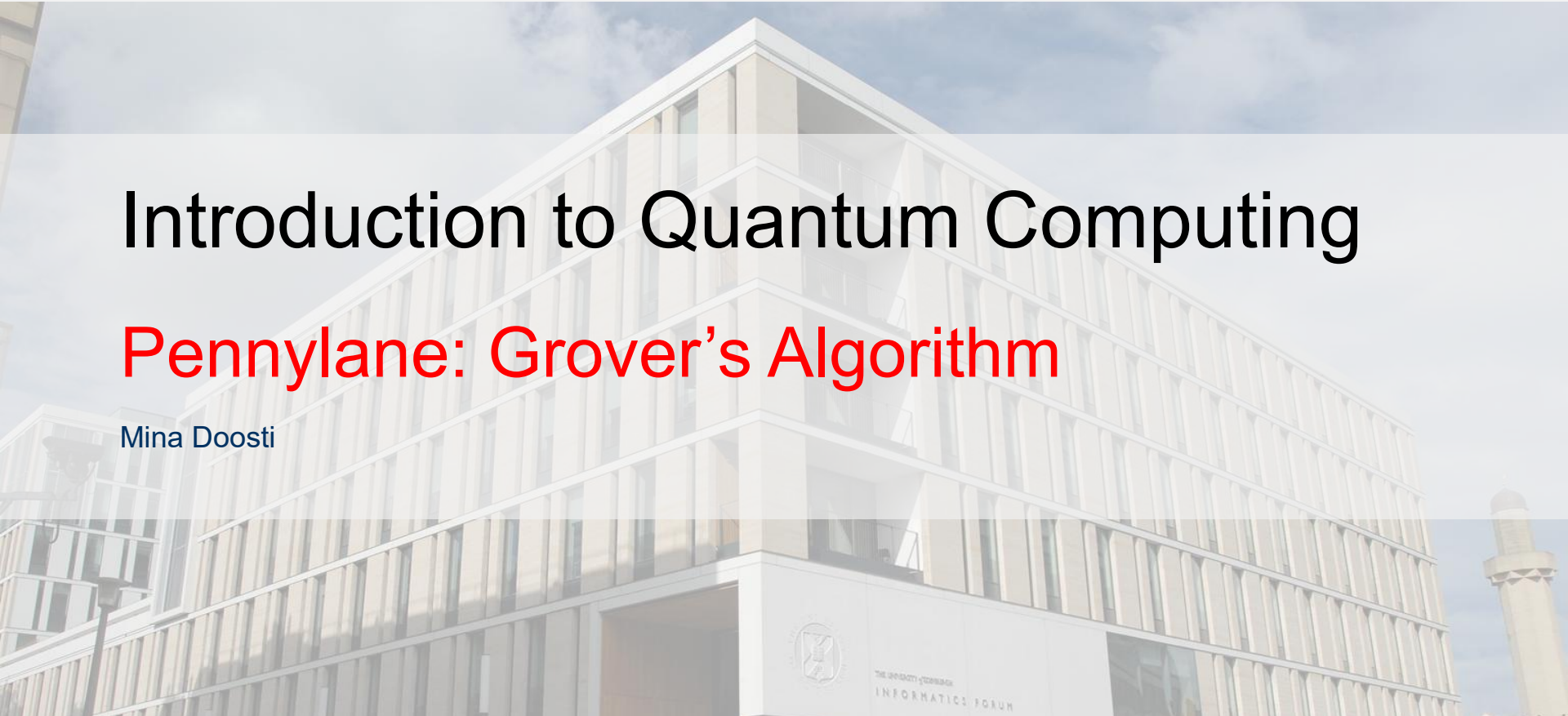


THE UNIVERSITY of EDINBURGH
informatics

Introduction to Quantum Computing

Pennylane: Grover's Algorithm

Mina Doosti

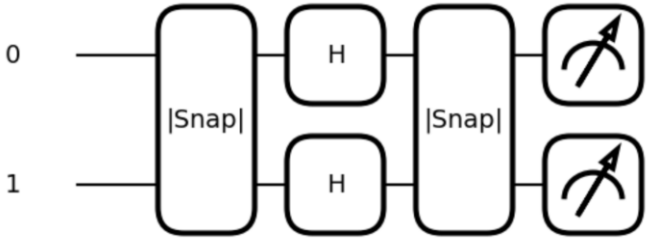
What does Grover's Algorithm do?

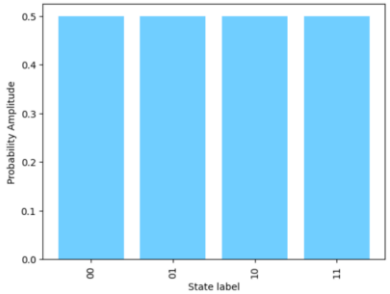
Search for a solution!
You have a promise: $f(x) : \{0,1\}^n \rightarrow \{0,1\}$

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

A bit like the lock picking we saw last time.
We use again superposition and oracle
But we also need to use a new trick called:
Amplitude Amplification

Exercise 1: Uniform superposition






```
n_bits = 2
dev = pl.device("default.qubit", wires=n_bits)
wires = list(range(n_bits))

@pl.qnode(dev)
def circuit():
    pl.Snapshot("Initial state")
    # build a uniform superposition
    pl.Snapshot("After applying the Hadamard gates")
    return pl.probs(wires=wires)

pl.drawer.use_style("black_white")
pl.draw_mpl(circuit)();
results = pl.snapshots(circuit)()
for k, result in results.items():
    print(f"{k}: {result}")
```

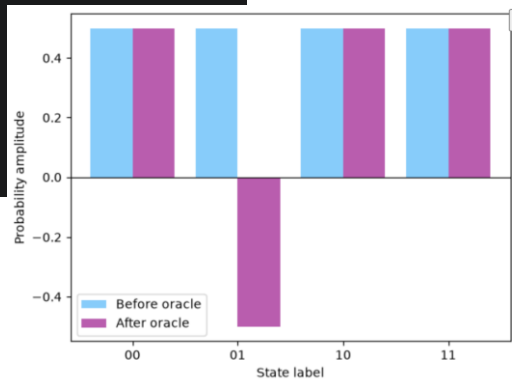
Exercise 2: Oracle as a reflection

$O_\omega = I - 2|\omega\rangle\langle\omega|$

↳ As a phase flip ⟹ $U_f|x\rangle|-\rangle = (-1)^{f(x)}|x\rangle|-\rangle$

```
keys = [[0,1]]  # one marked key
flag_wire = n_bits
dev = pl.device("default.qubit", wires=n_bits + 1)

def oracle(keys):
    # build the reversible checker for keys[0]
    # apply X to the data wires where the marked bit is 0
    # apply a controlled X to the extra (flag) qubit
    # undo the first X gates
    pass
```



Apply it to uniform superposition

Exercise 3: Diffusion Operator

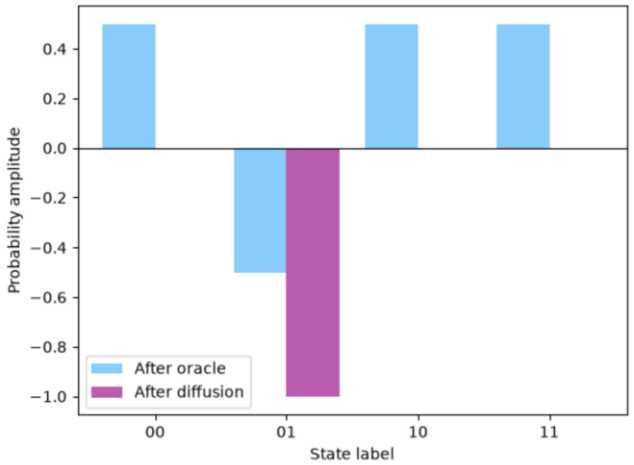
$D = 2|s\rangle\langle s| - I$ $\quad\quad$ $|s\rangle = H^{\otimes n}|0\cdots0\rangle$

```
def diffusion_operator(wires):
    # build the diffusion operator circuit
    pass
```

Exercise 4: One Grover Iteration

```python
@pl.qnode(dev)
def circuit():
    for i in range(n_bits):
        pl.Hadamard(wires=i)
    pl.PauliX(wires=flag_wire)
    pl.Hadamard(wires=flag_wire)
    pl.Snapshot("Uniform superposition")
    oracle(keys)
    pl.Snapshot("State marked by oracle")
    diffusion_operator(wires)
    pl.Snapshot("Amplitude after diffusion")
    return pl.probs(wires=wires)

results = pl.snapshots(circuit)()
# inspect the results; plot the amplitudes after the oracle and after diffusion
```



Exercise 5: Full Grover over multiple iterations

```python
n_bits = 4
wires = list(range(n_bits))
flag_wire = n_bits
dev = pl.device("default.qubit", wires=n_bits + 1)
keys = [[0,1,0,1]]
N = 2**n_bits
M = 1
theta = np.arcsin(np.sqrt(M / N))
r_opt = int(np.round(np.pi / (4 * theta) - 0.5))
print("Suggested Grover iterations:", r_opt)
```

```python
@pl.qnode(dev)
def circuit(iterations):
    # build the grover circuit, using the oracle and diffusion operator above
    # iterate it "iterations" times
    return pl.probs(wires=wires)
```

