superpositions



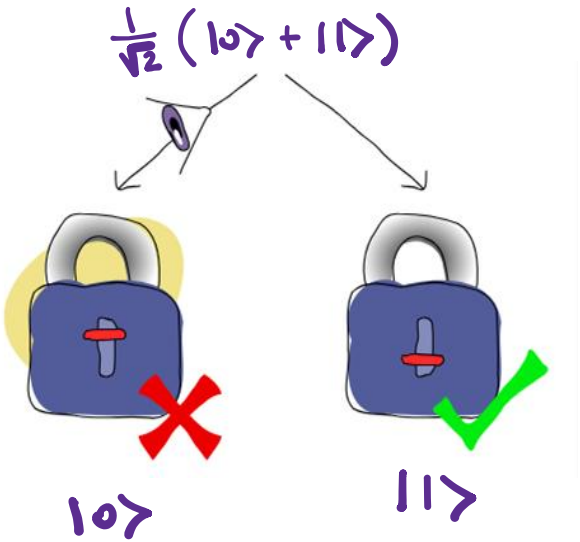

```
n_bits = 4
dev = pl.device("default.qubit", wires=n_bits)

@pl.qnode(dev)
def uniformsuperposition():
   """Build a circuit that creates an n-qubit uniform superposition"""
   return pl

pl.drawer.use_style("black_white")
pl.draw_mpl(uniformsuperposition)();
```

What does uniform superposition mean?

Can you always pick the lock with the uniform superposition?



We need to measure to get one of the answers, which happens with 50% probability

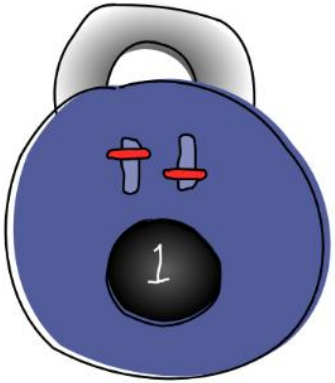
For 2 bits:

$$\frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle$$

For n-bits:

$$\{0,1\}^n \rightarrow 2^n \text{ options}$$

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle \xrightarrow{\text{Measure}} |x^*\rangle \quad \text{with} \quad p = \frac{1}{2^n}$$



Now let's say the lock has n bits, so we have 2^n combination where one of them, let's call s is the correct answer. Let's model this lock as a function:

$x = s \rightarrow \checkmark$
$x =$ anything else $\times$

$$f(x) = \begin{cases} 1 & x = s \\ 0 & \text{otherwise} \end{cases}$$

This function knows exactly the answer to "how to open the lock"
You only need to ask for an input x and it will tell you if it is the correct combination or not!
In computer science, we call these functions an Oracle!

How can you implement this with a unitary?

There are multiple ways!

$$U_f|x\rangle = \begin{cases} \text{some specific state} & x = s \\ \text{a different state} & \text{otherwise.} \end{cases}$$

$$U_f|x\rangle = (-1)^{f(x)}|x\rangle = \begin{cases} -|x\rangle & x = s \\ |x\rangle & x \neq s \end{cases}$$

This is one of the ways to quantumly implement a classical oracle

Exercise 2: Oracles

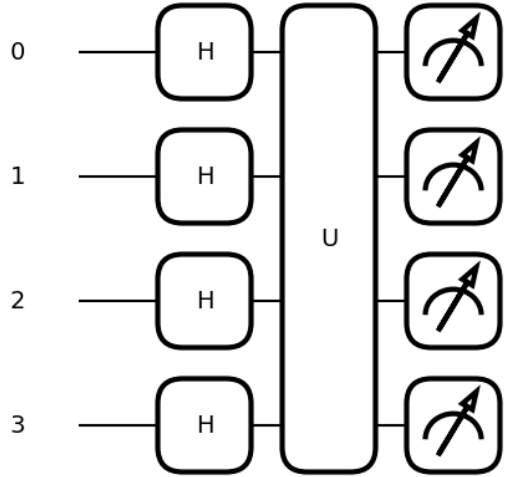
$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

```python
def oracle_matrix(key):
    """Create the unitary matrix corresponding to the binary key (list[int])
       e.g. key=[0,1,1] should give the diagonal matrix [1,1,1,-1,1,1,1,1]"""
    # Hint: use np.ravel_multi_index
    matrix = np.identity(2**len(key))
    return matrix

print(oracle_matrix([0,1,1,0]))
```

Exercise 2: Oracles

Asking your oracle a question!




```
@pl.qnode(dev)
def oracle_circuit(key):
    # Hint: use pl.QubitUnitary
    return pl.probs(wires=range(n_bits))

pl.draw_mpl(oracle_circuit)([0,1,1,0]);
print(oracle_circuit([0,1,1,0]))
```

The art of using a quantum oracle effectively!

Applying the oracle U_f by itself doesn't do anything interesting. You still need to brute force through all the combinations!

We need to translate that phase into a detectable phase. For example:

$$\frac{1}{\sqrt{2}}(|x_1\rangle + |x_2\rangle) \xrightarrow{U_f} \frac{1}{\sqrt{2}}\left((-1)^{f(x_1)}|x_1\rangle + (-1)^{f(x_2)}|x_2\rangle\right)$$

if $f(x_1) = 0$ & $f(x_2) = 1$ $\rightarrow$ $\frac{1}{\sqrt{2}}(|x_1\rangle - |x_2\rangle)$

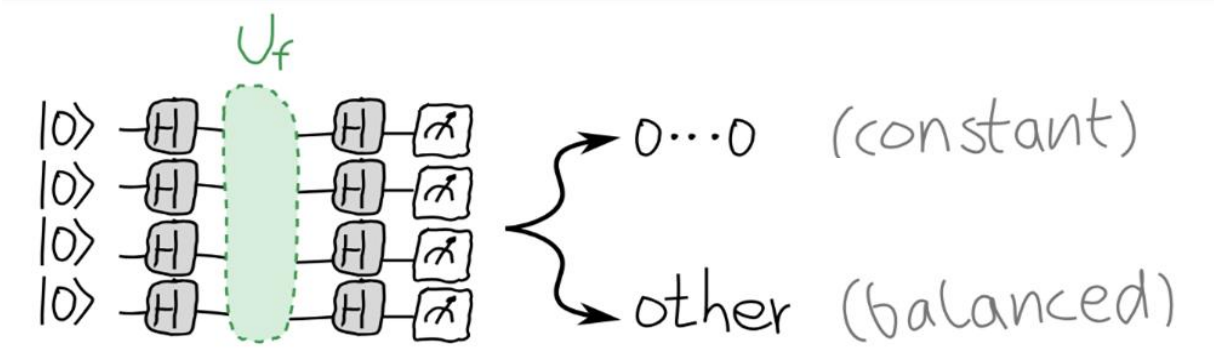
$f(x_1) = 1$ & $f(x_2) = 0$ $\rightarrow$ $\frac{1}{\sqrt{2}}(|x_2\rangle - |x_1\rangle)$

} distinguishable states.

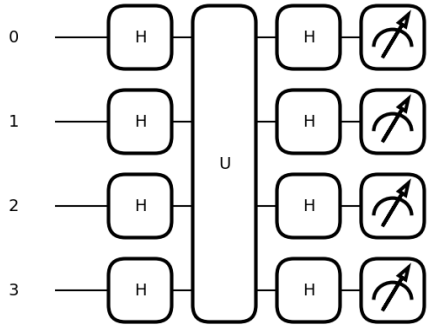
but if $f(x_1) = 0$ & $f(x_2) = 0$ $\rightarrow$ $\frac{1}{\sqrt{2}}(|x_1\rangle + |x_2\rangle)$

Deutsch-Jozsa

Here you have an oracle that tells you: is your function constant or balanced.



Exercise 4: Deutsch-Jozsa




```
def oracle_matrix(keys):
    """Create the unitary matrix corresponding to the binary key (list[int])
       e.g. key=[0,1,1] should give the diagonal matrix [1,1,1,-1,1,1,1,1]"""
    # Hint: use np.ravel_multi_index
    matrix = np.identity(2 ** n_bits)
    return matrix

@pl.qnode(dev)
def deutschjozsa(keys):
    """Build the Deutsch-Jozsa circuit"""
    return pl.probs(wires=range(n_bits))

keys = ([[0,0,0,0],[0,0,0,1],[1,0,0,0],[0,1,0,0],[1,1,0,0],[0,0,1,0],[0,0,1,1],[0,1,1,0]])
if np.isclose(deutschjozsa(keys)[0],0):
    print("balanced")
else:
    print("constant")
```

Bernstein-Vazirani

$$f(\vec{x}) = \sum_{i=0}^{n-1} a_i x_i \pmod 2.$$

What's the oracle here?

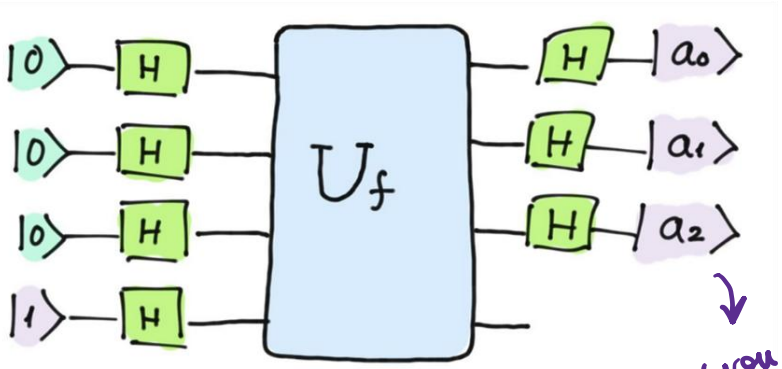
$U_f : |\vec{x}\rangle|y\rangle \longrightarrow |\vec{x}\rangle|y + \vec{a}\cdot\vec{x} \pmod 2\rangle$

ex: $\vec{a} = (0,1,0) \rightarrow \vec{a}\cdot\vec{x} = 1$

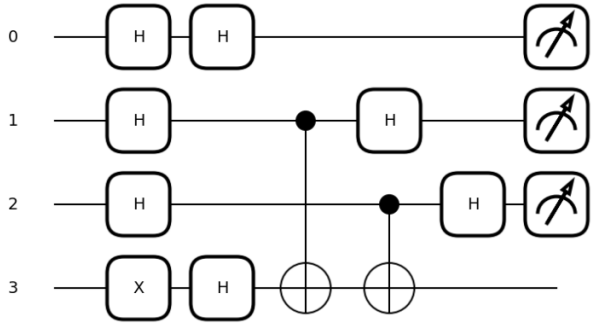
$\vec{x} = (1,1,1)$

$U_f|111\rangle|0\rangle = |111\rangle|1\rangle$

$U_f|111\rangle|1\rangle = |111\rangle|0\rangle$



Exercise 5: Bernstein-Vazirani




```
dev = pl.device("default.qubit", wires = 4, shots = 1)

def oracle():
    """Encode the hidden value in a circuit"""
    pl.CNOT(wires=[1, 3])
    pl.CNOT(wires=[2 ,3])

@pl.qnode(dev)
def bernsteinvazirani():
    """Sample the Bernstein-Vazirani circuit to return the hidden value"""
    return pl.sample(wires = range(3))

pl.draw_mpl(bernsteinvazirani)()
a = bernsteinvazirani()
print(f"The value of a is {a}")
```