C S 358H: Intro to Quantum Information Science

Sayam Sethi

November 2024

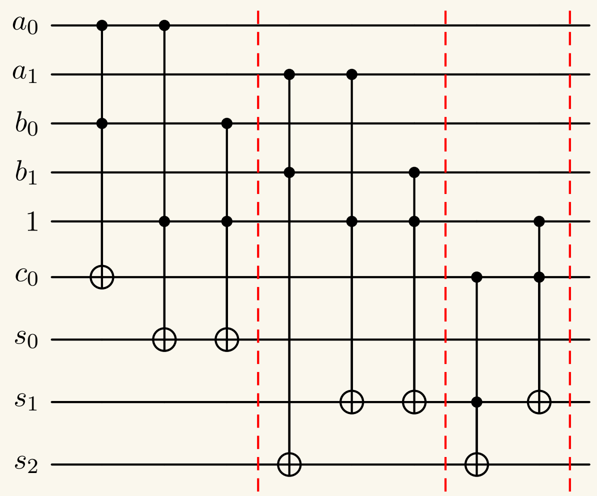
Contents

1 Toffoli-based Addition

Question 1.1

Question. *Work out an explicit circuit of Toffoli gates for adding two 2-bit integers to get a 3-bit integer — assume the integers are unsigned, encoded in binary in the usual, simplest way. You can use arbitrary ancilla bits initialized to 0 or 1. Be sure to designate your input, output, and garbage registers.*
Show the garbage bits that are generated by your circuit when 11 is added to 10.

Proof. The circuit that performs a full-adder for a 2-bit unsigned addition is:



We represent the two 2-bit inputs as a_1a_0 and b_1b_0 and the output as $s_2s_1s_0$. The circuit uses 8 Toffoli gates.
To see why it works, note that the group of three Toffolis before the first slice, and the group of three Toffolis between the first and second slice perform the same function but on different input and output bits. The first Toffoli in the sequence computes the carry bit (c_0, s_2) and the next two Toffolis compute the xor of the input bits to get the sum (s_0, s_2). The two Toffolis before the last slice are used to incorporate the carry bit from the first addition into the sum bits s_1, s_2. We use two garbage bits $1, c_0$, for executing CNOTs and storing the carry, respectively.
When 11 is added to 10, we have $a_1a_0 = 11$ and $b_1b_0 = 10$. The output of the circuit is $s_2s_1s_0 = 101$ and the garbage bits are $c_0 = 0$ and $1 = 1$. □

2 The Quantum Fourier Transform

The Quantum Fourier Transform QFT_d is a quantum gate acting on *qudits*, i.e. quantum systems with d levels. It is defined below for $x, y \in \{0, 1, ..., d-1\}$.

$$QFT_d |x\rangle = \frac{1}{\sqrt{d}} \sum_{y=0}^{d-1} \omega^{xy} |y\rangle$$

where $\omega = e^{2\pi i/d}$ is a primitive dth root of unity.

To be clear, a qudit is just like a qubit but it's a vector of d amplitudes instead of just 2 amplitudes. Then, a unitary acting on a qudit has dimension $d \times d$.

Question 2.1

Question. *Calculate QFT_2, QFT_3, and QFT_4 explicitly (either by writing down the corresponding matrix or equivalently by specifying the action on each of the standard basis states). By what other name is QFT_2 known?*

Solution. We have,

$$\begin{aligned} QFT_2 &= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \text{ also known as the Hadamard gate} \\ QFT_3 &= \frac{1}{\sqrt{3}} \begin{pmatrix} 1 & 1 & 1 \\ 1 & \omega & \omega^2 \\ 1 & \omega^2 & \omega \end{pmatrix}, \text{ where } \omega = e^{2\pi/3} \\ QFT_4 &= \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix} \end{aligned} \tag{1}$$

□

Question 2.2

Question. *Prove that QFT_d is unitary for all d.*

Proof. We will show that QFT_d is a unitary by showing that $\langle QFT_d |x_1\rangle | QFT_d |x_2\rangle\rangle = 0$ for any orthonormal basis vector $x_1 \neq x_2$ and it is equal to 1 when $x_1 = x_2$ (this is equivalent to showing that the columns of QFT_d form an orthonormal basis):

$$\begin{aligned}\langle QFT_d\,|x_1\rangle\,|QFT_d\,|x_2\rangle\rangle &= \langle \frac{1}{\sqrt{d}}\sum_{y=0}^{d-1}\omega^{x_1 y}\,|y\rangle\,|\frac{1}{\sqrt{d}}\sum_{z=0}^{d-1}\omega^{x_2 z}\,|z\rangle\rangle \\ &= \frac{1}{\sqrt{d}}\sum_{y=0}^{d-1}\sum_{z=0}^{d-1}\omega^{-x_1 y}\omega^{x_2 z}\,\langle y|z\rangle \\ &= \frac{1}{\sqrt{d}}\sum_{y=0}^{d-1}(\omega^{-x_1})^y(\omega^{x_2})^y = \frac{1}{\sqrt{d}}\sum_{y=0}^{d-1}(\omega^{x_2-x_1})^y \\ &= \begin{cases}1, & \text{if } x_1 = x_2\\ 0, & \text{otherwise}\end{cases}\end{aligned} \tag{2}$$

Hence, proved that QFT_d is unitary for all d. □

Question 2.3

Question. *For which values of d is QFT_d its own inverse?*

Solution. QFT_d is its own inverse when $QFT_d = QFT_d^\dagger$. We have,

$$\begin{aligned}&(QFT_d)_{ij} = (QFT_d^\dagger)_{ij} \\ \implies &\frac{1}{\sqrt{d}}\omega^{ij} = \frac{1}{\sqrt{d}}\omega^{-ij} \\ \implies &\omega^{2ij} = 1 \\ \implies &e^{4\pi ij/d} = 1 \text{ for all } i,j \in \{0,1,...,d-1\} \\ \implies &d = 1,2\end{aligned} \tag{3}$$

Therefore, QFT_d is its own inverse for $d = 1, 2$. □

Question 2.4

Question. *In Recitation 4, we saw that the qudit clock and shift matrices are respectively defined as $X_d\,|x\rangle = |x + 1 \bmod d\rangle$ and $Z_d\,|x\rangle = \omega^x\,|x\rangle$ for $x \in 0, 1, \ldots, d-1$. Show that*

$$QFT_d^\dagger X_d QFT_d = Z_d^\dagger.$$

Proof. We will show this relation holds by showing it holds for every basis state $x \in$

$\{0, 1, \ldots, d-1\}$:

$$\begin{aligned} QFT_d^\dagger X_d QFT_d \left|x\right\rangle &= QFT_d^\dagger X_d \left(\frac{1}{\sqrt{d}} \sum_{y=0}^{d-1} \omega^{xy} \left|y\right\rangle \right) \\ &= QFT_d^\dagger \left(\frac{1}{\sqrt{d}} \sum_{y=0}^{d-1} \omega^{xy} \left|y+1 \bmod d\right\rangle \right) \\ &= QFT_d^\dagger \left(\frac{1}{\sqrt{d}} \sum_{y=0}^{d-1} \omega^{x(y+1)} \omega^{-x} \left|y+1 \bmod d\right\rangle \right) \\ &= \omega^{-x} QFT_d^\dagger \left(\frac{1}{\sqrt{d}} \sum_{y=0}^{d-1} \omega^{x(y+1 \bmod d)} \left|y+1 \bmod d\right\rangle \right) \\ &= \omega^{-x} QFT_d^\dagger \left(\frac{1}{\sqrt{d}} \sum_{z=0}^{d-1} \omega^{xz} \left|z\right\rangle \right) \\ &= \omega^{-x} QFT_d^\dagger \left(QFT_d \left|x\right\rangle \right) \\ &= \omega^{-x} \left|x\right\rangle \\ &= Z_d^\dagger \left|x\right\rangle \end{aligned} \tag{4}$$

Since Equation 4 is true for all basis states, we have $QFT_d^\dagger X_d QFT_d = Z_d^\dagger$. □

3 RSA

Two spies of an enemy nation are known to use the RSA crypto-system with public keys

$$N = 4668619 \quad \text{and} \quad e = 3.$$

And intelligence has revealed that

$$\varphi(N) = 4664296.$$

Now we've intercepted a message sent between the two spies showing that they intend to meet at a certain time:

$$m = 1202997$$

Let's arrange for them (our enemies) to miss each other by an hour.

[Note that decrypted messages are encoded in ASCII with two ***decimal*** *digits per character, as described on* www.asciitable.com*. You're of course free to use computer help.* www.dcode.fr/modular-exponentiation *and* www.dcode.fr/ascii-code *may prove useful. But please don't forget to show intermediate steps in the solutions.]*

Question 3.1

Question. *Confirm that our intelligence is correct by using N and $\varphi(N)$ to determine the two factors p and q of N and checking that indeed $pq = N$. (Note that part (a) is not required in the parts that follow.)*

Proof. We will use the solution from Question 4.1 to solve for p, q. On solving, we get:

$$\begin{aligned} p &= 2237 \\ q &= 2087 \\ pq &= 2237 \times 2087 = 4668619 = N \end{aligned} \tag{5}$$

□

Question 3.2

Question. *Meanwhile an analyst has gone ahead and done the work of determining*

$$d = 3109531$$

to be the inverse of e in the multiplicative group $\mathbb{Z}^{\times}_{\varphi(N)}$. Verify that this is correct.

Proof. If d is computed correctly, we must have $e \cdot d = 1 \bmod \varphi(N)$. On solving this, we get $e \cdot d = 9328593$, which on dividing with $\varphi(N)$ gives 1 as a remainder. □

Question 3.3

Question. *Decrypt the given message.*

Solution. To decrypt the message, we need to compute $m_0 = m^d \bmod N$. On solving we get $m_0 = 548077$, which gives us the message as "6PM". □

Question 3.4

Question. *Encrypt a new message in the same format instructing to meet an hour later; we will send this to the second spy instead of the original message.*

Solution. We wish to encrypt the message "7PM", this can be written in the decimal encoding as $m_1 = 558077$. The encrypted message will be $m_1^e \bmod N = 3706174$. □

Code used for reference:

```
N = 4668619
e = 3
phi = 4664296
m = 1202997
d = 3109531


def solve_pq():
    temp = N - phi + 1
    sq = int(pow(pow(temp, 2) - 4 * N, 0.5))
    return (temp + sq) // 2, (temp - sq) // 2


def bin_exp(x, n, m):
    res = 1
    while n > 0:
        if n % 2 == 1:
            res = (res * x) % m
        x = (x * x) % m
        n = n // 2
    return res


p, q = solve_pq()
print(p, q)
print(p * q == N)

print(e * d)
print(e * d % phi == 1)

m0 = bin_exp(m, d, N)
print(m0)
print(bin_exp(m0, e, N) == m)

m1 = m0 + 10000
print(bin_exp(m1, e, N))
print(bin_exp(bin_exp(m1, e, N), d, N) == m1)
```

4 Factoring

Question 4.1

Question. *Let $N = pq$ be a product of two large primes. Show that given the order $(p-1)(q-1)$ of the multiplicative group $\pmod N$, one can efficiently recover the prime factors of N.*

Proof. Since we know N and $\varphi(N)$, we can obtain a quadratic equation in p or q:

$$\begin{aligned}
&(p-1)(q-1) = pq - p - q + 1 \\
\implies &\varphi(N) = N - p - q + 1 \\
\implies &q = (N + 1 - \varphi(N)) - p
\end{aligned}$$

We also know that,

$$\begin{aligned}
&N = pq \\
\implies &N = p((N+1-\varphi(N)) - p) \\
\implies &p^2 - (N+1-\varphi(N))p + N = 0 \\
\implies &p = \frac{(N+1-\varphi(N)) \pm \sqrt{(N+1-\varphi(N))^2 - 4N}}{2}
\end{aligned} \tag{6}$$

The two roots of this quadratic equation are p, q (since we cannot have more than one pair of prime factors as solutions to the equation). □