Quantum Computation **(CMU 18-859BB, Fall 2015)**

Lecture 7: Quantum Fourier Transform over Z_N

September 30, 2015

Lecturer: Ryan O'Donnell *Scribe: Chris Jones*

1 Overview

Last time, we talked about two main topics:

- The quantum Fourier transform over $\mathbb{Z}_2^n$
- Solving Simon's problem with the transform over $\mathbb{Z}_2^n$

In this lecture and the next, the theory will be developed again, but over a different group, $\mathbb{Z}_N$. We'll talk about:

- The quantum Fourier transform over $\mathbb{Z}_N$ (today)
- Solving the period-finding problem with the transform over $\mathbb{Z}_N$ (next time)

As a side note, for this lecture and in general, one should not stress too much about maintaining normalized quantum states, that is, states with squared amplitudes that sum to 1. Carrying around (and computing) the normalizing factor can be a big pain, and it's understood that the "true" quantum state has normalized amplitudes. To this end, we agree that the quantum state

$$\sum_{x \in \{0,1\}^n} \alpha_x |x\rangle$$

is in all respects equivalent to the normalized quantum state

$$\left(\sum_{x \in \{0,1\}^n} |\alpha_x|^2 \right)^{-1} \left(\sum_{x \in \{0,1\}^n} \alpha_x |x\rangle \right)$$

This tradition is due to physicists who tend to omit normalizing constants for convenience.

2 Defining the Fourier transform over $\mathbb{Z}_N$

2.1 Review of the transform over $\mathbb{Z}_2^n$

For now, we will forget all we know of quantum computing, except for what we learned last lecture. Let's think about the Fourier transform over $\mathbb{Z}_2^n$, and see if we can adopt the idea to $\mathbb{Z}_N$.

Consider an arbitrary function $g : \{0,1\}^n \to \mathbb{C}$. In the previous lecture, we looked at the vector space of functions from $\{0,1\}^n$ to $\mathbb{C}$, and thought of g as an element of that vector space. This vector space is 2^n-dimensional, with a "standard" basis being the indicator functions $\{\delta_y\}_{y \in \{0,1\}^n}$. With respect to this basis, we have the representation

$$g = \begin{bmatrix} g(0^n) \\ g(0^{n-1}1) \\ \vdots \\ g(1^n) \end{bmatrix}$$

We'll more often see vectors of the form

$$h = \frac{1}{\sqrt{N}} \begin{bmatrix} h(0^n) \\ \vdots \\ h(1^n) \end{bmatrix}$$

i.e. with respect to the basis $\{\sqrt{N}\delta_y\}_{y \in \{0,1\}^n}$. This is nicer because the standard dot product of such vectors is an expectation:

$$\langle g, h \rangle = \frac{1}{N} \sum_{x \in \{0,1\}^n} g(x)^* h(x) = \mathop{\mathbf{E}}_{x \sim \{0,1\}^n} [g(x)^* h(x)]$$

In particular, if $g : \{0,1\}^n \to \{-1,+1\}$, g is a unit vector.

Remember: no stress about the constant $\frac{1}{\sqrt{N}}$. It's just there to make more things unit vectors more often.

In this vector space, we found a particular orthonormal basis that made representations of f particularly nice to work with: the basis of *parity functions* $\{\chi_\gamma\}_{\gamma \in \mathbb{Z}_2^n}$, defined by

$$\chi_\gamma(x) = (-1)^{\gamma \cdot x}$$

where $\gamma \cdot x$ is the dot product in $\mathbb{Z}_2^n$. In the vector representation,

$$|\chi_\gamma\rangle = \frac{1}{\sqrt{N}} \begin{bmatrix} +1 \\ (-1)^{\gamma \cdot 0^{n-1}1} \\ \vdots \\ (-1)^{\gamma \cdot 1^n} \end{bmatrix}$$

We used the notation $\widehat{g}(\gamma)$ to denote the coefficient of χ_γ in the representation of g. All this notation gave us the final form

$$g(x) = \sum_{\gamma \in \mathbb{Z}_2^n} \widehat{g}(\gamma) \, |\chi_\gamma\rangle$$

If we endowed the domain with the group structure of $\mathbb{Z}_2^n$, or equivalently, the vector space $\mathbb{F}_2^n$, the χ_γ are nice because they are *characters* on $\mathbb{Z}_2^n$.

Definition 2.1. Let $\mathbb{C}^*$ denote the nonzero complex numbers. For a group G with operation $*$, a *character* on G is a function $\chi : G \to \mathbb{C}^*$ satisfying

$$\chi(g * h) = \chi(g) \cdot \chi(h)$$

In mathematical terms, χ is a homomorphism from G into $\mathbb{C}^*$.

We observed that a couple of properties were true of these characters:

- $\chi_{\mathbf{0}} \equiv 1$
- $\underset{x \sim \{0,1\}^n}{\mathbf{E}}[\chi_\gamma(x)] = 0$ for $\gamma \neq \mathbf{0}$.
- $\chi_\alpha(z) = \chi_z(\alpha)$

  This one is new. Though it's true, conceptually, it's generally good to segment the inputs from the index set of the Fourier coefficients.
- $\langle \chi_\gamma | g \rangle = \underset{x \sim \{0,1\}^n}{\mathbf{E}}[\chi_\gamma(x) g(x)] = \widehat{g}(\gamma)$

  This is a statement of a more general fact about orthonormal bases: if $\mathcal{F}$ is any orthonormal basis, the coefficient on $f \in \mathcal{F}$ in the representation of v with respect to $\mathcal{F}$ is given by $\langle f, v \rangle$.

As a consequence of the above properties, we had one further property:

- $\chi_\gamma(x)\chi_\sigma(x) = \chi_{\gamma+\sigma}(x)$

In summary, the Fourier transform over $\mathbb{Z}_2^n$ is defined as the unitary transformation that takes a function into its representation with respect to the χ_γ:

$$\frac{1}{\sqrt{N}} \sum_{x \in \{0,1\}^n} g(x) |x\rangle = \frac{1}{\sqrt{N}} \begin{bmatrix} g(0,\ldots,0) \\ g(0,\ldots,0,1) \\ \vdots \\ g(1,\ldots,1) \end{bmatrix} \mapsto \begin{bmatrix} \widehat{g}(0,\ldots,0) \\ \widehat{g}(0,\ldots,0,1) \\ \vdots \\ \widehat{g}(1,\ldots,1) \end{bmatrix} = \sum_{\gamma \in \mathbb{Z}_2^n} \widehat{g}(\gamma) |\gamma\rangle$$

Example 2.2. *Let*

$$g_x(y) = \begin{cases} \sqrt{N} & y = x \\ 0 & o.w. \end{cases}$$

then the vector representation of g_x is

$$\frac{1}{\sqrt{N}} \sum_{y \in \{0,1\}^n} g_x(y) |y\rangle = |x\rangle$$

and

$$\widehat{g}(\gamma) = \underset{y \sim \{0,1\}^n}{\mathbf{E}}[g_x(y)(-1)^{\gamma \cdot y}] = \frac{1}{N}\sqrt{N}(-1)^{\gamma \cdot x} = \frac{1}{\sqrt{N}}(-1)^{\gamma \cdot x}$$

$$|x\rangle = \frac{1}{\sqrt{N}} \sum_{\gamma \in \mathbb{Z}_2^n} (-1)^{\gamma \cdot x} |\chi_\gamma\rangle$$

Remark 2.3. Another way to compute the Fourier expansion of $|x\rangle$ is to note that the Fourier expansion of χ_γ is $|\gamma\rangle$, and that the Fourier transform is an involution (since H_N is its own inverse).

Representing a function with respect to this basis revealed patterns specific to the $\mathbb{Z}_2^n$ group structure. The main advantage of quantum computing, though, is that the Fourier transform over $\mathbb{Z}_2^n$ is efficiently computable on a quantum computer. The matrix that implements it, H_N, consists of exactly n gates. In comparison to the classical situation, the fastest known method to compute the Fourier expansion is the fast Walsh-Hadamard transform, which requires $O(N \log N)$ time. See [FA76] for an overview.

All this begs the questions: can we do the same for $\mathbb{Z}_N$? What are the characters on $\mathbb{Z}_N$, and do they form an orthonormal basis? If such a unitary transformation exists, can we implement it efficiently on a quantum computer? As we will now see, the answer is yes.

2.2 An Orthogonal Basis of Characters

Let's look first at the characters on $\mathbb{Z}_N$.

Theorem 2.4. *$\mathbb{Z}_N$ has exactly N characters.*

Proof. Let's try and first deduce what form the characters must have. If χ is a character, for any $x \in \mathbb{Z}_N$ we have

$$\chi(x) = \chi(x+0) = \chi(x)\chi(0)$$

If $\chi(x) \neq 0$ for some x, we can conclude $\chi(0) = 1$. We'll ignore the case where $\chi \equiv 0$, as this is the zero vector in this vector space and won't be helpful to forming an orthonormal basis. So let's conclude $\chi(0) = 1$.

We also know

$$\chi(\overbrace{x + \cdots + x}^{k \text{ times}}) = \chi(x)^k$$

In particular, if we take $k = N$, then $kx = Nx = 0$, modulo N. Combining this with the above, we have, for every $x \in \mathbb{Z}_N$,

$$\chi(x)^N = 1$$

That is, $\chi(x)$ is an N-th root of unity. We also know

$$\chi(k) = \chi(\overbrace{1 + \cdots + 1}^{k \text{ times}}) = \chi(1)^k$$

χ is completely determined by its value on 1! Let $\omega = e^{i\frac{2\pi}{N}}$ be a primitive N-th root of unity. $\chi(1)$ is an N-th root of unity, so write

$$\chi(1) = \omega^\gamma$$

for some $\gamma \in \mathbb{Z}_N$. From all of this we deduce that χ must be given by the formula

$$\chi(x) = \omega^{\gamma \cdot x}$$

where $\gamma \cdot x$ is regular integer multiplication.

For each $\gamma \in \mathbb{Z}_N$, define the function $\chi_\gamma : \mathbb{Z}_N \to \mathbb{C}$ by $\chi_N(x) = \omega^{\gamma \cdot x}$. To complete the theorem we check that every γ creates a distinct character i.e. χ_γ satisfies the homomorphism property:

$$\chi_\gamma(x+y) = \omega^{\gamma(x+y)} = \omega^{\gamma \cdot x + \gamma \cdot y} = \omega^{\gamma \cdot x}\omega^{\gamma \cdot y} = \chi_\gamma(x)\chi_\gamma(y)$$

□

With the set $\{\chi_\gamma\}_{\gamma \in \mathbb{Z}_N}$ in hand, we can check that these do indeed form an orthonormal basis. First we check that the analogous properties from the Fourier basis over $\mathbb{Z}_2^n$ carry over:

- $\chi_0 \equiv 1$

  Proof. $\chi_0(x) = \omega^{0 \cdot x} = 1$ □

- $\mathop{\mathbf{E}}_{x \sim \mathbb{Z}_N}[\chi_\gamma(x)] = 0$ for $\gamma \neq 0$.

  Proof. This is a happy fact about roots of unity. Geometrically, the powers of ω are equally spaced around the unit circle, and will cancel upon summation. Algebraically, when $\gamma \neq 0$,

  $$\frac{1}{N}\sum_{i=0}^{N-1} \omega^{\gamma x} = \frac{1}{N}\frac{\omega^{\gamma N} - 1}{\omega^\gamma - 1} = 0$$

  □

- $\chi_\alpha(z) = \chi_z(\alpha)$

  Proof. $\chi_\alpha(z) = \omega^{\alpha \cdot z} = \omega^{z \cdot \alpha} = \chi_z(\alpha)$ □

As before these can be used to deduce one further property,

- $\chi_\sigma(x)\chi_\gamma(x) = \chi_{\sigma+\gamma}(x)$

There's also a new, very important property that we wouldn't have noticed before, when we were only working over $\mathbb{R}$:

- $\chi_\gamma(x)^* = \chi_{-\gamma}(x) = \chi_\gamma(-x)$

Proof.

$$\begin{aligned}\chi_\gamma(x)^* = (\omega^{\gamma\cdot x})^* = \omega^{-\gamma\cdot x} &= \omega^{(-\gamma)\cdot x} = \chi_{-\gamma}(x) \\ &= \omega^{\gamma\cdot(-x)} = \chi_\gamma(-x)\end{aligned}$$

□

From these, the orthonormality of the characters falls right out:

Theorem 2.5. *The functions $\{\chi_\gamma\}_{\gamma\in\mathbb{Z}_N}$ form an orthonormal basis.*

Proof. For $\sigma, \gamma \in \mathbb{Z}_N$,

$$\begin{aligned}\langle\chi_\sigma|\chi_\gamma\rangle &= \mathop{\mathbf{E}}_{x\sim\mathbb{Z}_N}[\chi_\sigma(x)^*\chi_\gamma(x)] \\ &= \mathop{\mathbf{E}}_{x\sim\mathbb{Z}_N}[\chi_{-\sigma}(x)\chi_\gamma(x)] \\ &= \mathop{\mathbf{E}}_{x\sim\mathbb{Z}_N}[\chi_{\gamma-\sigma}(x)] \\ &= \begin{cases} 1 & \gamma-\sigma=0 \\ 0 & \text{o.w.}\end{cases}\end{aligned}$$

We have N orthonormal elements in a space of dimension N, implying they form a basis for the space. □

As before, we use the notation $\widehat{g}(\gamma)$ to denote the coefficient on χ_γ in the representation of a function g with respect to this basis. The notation has almost the same form as before:

$$g(x) = \sum_{\gamma\in\mathbb{Z}_N} \widehat{g}(\gamma)\,|\chi_\gamma\rangle$$

The difference is in how we think about the domain of summation: in the first case it was $\mathbb{Z}_2^n$, and now it is $\mathbb{Z}_N$.

Also as before, and as a consequence of orthonormality, we have a method of computing the Fourier coefficients:

$$\widehat{g}(\gamma) = \langle\chi_\gamma|g\rangle = \mathop{\mathbf{E}}_{x\sim\{0,1\}^n}[\chi_\gamma(x)^* g(x)]$$

We call the unitary transformation that takes a function to its Fourier representation the Fourier transform over $\mathbb{Z}_2^n$:

$$\frac{1}{\sqrt{N}}\sum_{x\in\{0,1\}^n} g(x)\,|x\rangle = \frac{1}{\sqrt{N}}\begin{bmatrix} g(0,\dots,0) \\ g(0,\dots,0,1) \\ \vdots \\ g(1,\dots,1)\end{bmatrix} \mapsto \begin{bmatrix} \widehat{g}(0,\dots,0) \\ \widehat{g}(0,\dots,0,1) \\ \vdots \\ \widehat{g}(1,\dots,1)\end{bmatrix} = \sum_{\gamma\in\mathbb{Z}_N}\widehat{g}(\gamma)\,|\gamma\rangle$$

Example 2.6. *Let*

$$g_x(y) = \begin{cases} \sqrt{N} & y = x \\ 0 & o.w. \end{cases}$$

Then the vector representation of g_x is $|x\rangle$, and

$$\widehat{g}_x(\gamma) = \mathop{\mathbf{E}}_{y\sim\{0,1\}^n}[\chi_\gamma(y)^* g_x(y)] = \chi_\gamma(x)^* \qquad\qquad |x\rangle = \frac{1}{\sqrt{N}} \sum_{\gamma\in\mathbb{Z}_N} \chi_\gamma(x)^* |\chi_\gamma\rangle$$

The Fourier transform over $\mathbb{Z}_N$ will help to solve the $\mathbb{Z}_N$-analogue of Simon's problem, the period-finding problem, in the next lecture. From there it is an easy step to Shor's factorization algorithm.

One thing remains for now: efficient implementation of a circuit to compute the transform. This is what the remainder of the lecture will do.

Remark 2.7. The entire analysis above goes through without assuming that N is a power of 2. Though for the most part, we will only concern ourselves with cases where N is a power of 2.

Remark 2.8. Taking $N = 2$ gives an equivalent formulation as taking $n = 1$: the characters are $\mathbf{1}$, and $(-1)^x$.

3 Implementing the Fourier transform over $\mathbb{Z}_N$

The goal of this section is to implement the quantum Fourier transform over $\mathbb{Z}_N$ *efficiently*, that is using only poly(n) 1- or 2-qubit gates. Once we have a circuit computing the Fourier transform over $\mathbb{Z}_N$, it will be a valuable tool for use in quantum algorithms.

A life lesson to take away: if a unitary matrix has easy-to-write-down entries, it can probably be computed using poly(n) gates.

Theorem 3.1. *The quantum Fourier transform over $\mathbb{Z}_N$ can be implemented with $O(n^2)$ 1- and 2-qubit gates.*

Proof. We will build a circuit with exactly $\binom{n+1}{2}$ gates. As usual for a quantum circuit, it suffices to create a circuit that is correct on each classical input $|x\rangle$, $x \in \{0,1\}^n$; by linearity such a circuit is correct on all superpositions.

We want to implement the map

$$|x\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{\gamma\in\mathbb{Z}_N} \chi_\gamma(x)^* |\gamma\rangle$$

where $\chi_\gamma(x)^* = \omega^{-\gamma\cdot x}$. Consider as example $n = 4$, $N = 2^n = 16$

$$|x\rangle \mapsto \frac{1}{4}\big(|0000\rangle + \omega^{-x}|0001\rangle + \omega^{-2x}|0010\rangle + \omega^{-3x}|0011\rangle + \cdots + \omega^{-15x}|1111\rangle\big)$$

A key realization is that the above output state is actually *unentangled*. That is, there are qubits $|\psi_0\rangle, |\psi_1\rangle, |\psi_2\rangle, |\psi_3\rangle$ such that the above state equals $|\psi_1\rangle \otimes |\psi_2\rangle \otimes |\psi_3\rangle \otimes |\psi_4\rangle$. In particular, it is equal to

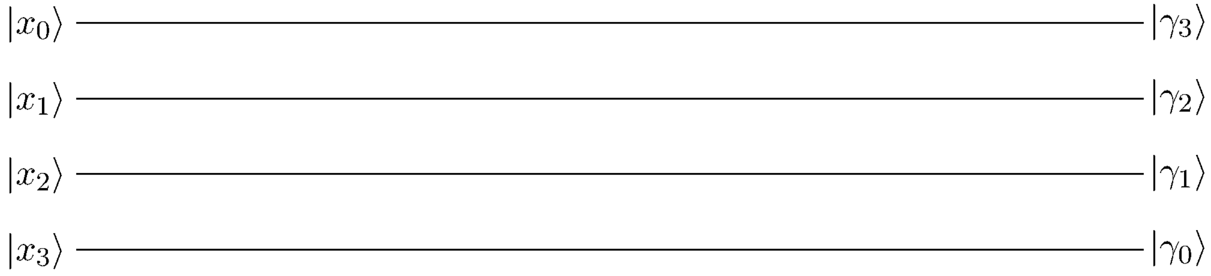
$$\left(\frac{|0\rangle + \omega^{-8x}|1\rangle}{\sqrt{2}}\right) \otimes \left(\frac{|0\rangle + \omega^{-4x}|1\rangle}{\sqrt{2}}\right) \otimes \left(\frac{|0\rangle + \omega^{-2x}|1\rangle}{\sqrt{2}}\right) \otimes \left(\frac{|0\rangle + \omega^{-x}|1\rangle}{\sqrt{2}}\right)$$

In the case for general n, we want to take

$$|x\rangle \mapsto \left(\frac{|0\rangle + \omega^{-2^n x}|1\rangle}{\sqrt{2}}\right) \otimes \left(\frac{|0\rangle + \omega^{-2^{n-1} x}|1\rangle}{\sqrt{2}}\right) \otimes \cdots \otimes \left(\frac{|0\rangle + \omega^{-x}|1\rangle}{\sqrt{2}}\right)$$

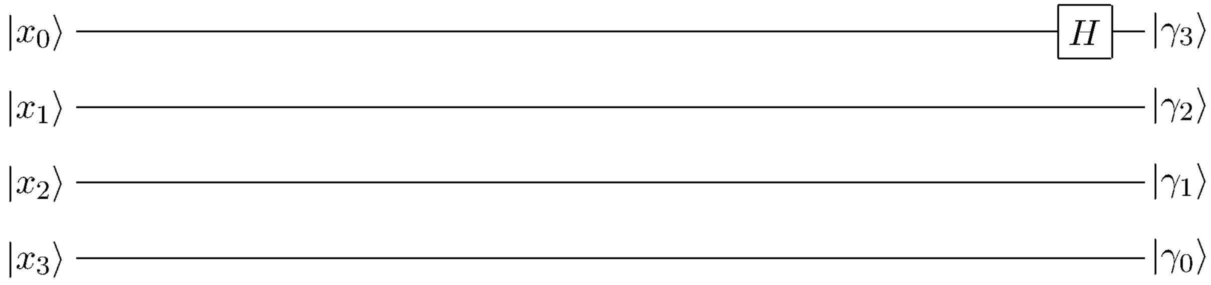
In order to do so, it suffices to perform the transformation wire-by-wire.

We return to our example. Let's write the input $|x\rangle = |x_0 x_1 \ldots x_{n-1}\rangle$ and the output $|\gamma\rangle = |\gamma_{n-1} \ldots \gamma_0\rangle$ and fill in gates as we need them:



As suggested by the notation, it's actually easier to do this transformation if we take the least significant bit of x to the most significant bit of the output. To see this, on the most significant output wire we want $\frac{1}{\sqrt{2}}(|0\rangle + \omega^{-8x}|1\rangle$. At first glance it looks like we need all of x for this. However, since $\omega^N = \omega^{16} = 1$, we only need the least significant bit $|x_0\rangle$. A similar situation occurs for other wires, as we will see. We can finish the circuit by reversing the order of the output bits, for example implemented by $\frac{n}{2}$ XOR swaps.

As we just computed, the most significant output bit will be $\frac{1}{\sqrt{2}}(|0\rangle + (-1)^{-x_0}|1\rangle)$. This is exactly the Hadamard gate applied to input $|x_0\rangle$. Here's a circuit that correctly computes the first wire:



What about the second wire? We want to perform

$$|x_1\rangle \mapsto \frac{1}{\sqrt{2}}(|0\rangle + \omega^{-4x}|1\rangle)$$

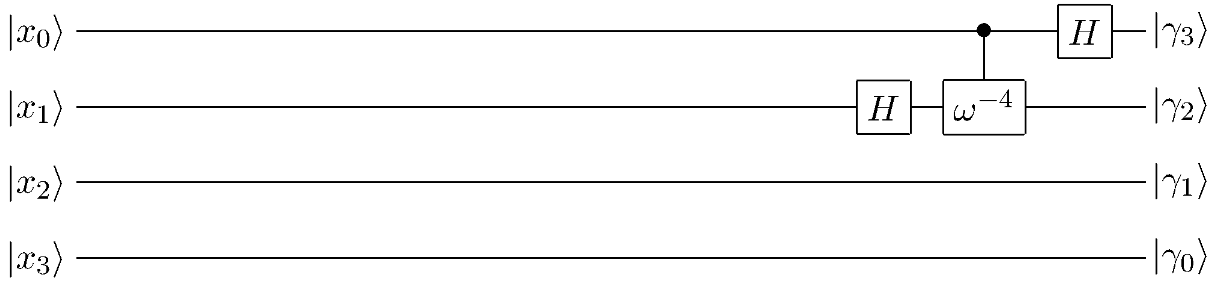
This time, ω^{-4x} depends only on the two lowest bits of x, and is equal to $\omega^{-8x_1-4x_0} = (-1)^{-x_1}\omega^{-4x_0}$. We need:

$$|x_1\rangle \mapsto \frac{1}{\sqrt{2}}(|0\rangle + (-1)^{-x_1}\omega^{-4x_0}|1\rangle)$$

First, apply Hadamard to $|x_1\rangle$ to get $\frac{1}{\sqrt{2}}(|0\rangle+(-1)^{-x_0}|1\rangle)$. What we need now is a "controlled ω^{-4} gate": if $x_0 = 1$, multiply the amplitude of $|x_1\rangle$ on $|1\rangle$ by ω^{-4}, else do nothing. Under the assumption that we can implement 1- and 2-qubit gates, use a gate that implements the (unitary) matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \omega^{-4} \end{pmatrix}$$

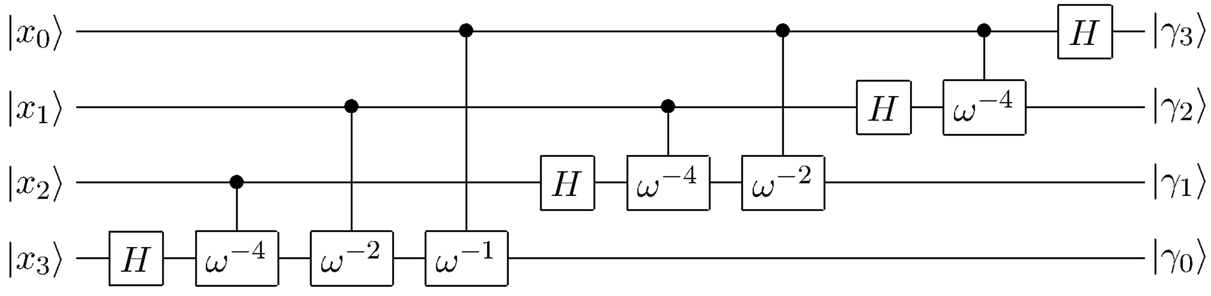
Adding these two gates to the circuit correctly computes the second wire.



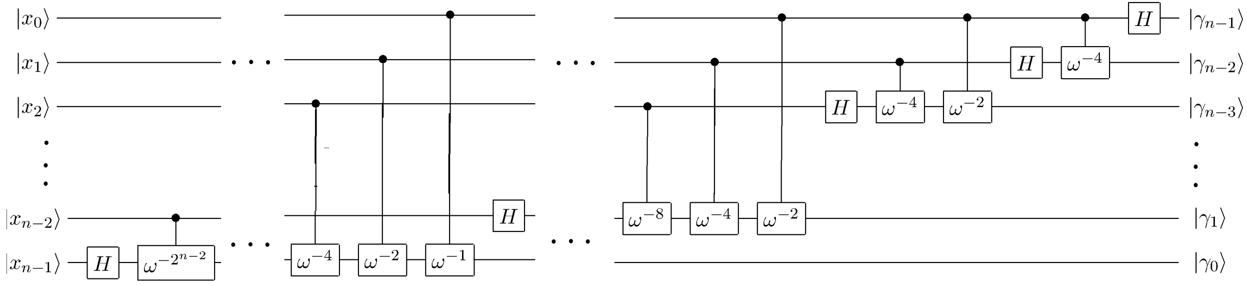
The situation for other wires, and other n, is similar. To finish off the example, we need to compute

$$|x_2\rangle \mapsto \left(\frac{|0\rangle + \omega^{-2x}|1\rangle}{\sqrt{2}}\right) = \left(\frac{|0\rangle + (-1)^{x_2}\omega^{-4x_1-2x_0}|1\rangle}{\sqrt{2}}\right)$$
$$|x_3\rangle \mapsto \left(\frac{|0\rangle + \omega^{-x}|1\rangle}{\sqrt{2}}\right) = \left(\frac{|0\rangle + (-1)^{-x_3}\omega^{-4x_2-2x_1-x_0}|1\rangle}{\sqrt{2}}\right)$$

We can build controlled ω^{-2} and ω^{-1} gates as well. Use the same paradigm: hit each wire with a Hadamard, then use the controlled ω^{-2^k} gates on the lower-order bits. The final circuit looks like this:



To generalize this to any n, transform on $|x_i\rangle$ by first applying a Hadamard gate, then applying a controlled ω^{-2^k} gate controlled by $|x_k\rangle$, for each $k < i$. This works so long as we first transform wire $|x_{n-1}\rangle$, then $|x_{n-2}\rangle$, and so on down to $|x_0\rangle$; no wire depends on wires that are below it. The circuit looks something like:



With the aforementioned reversing of the output wires, this completes the circuit to compute the Fourier transform over $\mathbb{Z}_N$. The total number of gates in this part of the circuit $1 + 2 + \cdots + n = \binom{n+1}{2} = O(n^2)$. Including the $O(n)$ for swapping gives $O(n^2)$ size overall.

□

Remark 3.2. The uncontrolled, 1-qubit version of the controlled ω^{-2^k} is called the "phase shift gate", with matrix

$$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix}$$

This gate does not change the probability of outcome $|0\rangle$ or $|1\rangle$ for this qubit.

Remark 3.3. One can compute the transform to Fourier transform over $\mathbb{Z}_N$ to accuracy ϵ using only $O(n \log \frac{n}{\epsilon})$ gates. The idea is simple enough: phase shifts of very small amounts will change the overall quantum state very little, so if we allow ourselves a little error we can afford to skip a few. A more exact analysis can be found in [HH00].

References

[FA76] B.J. Fino and V.R. Algazi. Unified matrix treatment of the fast walsh-hadamard transform. *IEEE Transactions on Computers*, 25(11):1142–1146, 1976.

[HH00] L. Hales and S. Hallgren. An improved quantum fourier transform algorithm and applications. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 515–525, 2000.