CADT
IDT វិទ្យាស្ថានបច្ចេកវិទ្យាឌីជីថល
Institute of Digital Technology

Quantum Computing

From Theory to Practical

Gen 9 SE Y4 2025 (G1, G2, & G3)

Lecturer: **Ms. HENG Sovanmonynuth**

Software Engineering (SE), Faculty of Digital Engendering, Department of Computer Science

Cambodia Academy of Digital Technology (CADT)

Email: sovanmonynuth.heng@cadt.edu.kh

Mobile: (+855) 95 479 272

Course Outline

Chapter 1: Introduction to Quantum Computing

Chapter 2: Basic of Quantum Mechanics

Chapter 3: Quantum Algorithms

Chapter 4: Quantum Fourier Transform

Chapter 5: Quantum Errors & Noises

Chapter 6: Quantum Machine Learning

Chapter 7: Quantum Chemistry

Chapter 8: Quantum Cryptography

Chapter 9: Quantum Information Science

Chapter 10: Project Development & Project Report

Final Presentation

Chapter 4: Quantum Fourier Transform

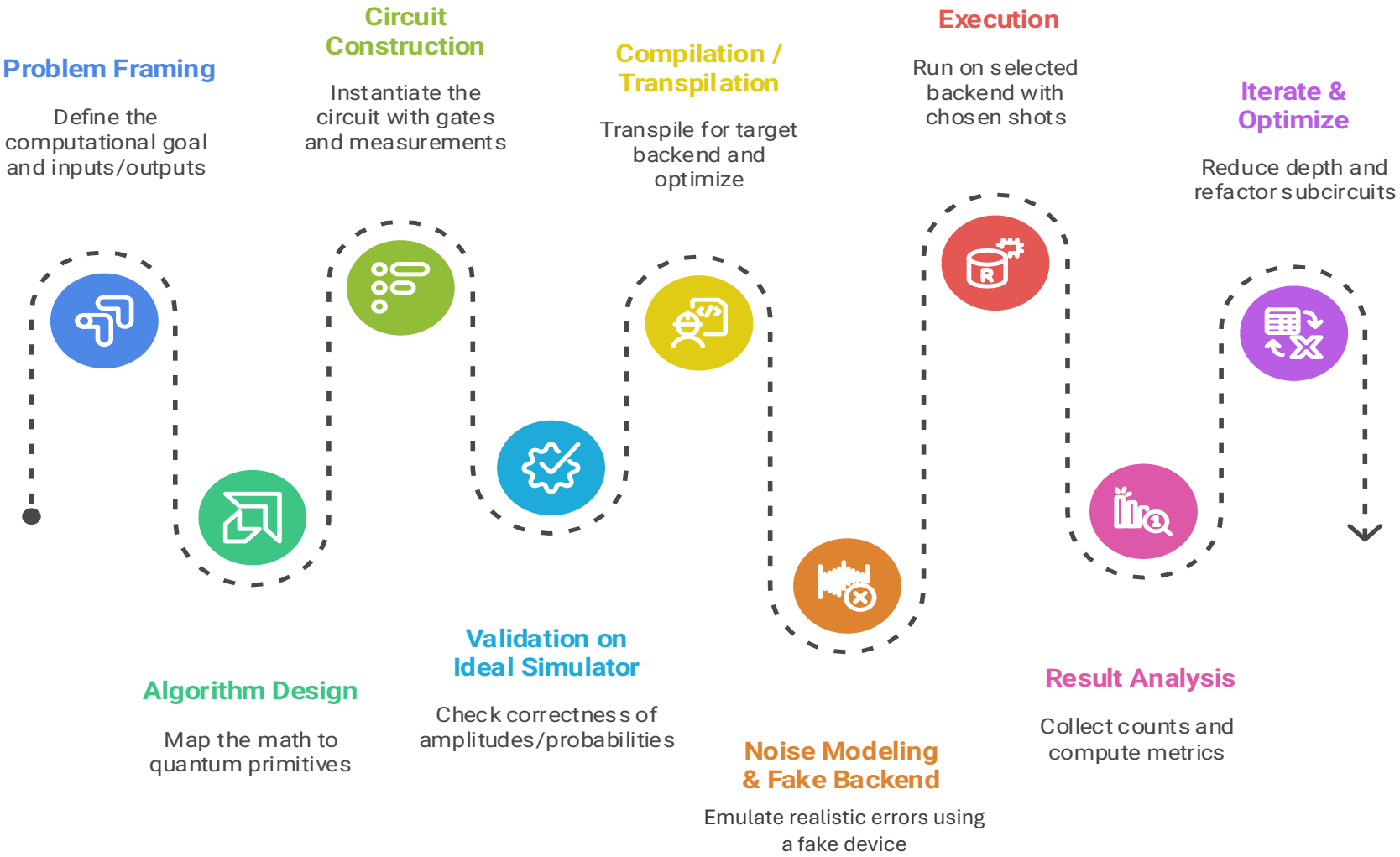
- What is Quantum Algorithm?
- Quantum Algorithm Development Process
- Quantum Fourier Transform (QFT)
- Bernstein-Vazirani (BV) Algorithms
- Quantum Phase Estimation (QPE) Algorithms
- Grover's Search Algorithm
- Shor’s Factoring Algorithm
- Google Colob: Jupyter Notebook

What is Quantum Algorithm?

- **What is a Quantum Algorithm?**
- A **quantum algorithm** is a step-by-step procedure that uses **quantum bits (qubits)** and **quantum gates** to exploit uniquely quantum effects—**superposition, interference, and entanglement**—to solve certain problems with different (sometimes better) complexity than classical methods.

It's a **program** that choreographs complex probability amplitudes so that ***right answers interfere constructively*** and ***wrong answers cancel out***.

Quantum Algorithm Development Process



Game Title: **“Find the Hidden Beat!”**

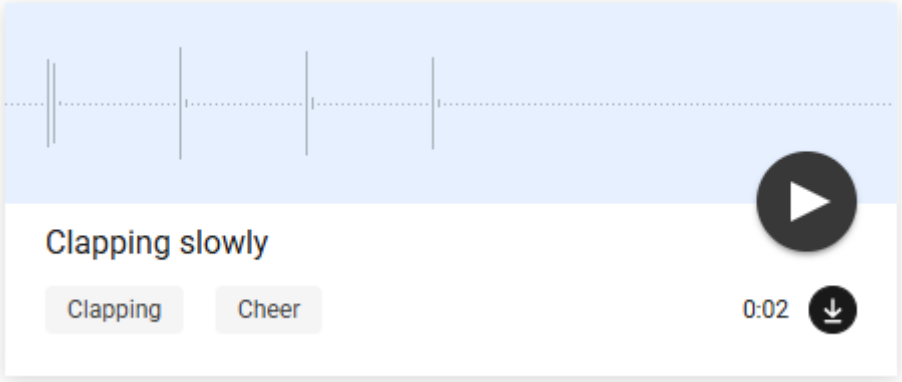
Easy

- **Goals:** Students will **discover the “hidden period”** in a repeating pattern — just like the **QFT** finds **hidden periodicities** in **quantum data**.

Find hidden beat!

How many beat did you hear?

3 Clap-Clap-Clap (period = 3)



Sound Effect Link: https://mixkit.co/free-sound-effects/light/

Game Title: **“Find the Hidden Beat!”**

Medium

- **Goals:** Students will **discover the “hidden period”** in a repeating pattern — just like the **QFT** finds **hidden periodicities** in **quantum data**.

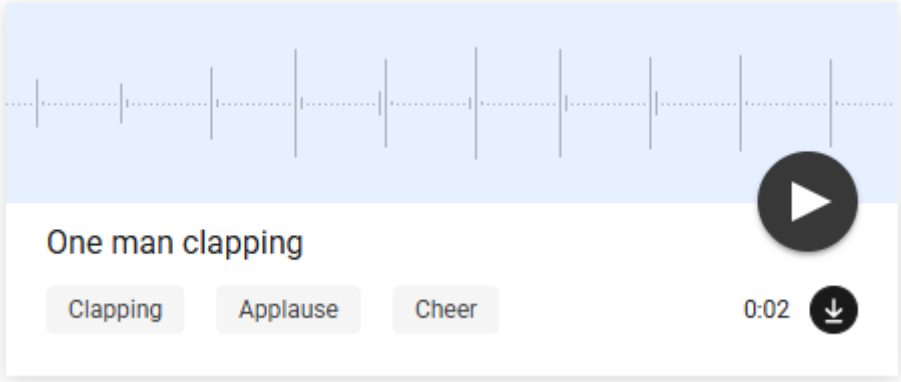
Find hidden beat!

How many beat did you hear?

10

Clap-Clap-Clap-Clap-Clap-
Clap-Clap-Clap-Clap-Clap

(period = 10)



Sound Effect Link: https://mixkit.co/free-sound-effects/light/

Game Title: **“Find the Hidden Beat!”**

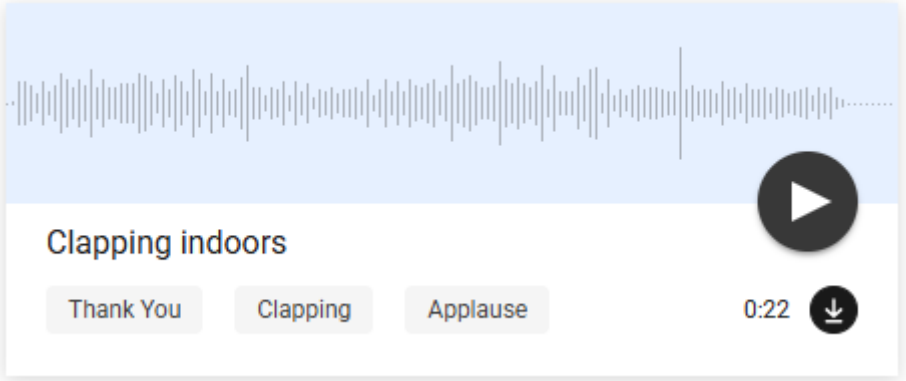
Hard

- **Goals:** Students will **discover the “hidden period”** in a repeating pattern — just like the **QFT** finds **hidden periodicities** in **quantum data**.

Find hidden beat!

How many beat did you hear?

N beats



Sound Effect Link: https://mixkit.co/free-sound-effects/light/

Game Title: **“Find the Hidden Beat!”**

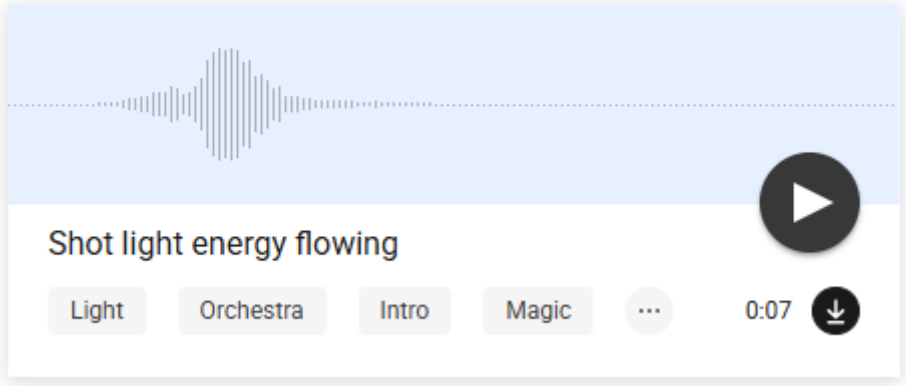
Hard

- **Goals:** Students will **discover the “hidden period”** in a repeating pattern — just like the **QFT** finds **hidden periodicities** in **quantum data**.

Find hidden beat!

How many beat did you hear?

H beats



Sound Effect Link: https://mixkit.co/free-sound-effects/light/

Quantum Fourier Transform (QFT)

Classical FT → QFT definition

- The **Fourier Transform (FT)** expresses a signal as a sum of sinusoids with different frequencies.
- The **Discrete Fourier Transform (DFT)** for *N* elements transforms vector $x = (x_0, ., .. x_{N-1})$ into another vector $y = (y_0, ., .. y_{N-1})$:

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

- In **quantum computing**, amplitudes α_j of basis states $|j\rangle$ are analogous to classical coefficients x_j.

  The **Quantum Fourier Transform (QFT)** maps: $|j\rangle \rightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i j k / N} |k\rangle$

  where $N = 2^n$ for n qubits.

Quantum Fourier Transform (QFT)

Classical FT → QFT definition

- It changes the **computational basis** into the **Fourier basis**.
- Each basis state **| j⟩** becomes a superposition with phase rotations.
- QFT provides **exponential compression** — operating on all amplitudes ***simultaneously*** via **superposition**.

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2\pi i j k / N} \qquad |j\rangle \rightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i j k / N} |k\rangle$$

- **Note**: The matrix is **unitary**, meaning $U_f^{\dagger} U_f = I$

Quantum Fourier Transform (QFT)

QFT Circuit design

- **Goal:** helps us **analyze the periodicity (repeating patterns)** encoded in quantum states.
- Construction Steps (for n qubits)
  - For **each qubit j^{th}** from **0 to n−1**:
    - Apply **H** to qubit j^{th}
    - For ***each subsequent*** qubit k > j:
      - Apply **Controlled-$R_{k, j+1}$** between qubit k (control) → j (target)
  - Apply **bit reversal (swap)** to reorder qubits

- **QFT** does exactly that — but instead of sound waves, it detects **hidden periodic patterns** in **quantum amplitudes**.

Quantum Fourier Transform (QFT)

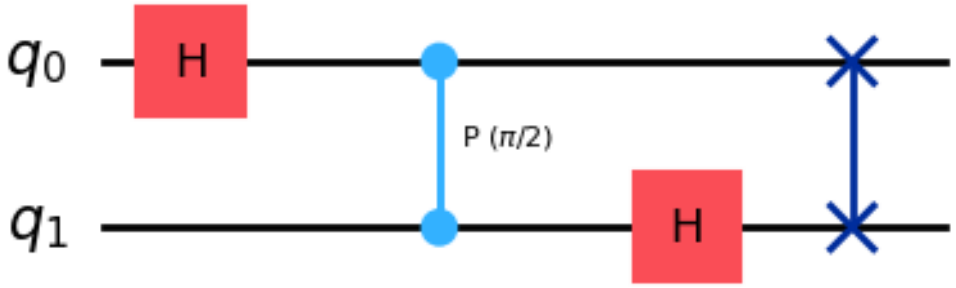
QFT Example

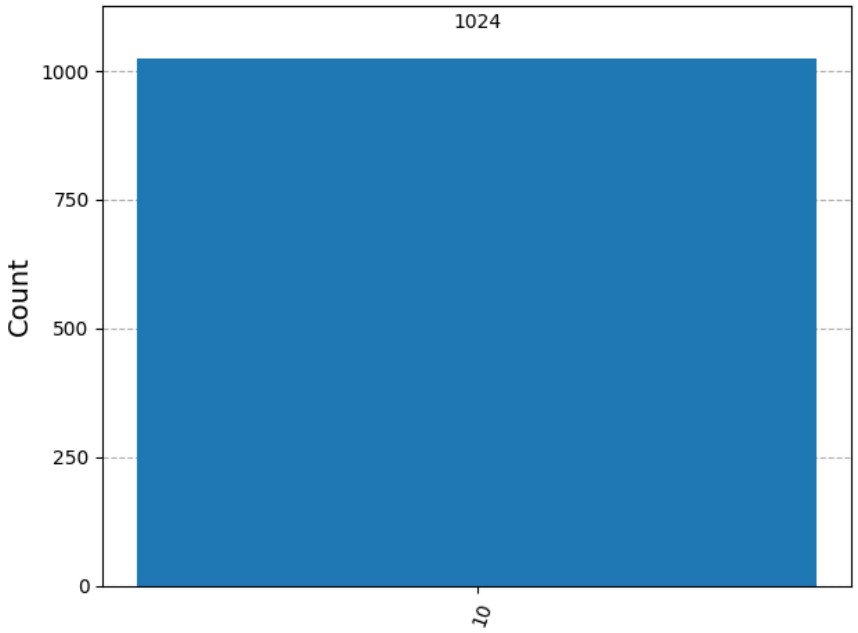
- **Goal:** We want to find the ***hidden period*** of a repeating number pattern.
  - Let's use a simple repeating sequence: **[1, 2, 3, 1, 2, 3, 1, 2, 3, …]**
- So the **true period = 3** (because after every 3 steps, it repeats).

Quantum Fourier Transform (QFT)

2-Qubit QFT Circuit design

- Apply H to q_0
- Apply Controlled-R from $q_1 \rightarrow q_0$
- Apply H to q_1
- Swap q_0 and q_1





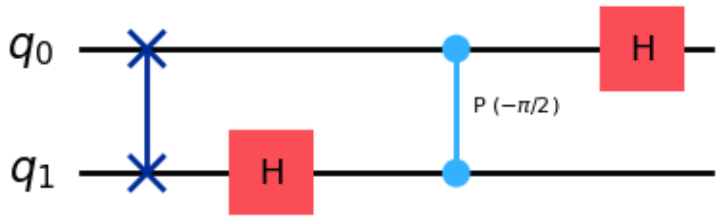
Quantum Fourier Transform (QFT)

Inverse QFT (QFT^{-1})

- **Definition**
  - **Same as QFT** but with **negative phase rotations**.
  - Circuit = reverse order of QFT gates, each phase gate inverted.

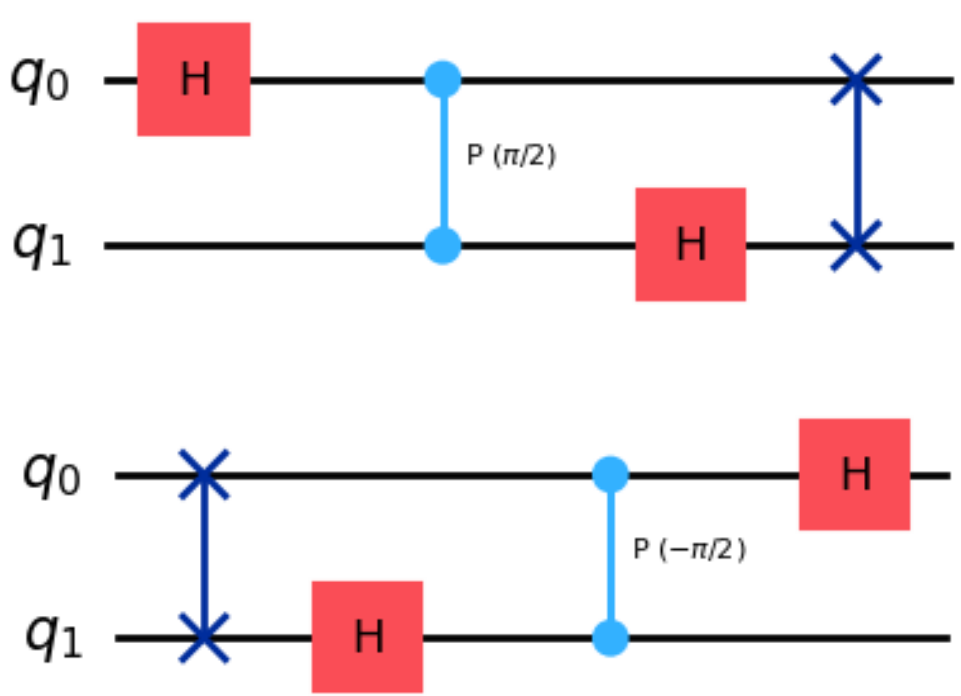
$$QFT^{-1}|k\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{-2\pi i jk/N} |j\rangle$$

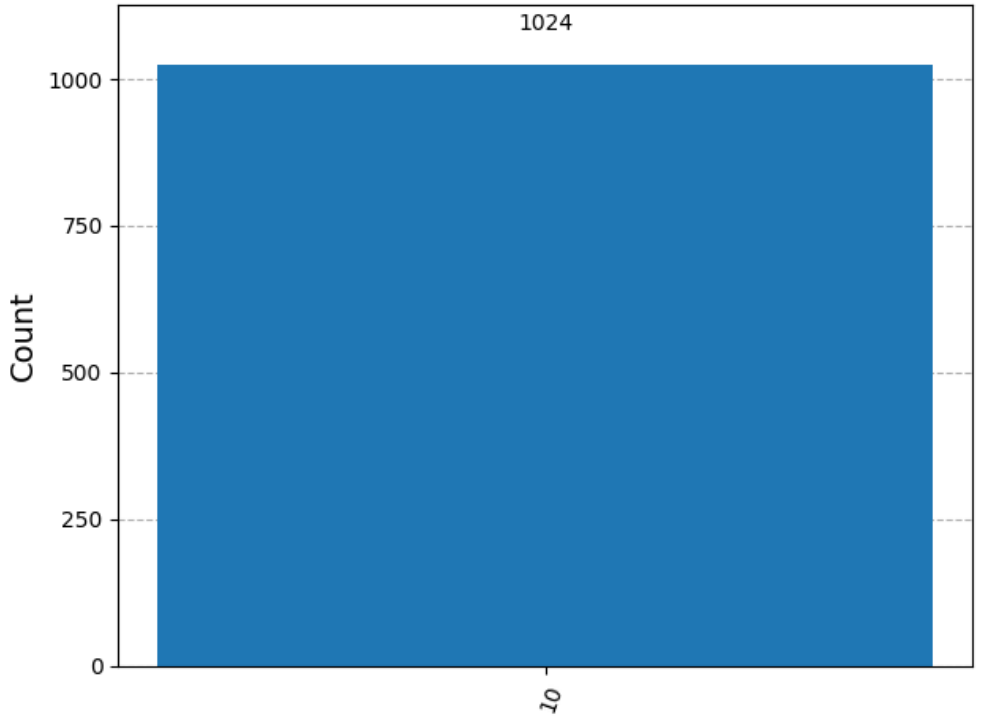
- Construction Steps
  - Reverse gate sequence of **QFT**
  - Replace each R_k by $\mathbf{R_k^\dagger}$ (negative angle)
  - Perform **swaps** at end again
- **Note:** $R_k R_k^\dagger$ = I



Lab Demo

- Lab: build **2-qubit QFT** and **verify inverse QFT**



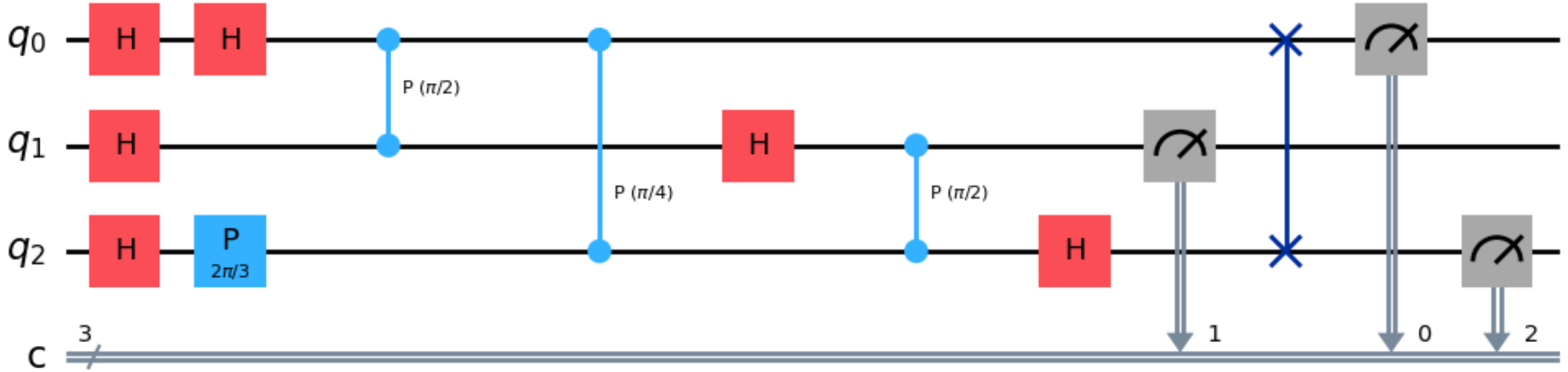


Lab Practice

- Build **3-Qubit QFT** with the period(s):
  - **Step 1: Define QFT**
  - **Step 2: Create hidden period**
  - **Step 3: Build the 3-Qubits QFT circuit**
  - **Step 4: Simulate the result**
  - **Step 5: Visualization the result using histogram**

Lab Practice

- Build **3-Qubit QFT** with the period of 2, 3, & 4
  - **Step 1: Define QFT**
  - Step 2: Create hidden period
  - Step 3: Build the cull circuit
  - Step 4: Simulate
  - Step 5: Visualization



Quantum Fourier Transform (QFT)

Does the *number of qubit* matter to this period?

- Simple repeating sequence: **[1, 2, 3, 1, 2, 3, 1, 2, 3, …]**
- **Yes**, because it effects the **search space** and **accuracy of the result**.
- When you run QFT on n **qubits**, you're effectively working in a **space of size** $N = 2^n$.

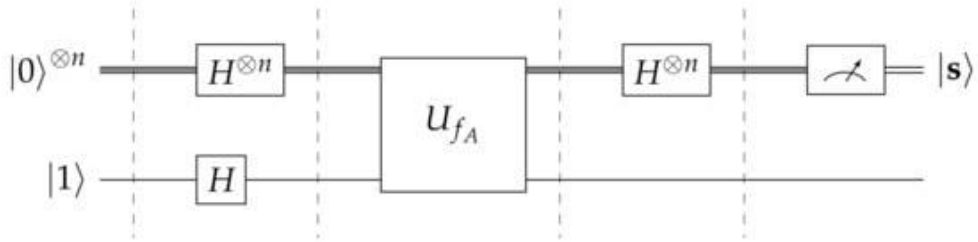
Qubits (n)	$N = 2^n$	Period (r)	N/r
1	2	4	0.5
2	4	4	1
3	8	4	2
4	16	4	4
5	32	4	8

- **What if we don't know the hidden period?**

  [1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 1, 2, 3, 1, 2, 3, 1, 2, 3

Bernstein–Vazirani Algorithm

Bernstein–Vazirani (BV) Algorithm



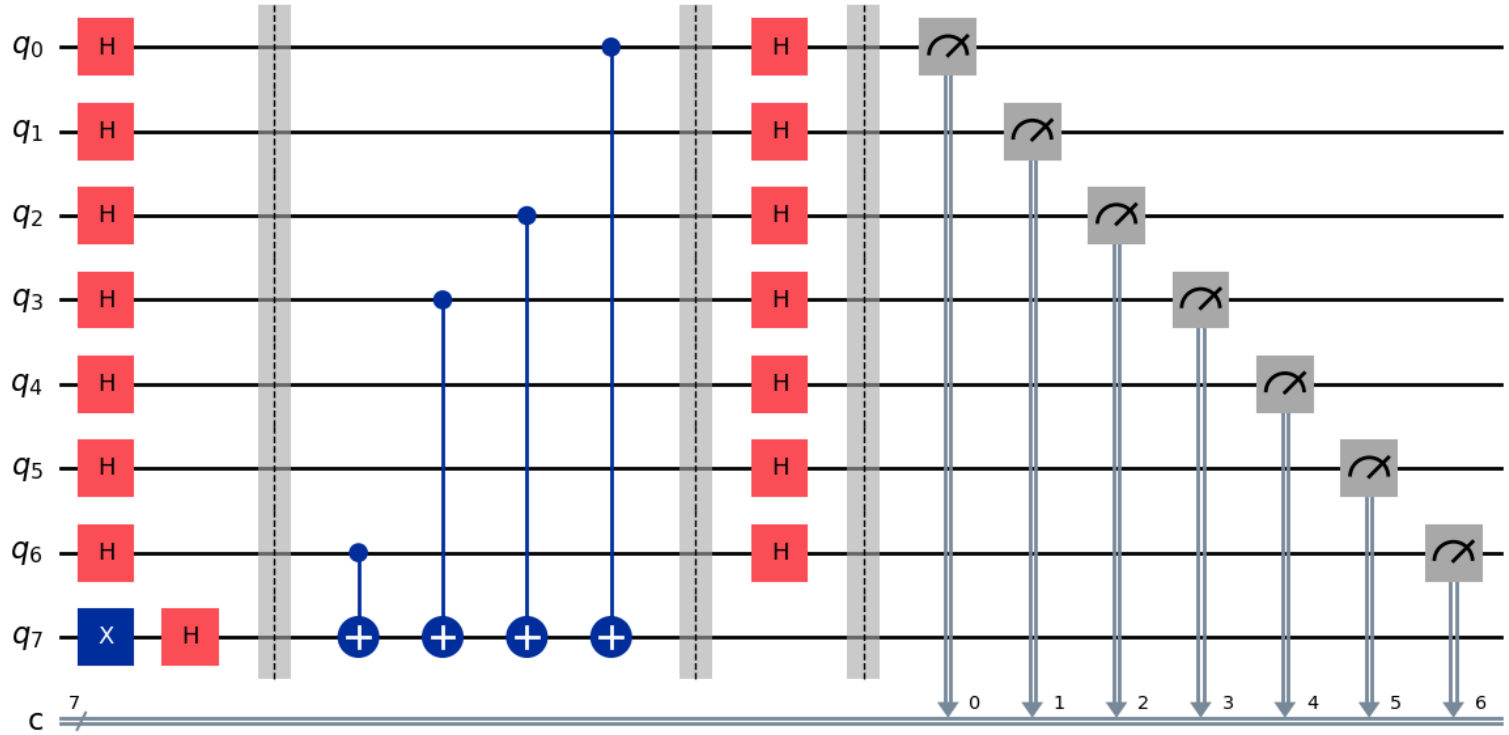
- **Goal:** Find hidden bit-string a in f(x)=a·x (mod 2).
- **Classical:** n queries; **Quantum:** 1 query.
- **Quantum Circuit Construction:**
  - Initialize $|0\rangle^n|1\rangle$ → apply H to all.
  - Apply Uf (phase kickback encodes *a* into *phase A*).
  - Apply H again, then measure first n qubits → reveals A.
- **Key Idea:** Quantum Phase Kickback - information moves from target to control qubits.

- **Notes:** BV algorithm can be referred to as an extension of Deutsch-Jozsa algorithm. In both cases, we depend on an oracle-based model.

Bernstein–Vazirani Algorithm

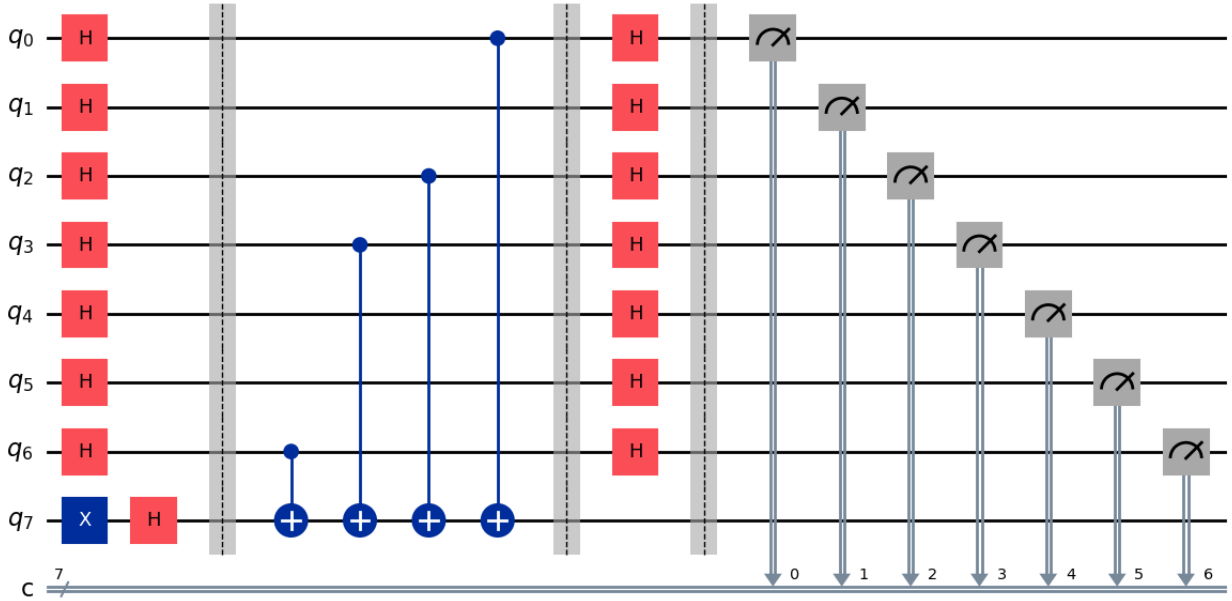
Bernstein–Vazirani (BV) Algorithm

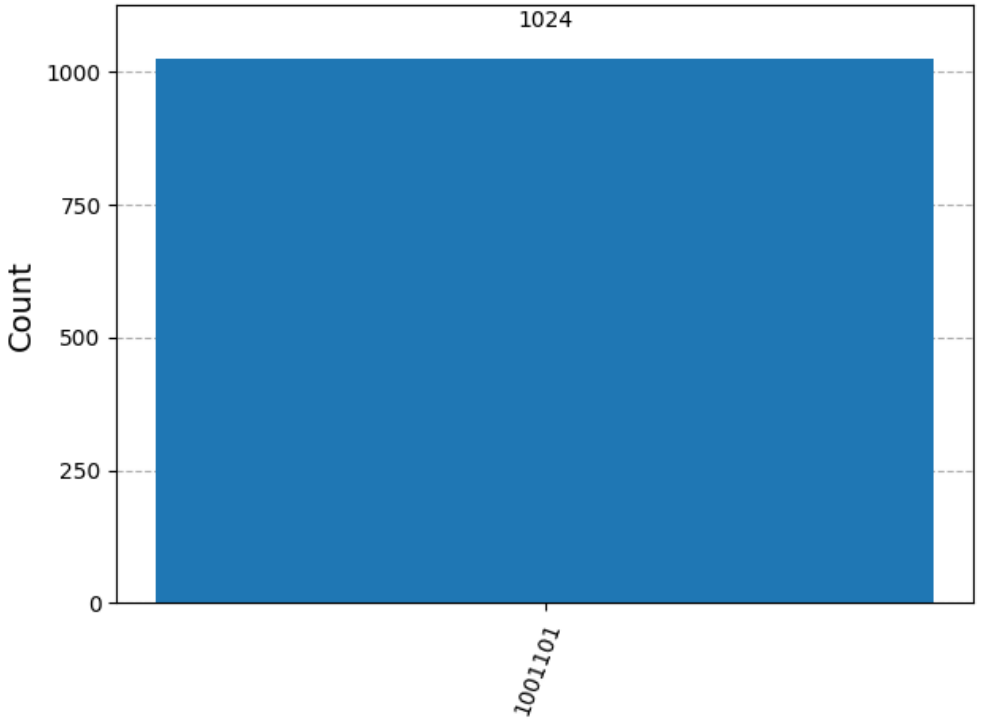
Let us first solve a number-specific example. As *secret_number* = ***'1011001'*** (7-bit long string), we need to initialize a quantum circuit with 7+1 qubits and 7 classical bits where we will store the results.



Lab Demo

- Build **BV-8** with:
  - **Step 1: Define BV-8**
  - **Step 2: Simulate outputs**

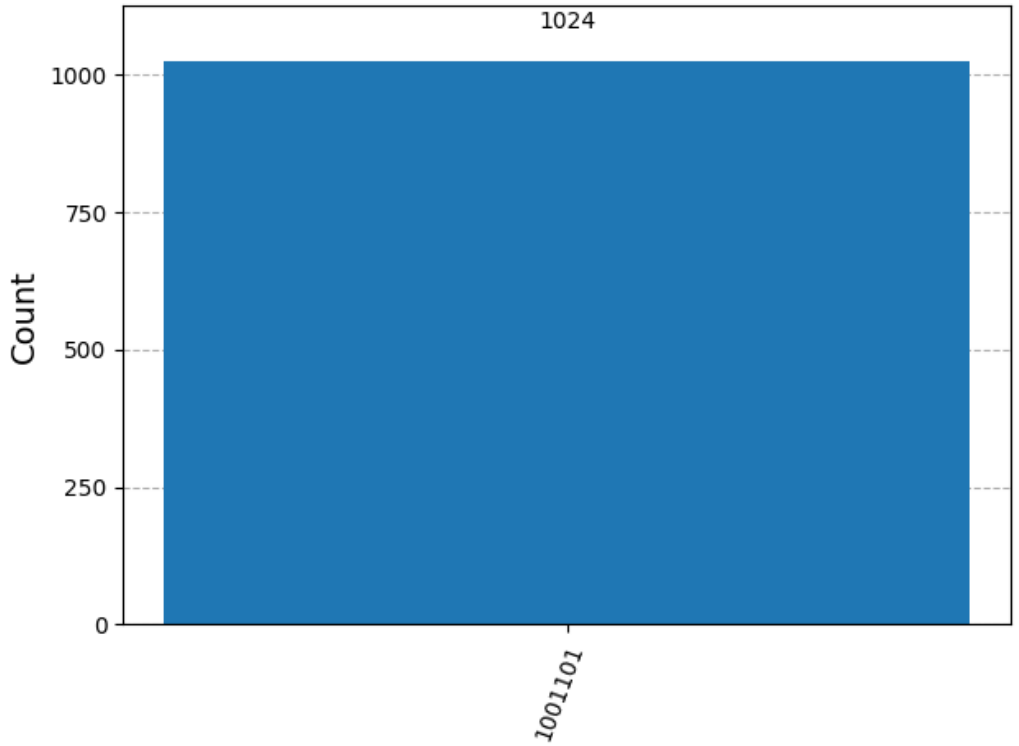




Lab Practice

- Build **BV-8** with:
  - **Step 1: Define BV-8 (7 max index)**
  - **Step 2: Simulate outputs**
  - **Step 3: Let user input *secret_number***
  - **Step 4: Re-define your BV-8 algorithm**
  - **Step 5: Simulate outputs**

  (Can you find the correct output?)

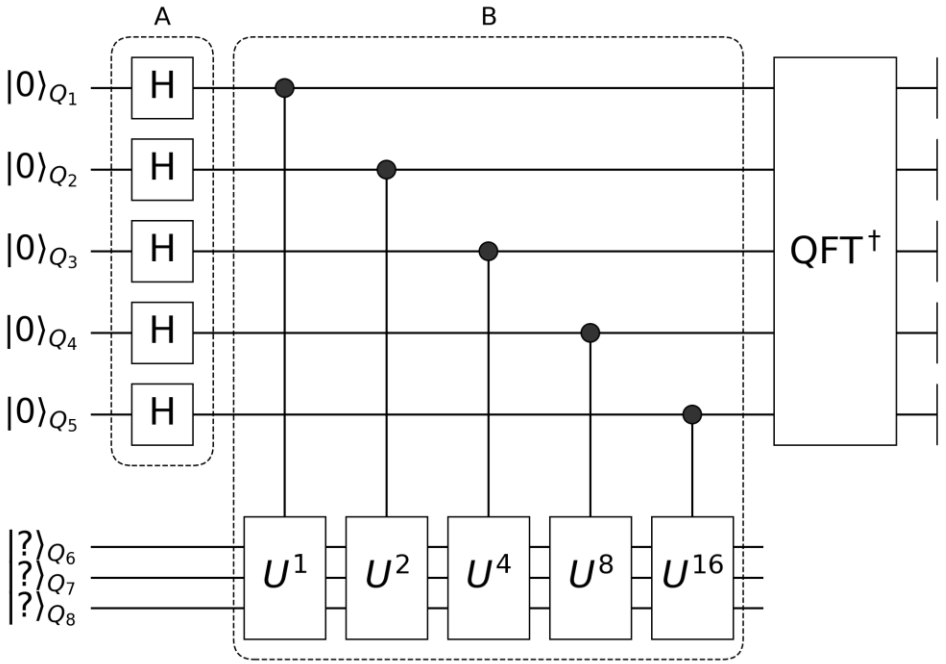


Quantum Phase Estimation (QPE)

Quantum Phase Estimation (QPE) Algorithm

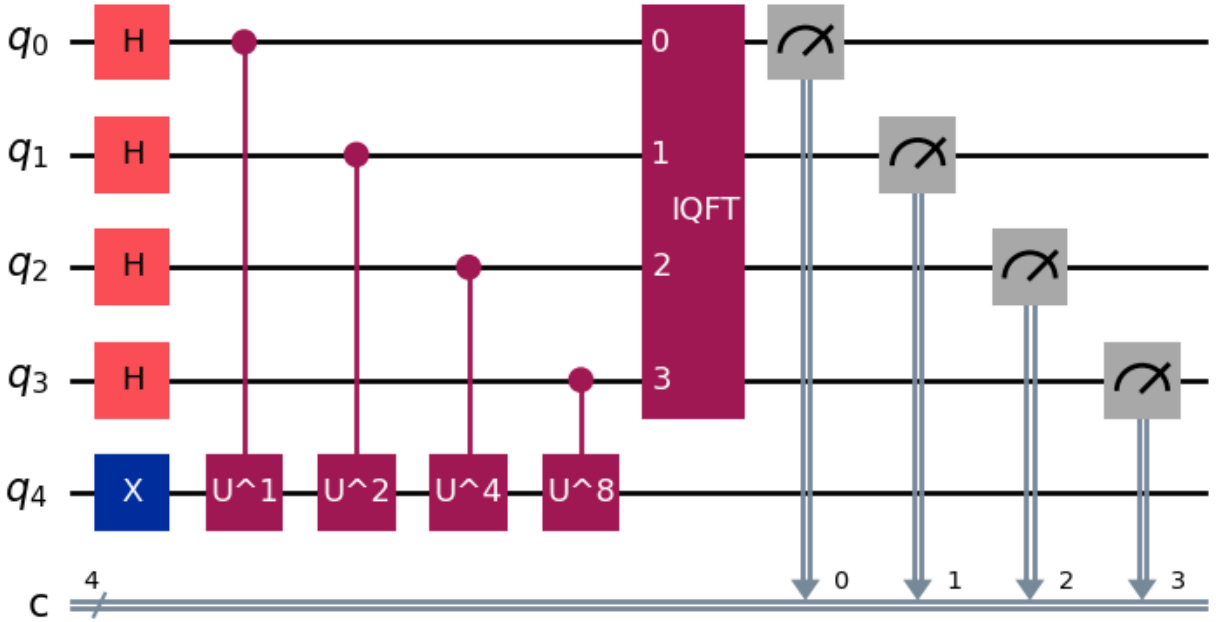
- **Goal:** Estimate the **phase ϕ** in the eigenvalue equation $U|\psi\rangle=e^{2\pi i\phi}|\psi\rangle$ where U is a unitary operator and $|\psi\rangle$ is one of its eigenstates.
- **Classical:** requires $O(2^n)$; **Quantum:** achieved efficiently using **O(n)** qubits.
- **Quantum Circuit Construction:**
  - Initialize registers: $|\mathbf{0}\rangle^{\otimes n} \otimes |\psi\rangle$
  - Apply $H^{\otimes n}$ to the first (control) register
  - For each control qubit j (from 0 to $n-1$):
    - Apply U^{2^j} to the target register only

    (if control qubit **j= 1**)
  - Apply **inverse QFT**

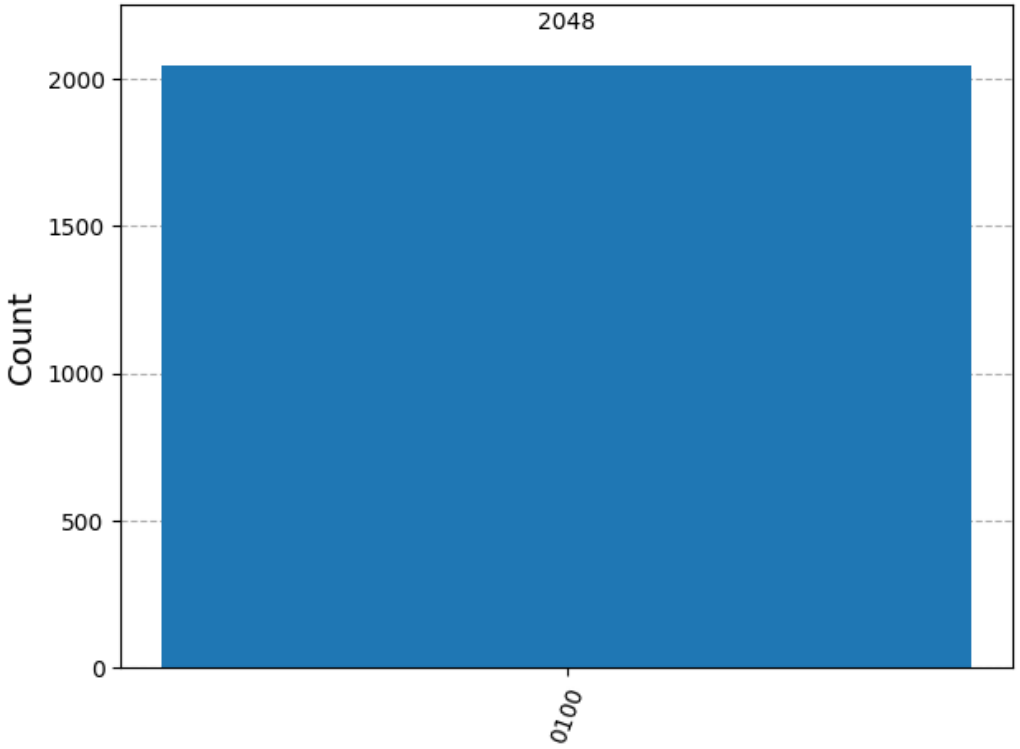


Lab Demo & Practice

- Build **QPE** with:
  - **Step 1: Define phase to estimate**
  - **Step 2: Build the QPE circuit**
  - **Step 3: Run the simulation**
  - **Step 4: Interpret the Results**




```
Actual phase: 0.25
Measured binary: 0100
Estimated phase: 0.25
```



Grover's Search Algorithm

Grover's Search Algorithm

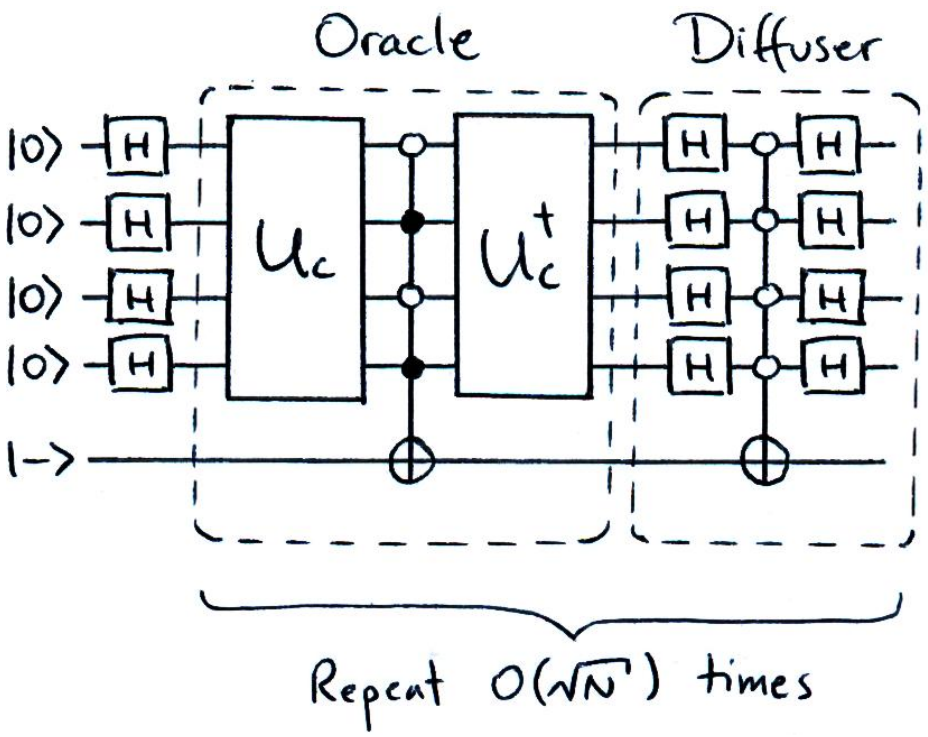
- **Goal :** Find a marked element among N possibilities f(x)=1 for solution x_0.

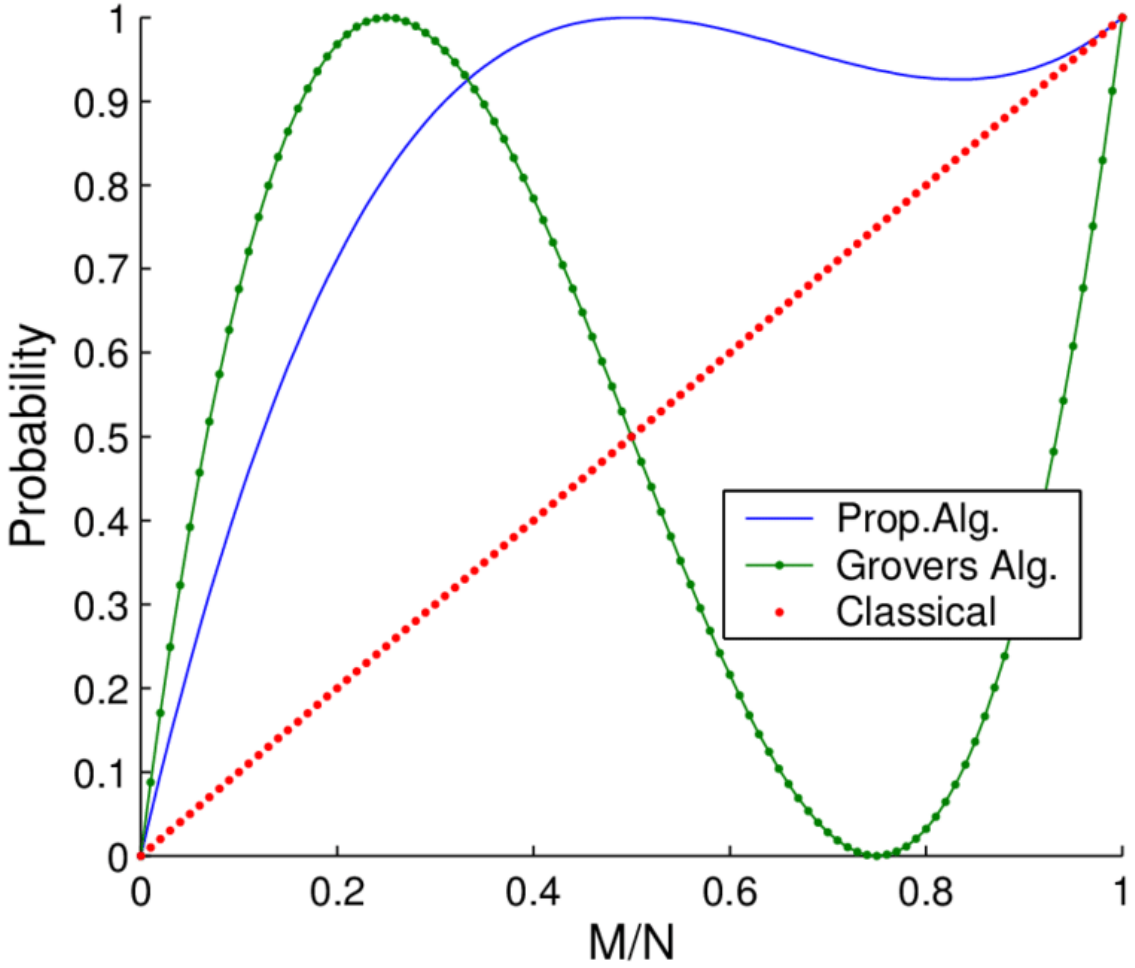
$$f(x) = \begin{cases} 1 & \text{if } x = x_0 \\ 0 & \text{otherwise} \end{cases}$$

- **Classical :** O(N) queries. **Quantum**: O(√N) queries — a **quadratic speedup**.
- **Core Steps:**
  - Initialize n-qubit state $|0\rangle^{\otimes n}$.
  - Apply **Hadamard** gates to all qubits.
  - Apply **oracle U_f** to flip the phase of the marked state.
  - Apply **diffusion** to amplify the marked state's amplitude.
  - Measure the state → output corresponds to the marked element x_0.

Grover's Search Algorithm

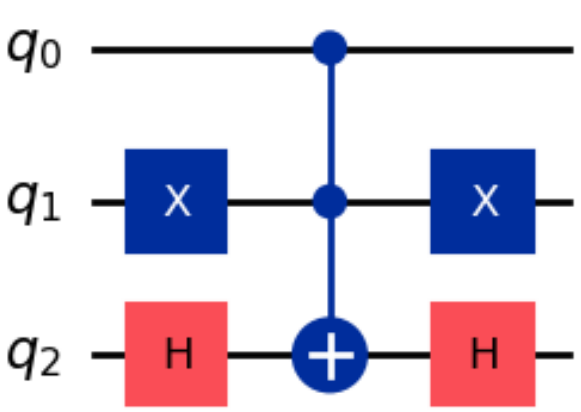
Grover's Search Algorithm



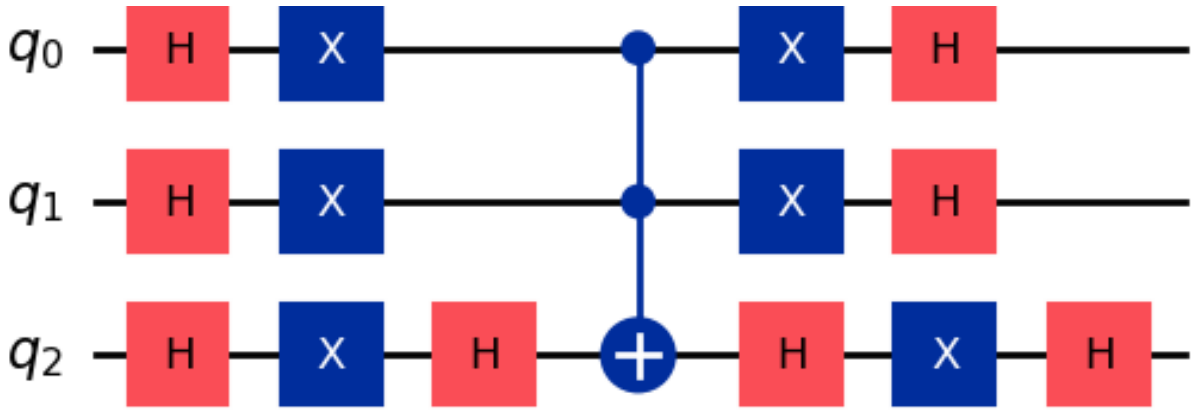


Lab Demo & Practice

- Build **Grover** with:
  - **Step 1: Define basic parameters**
  - **Step 2: Define oracle of the circuit**
  - **Step 3: Define diffuser of the circuit**
  - **Step 4: Build Full Grover Circuit**
  - **Step 5: Run the simulation**



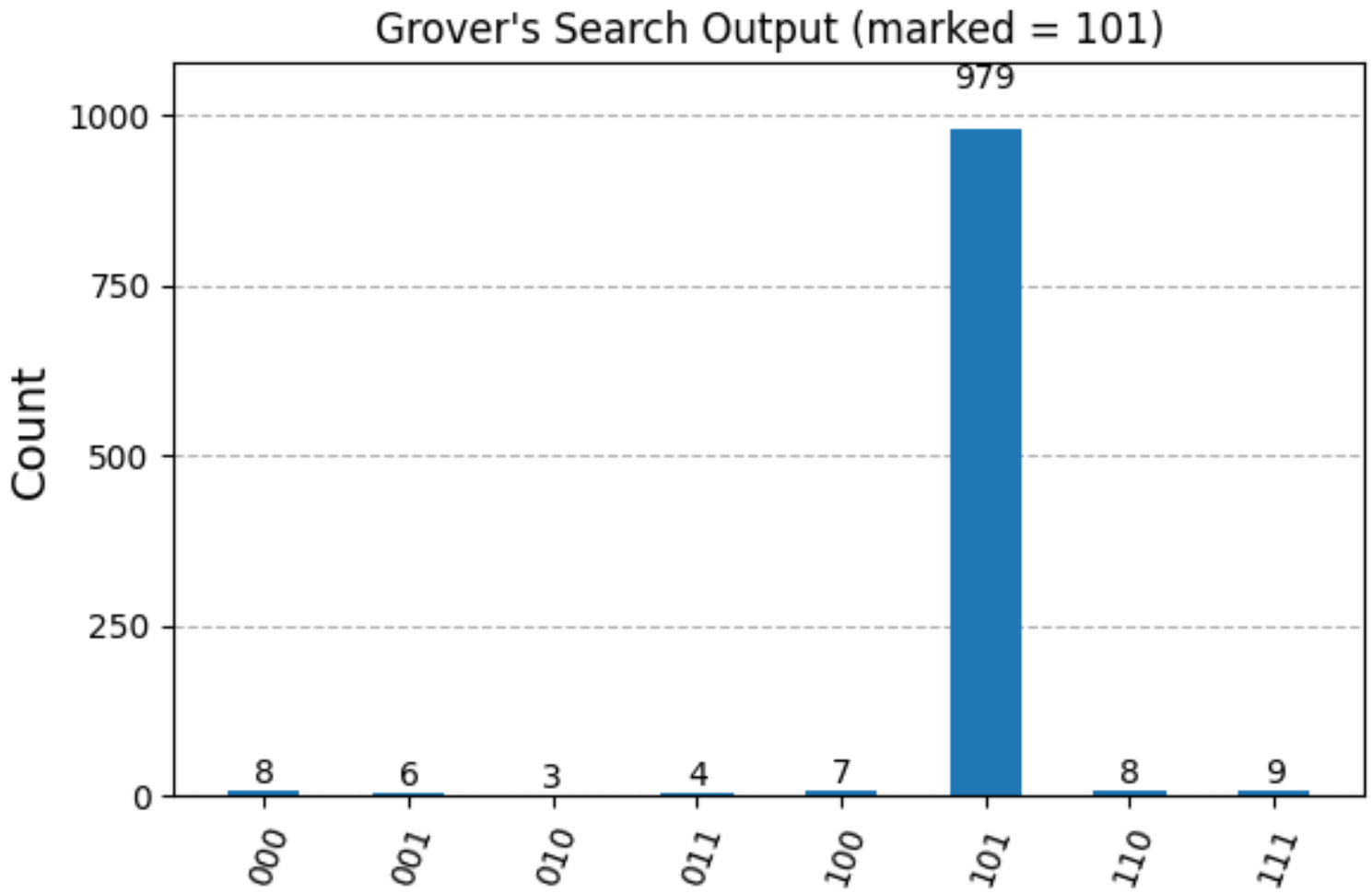
Oracle



Diffuser

Lab Demo & Practice

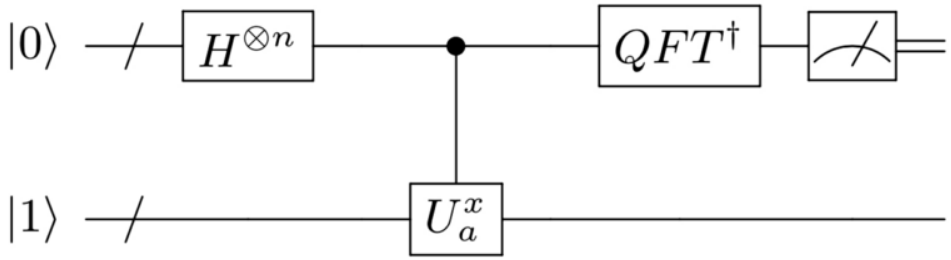
Grover’s Search Algorithm



Shor's Factoring Algorithm

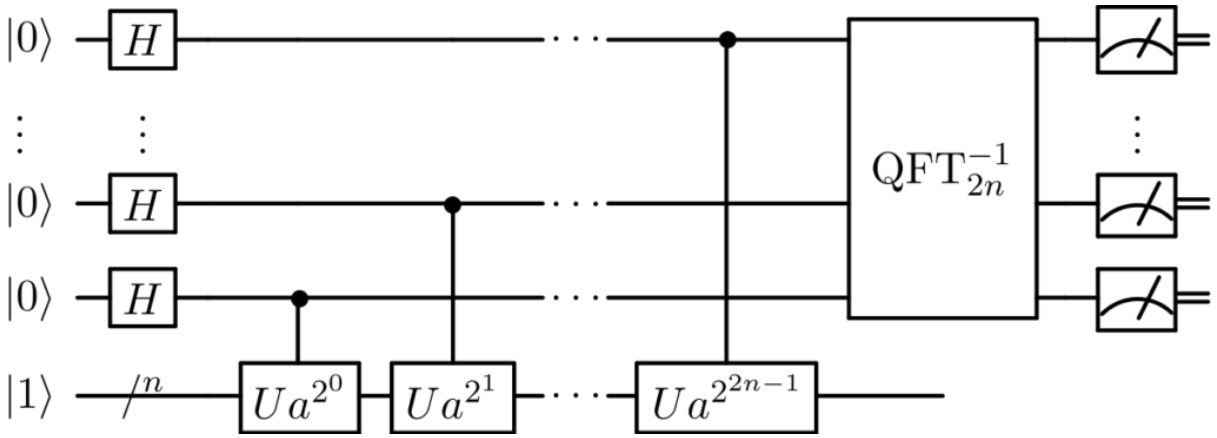
Shor's Factoring Algorithm

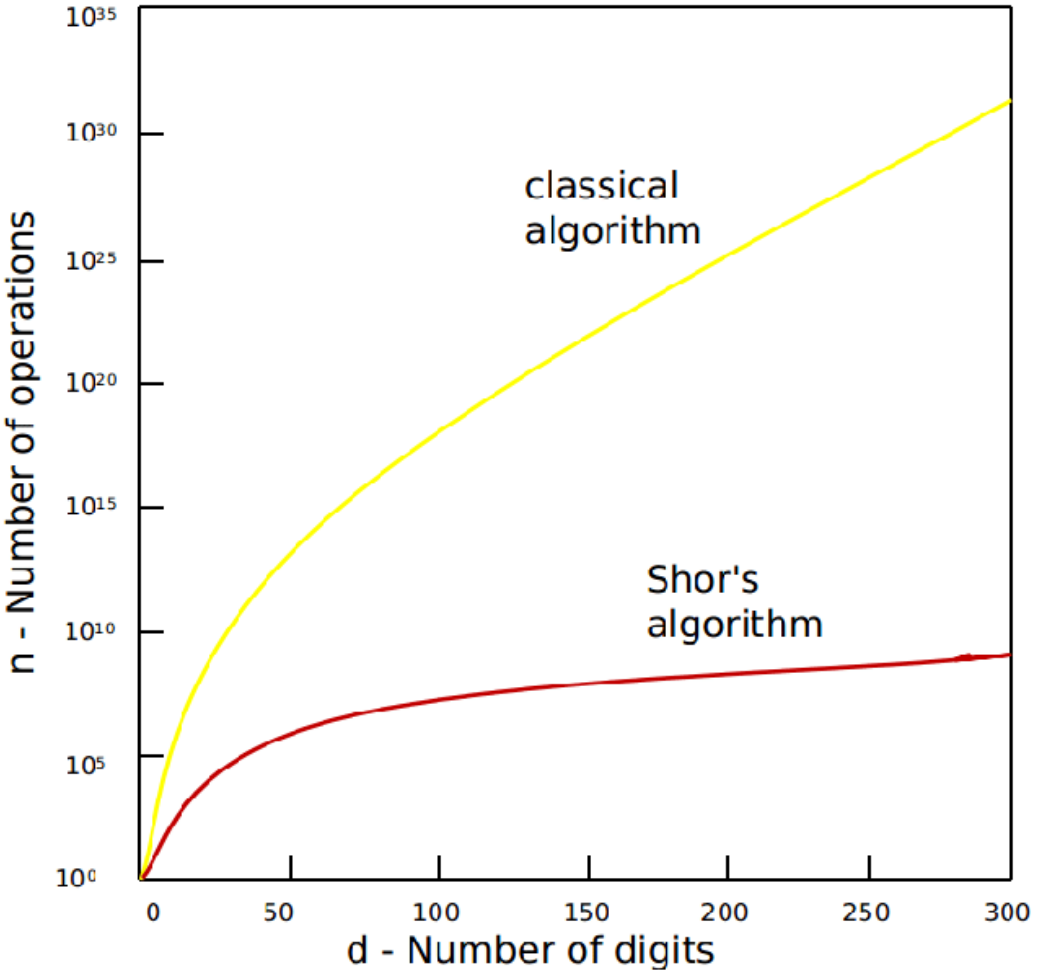
- **Goal:** Find a **non-trivial factor** of a composite integer N efficiently using quantum computation.
- **Importance:** Exponential speed-up for factoring → breaks RSA.
- **Quantum Circuit Construction:**
  - Choose **random a < N** such that **gcd(a, N) = 1**
  - Define **periodic function** $f(x) = a^x \bmod N$
  - Use a quantum circuit to find the ***period r*** of f(x):
    - Apply **Hadamard** gates
    - Apply **controlled modular** exponentiation
    - Apply **reverse QFT**
  - Measure and **compute r** via **classical post-processing**
  - Use ***r*** to **find factors**: $p = \gcd(a^{r/2} - 1, N)$, $q = \gcd(a^{r/2} + 1, N)$



Shor’s Factoring Algorithm

Shor’s Factoring Algorithm





Lab Demo & Practice

- Build **Shor's Factoring** with:
  - Step 1 — Choose a **Random Base a**
  - Step 2 — Find the **Period r Quantumly**
    - Create a quantum circuits (with QFT)
  - Step 3 — Measure and Estimate ***Period r***
  - Step 4 — Compute Factors of **user input *N***

```
Factoring N = 15

Attempt 1: using a = 2
Estimated period (r) = 4

Factors found: (3, 5)
```

```
Factoring N = 55

Attempt 1: using a = 36
Estimated period (r) = 5
r is odd, retrying...

Attempt 2: using a = 39
Estimated period (r) = 11
r is odd, retrying...
Found trivial factor: 5

Factors found: (5, 11)
```

Comparison & Discussion

Table of Comparison

Algorithm	Problem Type / Purpose	Classical Complexity	Quantum Complexity / Speedup	Output / Insight
QFT	Quantum amplitudes	N/A	$O((\log N)^2)$	Basis change in quantum state amplitudes
BV	Find hidden bit string	n queries	1 query (linear → constant)	Reveals hidden string
QPE	Estimate eigenphase $\|\psi\rangle$ of U	$\|\psi\rangle = e^{2\pi i\phi}$	$\|\psi\rangle$	Controlled-U + inverse QFT
Grover	Search unsorted database for target element	O(N)	$O(\sqrt{N})$ (quadratic speedup)	Probabilistic target found
Shor	Factor large integer N via period finding	Exponential in log N	Polynomial in logN (exponential speedup)	Prime factors of N

Comparison & Discussion

Lab Summary

- **Lab 1:** QFT-3
- **Lab 2:** BV-8 (by insert secret number)
- **Lab 3:** Grover’s Search on 3 qubits
- **Lab 4:** Shor’s Factoring
- Bonus Point: test each on **FakePerth** vs. **Other simulator** to compare fidelity


Thank you!