ĐẠI HỌC QUỐC GIA TPHCM

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

Bài tập 4

Đề tài: Tính toán lượng tử

Môn học: Nhập môn Tính toán lượng tử

Sinh viên thực hiện:

Lưu Thượng Hồng (23122006)

Giáo viên hướng dẫn:

ThS. Vũ Quốc Hoàng

Ngày 17 tháng 1 năm 2026

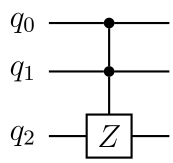
Mục lục

1 Câu 1

Đề bài: Thiết kế mạch 3 qubit để cài đặt R_u, phép toán phản xạ quanh trục $|u\rangle$, với:

(a) $|u\rangle = |111\rangle$

Trường hợp này tương đương với cổng Controlled-Controlled-Z (CCZ).



Kiểm tra lại với code:

```
qc = QuantumCircuit(3)

# 1. Chuẩn bị |111>
qc.x([0, 1, 2])

# 2. Mạch phản xạ (CCZ)
qc.ccz(0, 1, 2)

# 3. Check
print("State (a):", Statevector(qc))
```

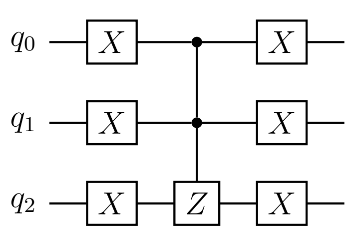
Kết quả thu được:

```
Statevector([ 0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,
             -1.+0.j],
            dims=(2, 2, 2))
```

Đúng với mong đợi.

(b) $|u\rangle = |000\rangle$

Ta dùng cổng X để chuyển $|000\rangle \to |111\rangle$, thực hiện phản xạ, rồi trả về lại.



Kiểm tra lại với code:

```
qc = QuantumCircuit(3)

# 1. Chuẩn bị |000>

# 2. Mạch phản xạ: X -> CCZ -> X
qc.x([0, 1, 2])
qc.ccz(0, 1, 2)
qc.x([0, 1, 2])

# 3. Check
print("State (b):", Statevector(qc))
```

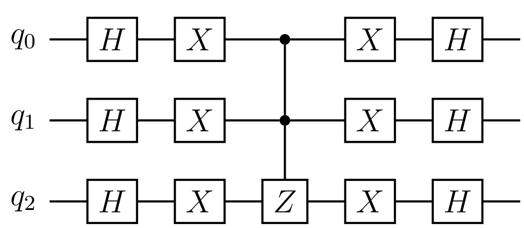
Kết quả thu được:

```
Statevector([-1.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,  0.+0.j,
              0.+0.j],
            dims=(2, 2, 2))
```

Đúng với mong đợi.

(c) $|u\rangle = |+\rangle^{\otimes 3}$

Vì $|+\rangle \xrightarrow{H} |0\rangle \xrightarrow{X} |1\rangle$, ta cần kẹp giữa bởi chuỗi H và X.



Kiểm tra lại với code:

```
qc = QuantumCircuit(3)

# 1. Chuẩn bị |+++>
qc.h([0, 1, 2])

# 2. Mạch phản xạ: H -> X -> CCZ -> X -> H
qc.h([0, 1, 2])
```

```python
qc.x([0, 1, 2])
qc.ccz(0, 1, 2)
qc.x([0, 1, 2])
qc.h([0, 1, 2])

# 3. Check
print("State (c):", Statevector(qc))
```

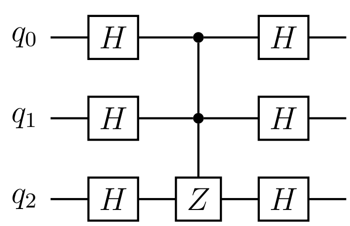
Kết quả thu được:

```
Statevector([-0.35355339+0.j, -0.35355339+0.j, -0.35355339+0.j,
             -0.35355339+0.j, -0.35355339+0.j, -0.35355339+0.j,
             -0.35355339+0.j, -0.35355339+0.j],
            dims=(2, 2, 2))
```

Đúng với mong đợi.

(d) $|u\rangle = |-\rangle^{\otimes 3}$

Vì $|-\rangle \xrightarrow{H} |1\rangle$, ta chỉ cần kẹp giữa bởi cổng H.



Kiểm tra lại với code:

```python
qc = QuantumCircuit(3)

# 1. Chuẩn bị |---> (X rồi H)
qc.x([0, 1, 2])
qc.h([0, 1, 2])

# 2. Mạch phản xạ: H -> CCZ -> H
qc.h([0, 1, 2])
qc.ccz(0, 1, 2)
qc.h([0, 1, 2])

# 3. Check
```

```
print("State (d):", Statevector(qc))
```

Kết quả thu được:

```
Statevector([-0.35355339+0.j,  0.35355339+0.j,  0.35355339+0.j,
             -0.35355339+0.j,  0.35355339+0.j, -0.35355339+0.j,
             -0.35355339+0.j,  0.35355339+0.j],
            dims=(2, 2, 2))
```

Đúng với mong đợi.

2 Câu 2

Đề bài: Cần sửa đổi giao thức E91 như thế nào nếu các cặp qubit ở trạng thái vướng víu là $|\Psi^+\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$ thay vì $|\Phi^+\rangle$.

Bài làm: Trạng thái $|\Psi^+\rangle$ có tính chất tương quan khác biệt so với $|\Phi^+\rangle$ khi đo đạc:

- **Trong cơ sở Z ($\{|0\rangle, |1\rangle\}$):** Kết quả đo luôn **ngược nhau** (Alice ra 0 thì Bob ra 1 và ngược lại) do cấu trúc $|01\rangle + |10\rangle$.
- **Trong cơ sở X ($\{|+\rangle, |-\rangle\}$):** Ta có $|\Psi^+\rangle = \frac{1}{\sqrt{2}}(|{+}{+}\rangle - |{-}{-}\rangle)$, nên kết quả đo luôn **giống nhau**.

Cách sửa đổi: Trong giai đoạn sàng lọc khóa (sifting phase), khi cả hai người cùng đo trên cơ sở Z, một trong hai bên (ví dụ Bob) cần thực hiện phép **đảo bit** (bit-flip: $0 \to 1, 1 \to 0$) đối với các kết quả đo của mình. Với các cơ sở khác mà kết quả tương quan dương (như cơ sở X), ta giữ nguyên kết quả.

3 Câu 3

Đề bài: Ta thấy rằng R_0, thao tác phản xạ qua $|0^n\rangle$, có thể cài đặt như oracle pha U_f với $f : \{0,1\}^n \to \{0,1\}$ được định nghĩa cho $x \in \{0,1\}^n$ là

$$f(x) = \begin{cases} 0 & x = 0^n \\ 1 & x \neq 0^n \end{cases}.$$

Quan sát kỹ hơn, ta thấy f chính là phép OR bit

$$\mathrm{OR}(x_{n-1}...x_1x_0) = \sum_{i=0}^{n-1} x_i \pmod 2.$$

Như vậy bằng cách dùng các phiên bản lượng tử cho các cổng logic trong mạch logic cài đặt phép OR bit ta có thể cài đặt R_0. Cài đặt cụ thể R_0 theo cách này trong trường hợp $n = 3$. Kiểm tra kết quả trên thư viện Qiskit.

Bài làm:

Hàm $f : \{0,1\}^3 \to \{0,1\}$ được định nghĩa:

$$f(x) = \begin{cases} 0 & x = 000 \\ 1 & x \neq 000 \end{cases}$$

Đây là hàm OR của 3 bit. Để cài đặt Oracle pha U_f sao cho $U_f |x\rangle = (-1)^{f(x)} |x\rangle$, ta sử dụng kỹ thuật *Phase Kickback* với một qubit hỗ trợ (ancilla) ở trạng thái $|-\rangle$.

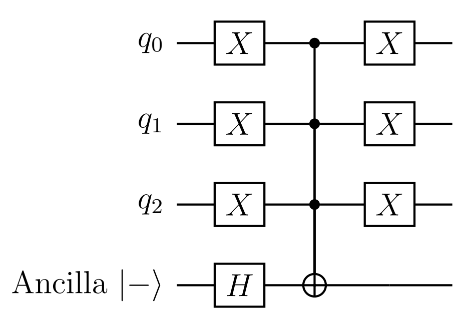
Theo định luật De Morgan:

$$x_0 \vee x_1 \vee x_2 \iff \neg(\neg x_0 \wedge \neg x_1 \wedge \neg x_2)$$

Do đó, mạch OR có thể được xây dựng bằng cách đảo ngược đầu vào (cổng X), thực hiện phép AND (cổng Multi-controlled Toffoli), và đảo ngược lại.

Thiết kế mạch

Mạch dưới đây thực hiện phép toán $-R_0$ (tương đương về mặt vật lý với R_0). Nó sẽ đảo dấu pha của trạng thái $|000\rangle$ và giữ nguyên các trạng thái khác.



Giải thích:

- Nếu đầu vào là $|000\rangle$, các cổng X chuyển nó thành $|111\rangle$. Cổng Toffoli (MCX) được kích hoạt, tác động lên qubit Ancilla ($|-\rangle$), gây ra hiệu ứng *phase kickback*, làm toàn bộ hệ thống nhân với -1. Sau đó các cổng X trả lại trạng thái $|000\rangle$. Kết quả: $-|000\rangle$.
- Nếu đầu vào khác $|000\rangle$ (ví dụ $|001\rangle$), sau cổng X sẽ là $|110\rangle$. Cổng Toffoli không kích hoạt. Pha giữ nguyên.

Cài đặt trong Qiskit:

Cài đặt mạch R_0:

```python
def build_r0_circuit():
    # 3 qubit dữ liệu + 1 qubit ancilla (index 3)
    qc = QuantumCircuit(4)

    # 1. Chuẩn bị Ancilla về trạng thái |->
    qc.x(3)
    qc.h(3)

    # 2. Cài đặt mạch Logic (De Morgan OR)
    # Bước A: Đảo bit (X) để phát hiện trạng thái 000
    qc.x([0, 1, 2])

    # Bước B: Multi-controlled X (Kickback phase)
    # Kích hoạt khi q0=q1=q2=1 (tức là input gốc là 000)
    qc.mcx([0, 1, 2], 3)

    # Bước C: Đảo bit lại
    qc.x([0, 1, 2])

    # 3. Trả Ancilla về |0>
    qc.h(3)
    qc.x(3)

    return qc
```

Kiểm tra kết quả:

```python
# Lấy toán tử của mạch
circuit = build_r0_circuit()
```

```
state = Statevector(circuit)

print("Kiểm tra pha của các trạng thái:")

# 1. Kiểm tra trạng thái |000> (Index 0)
amp_000 = state.data[0]
print(f"Input |000>: {amp_000.real:.2f} (Mong đợi: -1.0)")

# 2. Kiểm tra trạng thái |001> (Index 1)
qc_test = QuantumCircuit(4)
qc_test.x(0) # Tạo input 001
qc_test.append(circuit, [0,1,2,3])
amp_001 = Statevector(qc_test).data[1] # Lấy biên độ tại index 1
print(f"Input |001>: {amp_001.real:.2f}  (Mong đợi:  1.0)")
```

Kết quả thu được:

```
Kiểm tra pha của các trạng thái:
Input |000>: -1.00
Input |001>: 1.00
```

Đúng với mong đợi.

4 Câu 4

Đề bài: Cài đặt hoàn chỉnh thuật toán Grover và thực hiện các tính toán cho trường hợp $n = 4$ với lời giải là 0^n.

Bài làm:

1. Phân tích tham số

- Số qubit: $n = 4$. Không gian tìm kiếm $N = 2^4 = 16$.
- Trạng thái cần tìm: $|w\rangle = |0000\rangle$.
- Số lần lặp tối ưu (Grover iterations): $k \approx \frac{\pi}{4}\sqrt{N} \approx 3.14 \Rightarrow$ chọn $\mathbf{k = 3}$.

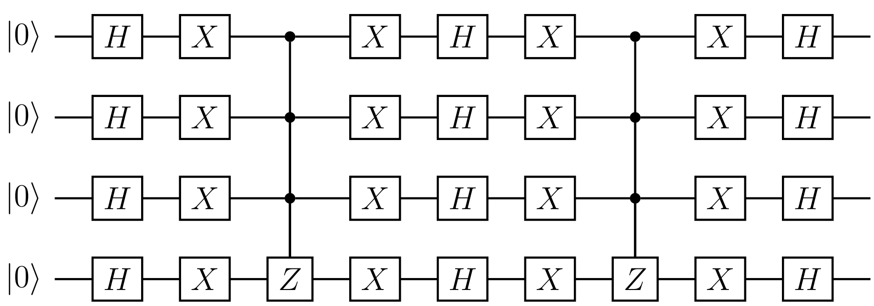
2. Thiết kế mạch Mạch bao gồm khởi tạo trạng thái chồng chập $|s\rangle = H^{\otimes 4} |0\rangle^{\otimes 4}$, theo sau là 3 lần lặp của toán tử Grover $G = U_s U_w$.

a) Oracle (U_w): Đảo dấu pha của trạng thái $|0000\rangle$. Ta dùng cổng Multi-controlled Z (MCZ). Vì MCZ chuẩn tác động lên $|1111\rangle$, ta cần bọc nó bởi các cổng X (NOT).

$$U_w = X^{\otimes 4} \cdot \mathrm{MCZ} \cdot X^{\otimes 4}$$

b) Diffuser (U_s): Phản xạ qua trạng thái trung bình $|s\rangle$. Công thức là $H^{\otimes 4}(2|0\rangle\langle 0| - I)H^{\otimes 4}$. Phần lõi $(2|0\rangle\langle 0| - I)$ chính là mạch phản xạ quanh $|0000\rangle$ tương tự như Oracle.

Sơ đồ mạch (Minh họa 1 vòng lặp):



Cài đặt trong Qiskit như sau:

```
def grover():
    n = 4
    qc = QuantumCircuit(n)

    # 1. Khởi tạo |s>
    qc.h(range(n))

    # Số lần lặp tối ưu cho N=16 là 3
    for _ in range(3):
        # --- ORACLE: Mark |0000> ---
        # (X -> MCZ -> X)
        qc.x(range(n))
        qc.mcp(np.pi, list(range(n-1)), n-1)
        qc.x(range(n))

        # --- DIFFUSER: Reflect |s> ---
        # (H -> X -> MCZ -> X -> H)
        qc.h(range(n))
        qc.x(range(n))
```

```
        qc.mcp(np.pi, list(range(n-1)), n-1)
        qc.x(range(n))
        qc.h(range(n))

    return qc
```

Chạy thuật toán và kiểm tra kết quả:

```
# 1. Tạo mạch
qc = grover()

# 2. Tính toán Statevector
final_state = Statevector(qc)

# 3. Lấy mẫu kết quả
counts = final_state.sample_counts(shots=1000)

# 4. In kết quả
print("Kết quả")
sorted_counts = sorted(counts.items(), key=lambda x: x[1], reverse=True)

for state, count in sorted_counts:
    prob = (count / 1000) * 100
    print(f"Trạng thái |{state}>: {count} lần ({prob:.1f}%)")

# Lấy biên độ chính xác của |0000>
amp_0000 = final_state.data[0]
prob_theoretical = np.abs(amp_0000)**2 * 100
print("-" * 30)
print(f"Xác suất lý thuyết của |0000>: {prob_theoretical:.2f}%")
```

Kết quả thuật toán được trình bày trong Bảng 1.

Bảng 1: Kết quả thực nghiệm thuật toán Grover ($n = 4$, 1000 shots)

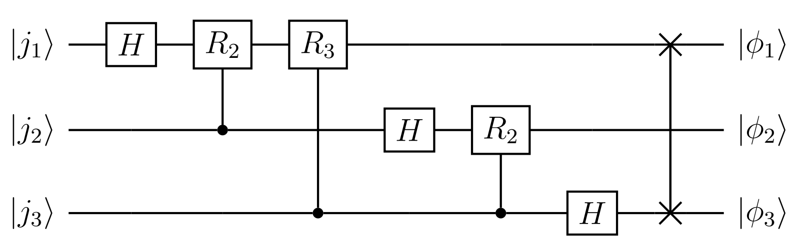
Trạng thái	**Số lần đo (Counts)**	**Tần suất (%)**
$\|0000\rangle$	**959**	**95.9%**
$\|0111\rangle$	6	0.6%
$\|1011\rangle$	5	0.5%
$\|0010\rangle$	4	0.4%
$\|1001\rangle$	4	0.4%
$\|1010\rangle$	4	0.4%
$\|0001\rangle$	3	0.3%
$\|1111\rangle$	3	0.3%
$\|0011\rangle$	2	0.2%
$\|0100\rangle$	2	0.2%
$\|0110\rangle$	2	0.2%
$\|1000\rangle$	2	0.2%
$\|1100\rangle$	2	0.2%
$\|0101\rangle$	1	0.1%
$\|1110\rangle$	1	0.1%
So sánh lý thuyết		
Xác suất lý thuyết của $\|0000\rangle$		96.13%

5 Câu 5

Đề bài: Thiết kế mạch QFT 3 qubit. Tính toán từng bước để thấy mạch biến các vector cơ sở tính toán $|j\rangle$ thành các vector $|\phi_j\rangle$ tương ứng.

Bài làm:

1. Thiết kế mạch Mạch QFT cho 3 qubit (q_1, q_2, q_3 tương ứng với các bit của đầu vào $|j\rangle = |j_1 j_2 j_3\rangle$) sử dụng các cổng Hadamard (H) và Controlled-Phase R_k (với pha $2\pi/2^k$). Sơ đồ mạch như sau:



2. Tính toán từng bước Xét trạng thái đầu vào $|j\rangle = |j_1 j_2 j_3\rangle$. Ta biểu diễn dưới dạng thập phân $j = j_1 2^2 + j_2 2^1 + j_3 2^0 = 4j_1 + 2j_2 + j_3$.

Bước 1: Biến đổi trên qubit 1 ($|j_1\rangle$)

- Áp dụng H: $\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i(0.j_1)}|1\rangle)$.

- Áp dụng R_2 (điều khiển bởi j_2): Thêm pha $2\pi i \frac{j_2}{2^2} = 2\pi i(0.0j_2)$.
- Áp dụng R_3 (điều khiển bởi j_3): Thêm pha $2\pi i \frac{j_3}{2^3} = 2\pi i(0.00j_3)$.

Trạng thái qubit 1 lúc này: $\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i(0.j_1j_2j_3)}|1\rangle)$.

Bước 2: Biến đổi trên qubit 2 ($|j_2\rangle$)

- Áp dụng H: $\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i(0.j_2)}|1\rangle)$.
- Áp dụng R_2 (điều khiển bởi j_3): Thêm pha $2\pi i(0.0j_3)$.

Trạng thái qubit 2 lúc này: $\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i(0.j_2j_3)}|1\rangle)$.

Bước 3: Biến đổi trên qubit 3 ($|j_3\rangle$)

- Chỉ áp dụng H: $\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i(0.j_3)}|1\rangle)$.

Bước 4: Cổng SWAP Mạch QFT yêu cầu đảo ngược thứ tự qubit để phù hợp với định nghĩa toán học. Ta đổi chỗ qubit 1 và qubit 3. Kết quả cuối cùng là tích tensor:

$$|\psi\rangle = \frac{1}{\sqrt{8}}\left(|0\rangle + e^{2\pi i 0.j_3}|1\rangle\right) \otimes \left(|0\rangle + e^{2\pi i 0.j_2j_3}|1\rangle\right) \otimes \left(|0\rangle + e^{2\pi i 0.j_1j_2j_3}|1\rangle\right)$$

Đây chính xác là biểu diễn của QFT dưới dạng tích tensor (product state).

Cài đặt bằng Qiskit:

```
def qft_3_qubit():
    qc = QuantumCircuit(3)

    # --- Thiết kế thủ công (Manual Design) ---
    # 1. H trên q2 (tương ứng j1 trong bài làm)
    qc.h(2)
    # Controlled-Phase từ q1 lên q2 (góc pi/2)
    qc.cp(np.pi/2, 1, 2)
    # Controlled-Phase từ q0 lên q2 (góc pi/4)
    qc.cp(np.pi/4, 0, 2)

    # 2. H trên q1 (tương ứng j2)
    qc.h(1)
    # Controlled-Phase từ q0 lên q1 (góc pi/2)
```

```
    qc.cp(np.pi/2, 0, 1)

    # 3. H trên q0 (tương ứng j3)
    qc.h(0)

    # 4. SWAP đầu cuối
    qc.swap(0, 2)

    return qc
```

Chạy và kiểm tra:

```
# 1. Tạo mạch thủ công
manual_qc = qft_3_qubit()

# 2. Tạo mạch chuẩn thư viện để so sánh
library_qc = QFT(num_qubits=3, do_swaps=True).decompose()

# 3. Tính toán statevector (Ma trận toán tử)
# Ta kiểm tra xem 2 mạch có tạo ra cùng một toán tử Unitary không
sv_manual = Statevector(manual_qc)
sv_lib = Statevector(library_qc) # Input mặc định là |000>

# Tính độ tương đồng (Fidelity)
fidelity = sv_manual.inner(sv_lib)
print(f"Độ trùng khớp (Fidelity): {abs(fidelity):.5f}")

# In thử kết quả với input |000>
print("Statevector output:")
print(np.round(sv_manual.data, 3))
```

Kết quả thu được:

```
Độ trùng khớp (Fidelity): 1.00000
Statevector output:
[0.354+0.j 0.354+0.j 0.354+0.j 0.354+0.j 0.354+0.j 0.354+0.j 0.354+0.j
 0.354+0.j]
```

Như vậy, mạch QFT thủ công đã được thiết kế đúng và cho kết quả trùng khớp hoàn toàn với mạch QFT chuẩn từ thư viện Qiskit.