LECTURE 12 February 26^{th}, 2025

PART II Fundamental Quantum Algorithms

Today Simon's Algorithm (wrapup)
Quantum Fourier Transform

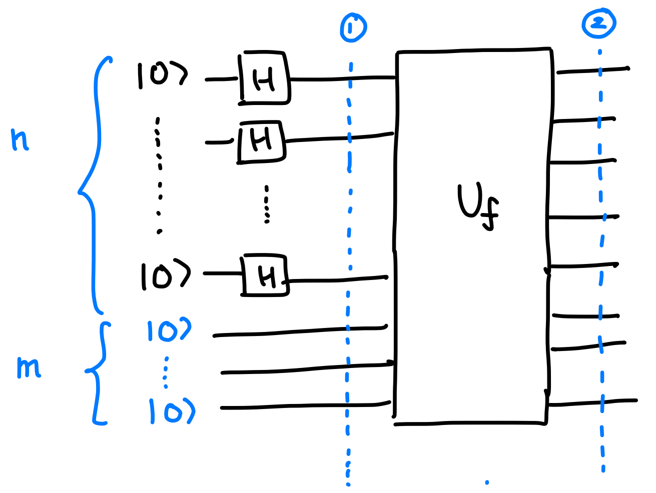
RECAP Given black-box access to f that is L-periodic, determine $L \in \{0,1\}^n$

Black-box access : $\underbrace{|x\rangle}_{n}\underbrace{|y\rangle}_{m} \to \boxed{U_f} \to \underbrace{|x\rangle}_{n}\underbrace{|y \oplus f(x)\rangle}_{m=n-1}$

L-periodic : $f(x) = f(y)$ iff $x \oplus y = L$ $\Rightarrow$ pairs $(x, x+L)$ get a distinct color

$(1.4)^n$ classical vs $4n$ quantum

The algorithm ① Evaluate f on all the inputs in superposition



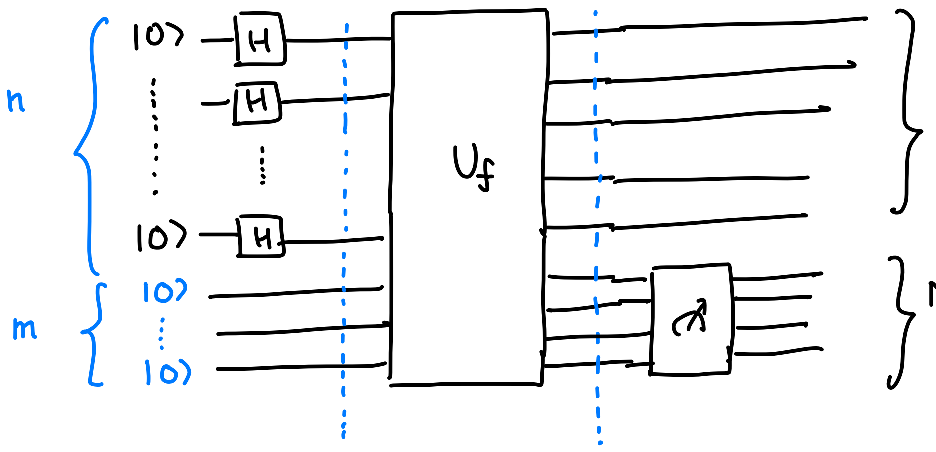
$$= \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x, f(x)\rangle$$ ← All queries evaluated in a superposition

E.g. the state at ② is

$$\frac{1}{\sqrt{8}}\left(|000\rangle \otimes |\text{RED}\rangle + |001\rangle \otimes |\text{YELLOW}\rangle + \dots\right)$$

Goal Learn one bit of information about L from this superposition

① Measure the ancillas



?

Measurement Outcome is a COLOR
Let's call it $c^* \in \{0,1\}^m$

$$\mathbb{P}[\text{measuring } c^*] = \frac{1}{\#\text{COLORS}} = \frac{1}{2^{n-1}}$$

And the joint state becomes $\frac{1}{\sqrt{2}}|x^*\rangle|c^*\rangle + \frac{1}{\sqrt{2}}|x^*+L\rangle|c^*\rangle$

where x^* and x^*+L are the pairs where f has value c^*

E.g. $\frac{1}{\sqrt{8}}\left(|000\rangle \otimes |\text{RED}\rangle + |001\rangle \otimes |\text{YELLOW}\rangle + \dots + |101\rangle \otimes |\text{RED}\rangle + \dots\right)$

$\mathbb{P}[\text{each color}] = \frac{1}{4}$

and if we measure RED, (joint) state collapses to

$$\frac{1}{\sqrt{2}}|000\rangle \otimes |\text{RED}\rangle + \frac{1}{\sqrt{2}}|101\rangle \otimes |\text{RED}\rangle$$

So, State of the first n qubits becomes $\frac{1}{\sqrt{2}}|x^*\rangle + \frac{1}{\sqrt{2}}|x^*+L\rangle$

This is very simple state! Almost looks like we are done! But are we?

Let us try some natural things

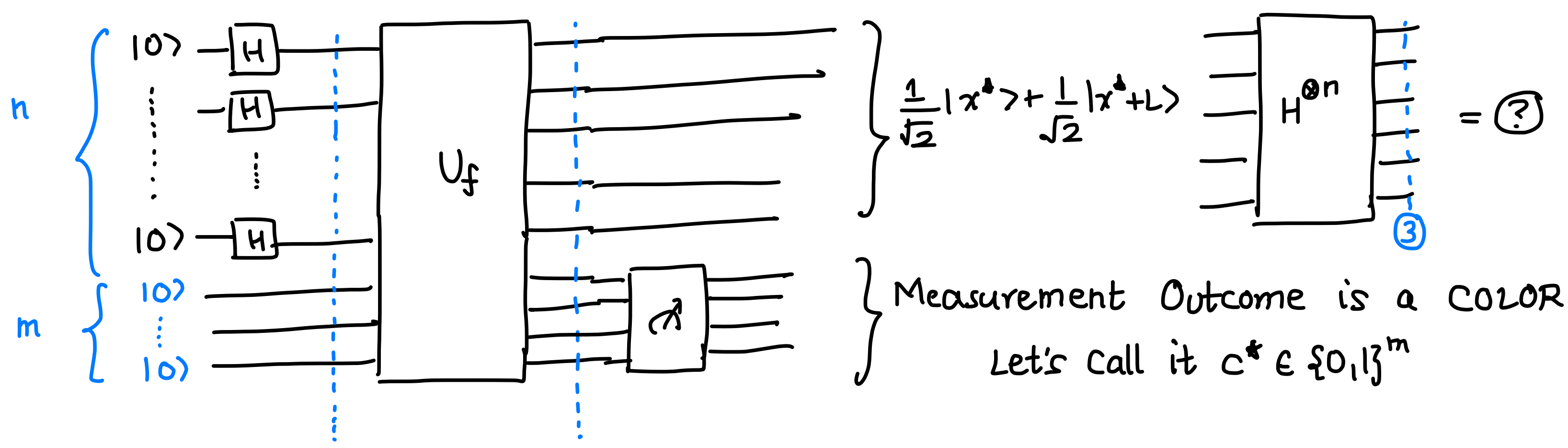
Try 1 Measure with 50% chance get x^* and x^*+L
but can't do it twice with one copy of the state
since it's destroyed after measurement

Try 2 Prepare another copy

but we will get a different c^* and the pair
associated to that $\rightarrow$ Again not helpful

Try 3 Unitary transformation on $\frac{1}{\sqrt{2}}|x^*\rangle + \frac{1}{\sqrt{2}}|x^*+L\rangle$

③ Let's apply a Hadamard gate H on each qubit and see what happens!



At step 3, the state is $H^{\otimes n}\left(\frac{1}{\sqrt{2}}|x^*\rangle + \frac{1}{\sqrt{2}}|x^*+L\rangle\right)$

$$= \frac{1}{\sqrt{2}} H^{\otimes n}|x^*\rangle + \frac{1}{\sqrt{2}} H^{\otimes n}|x^*+L\rangle$$

What is $H^{\otimes n}|x\rangle$? E.g. if $|x\rangle = |0 \cdots 0\rangle$

$$H^{\otimes}|0 \cdots 0\rangle = (H|0\rangle)\otimes(H|0\rangle)\otimes \cdots \otimes(H|0\rangle)$$

$$= |+\rangle^{\otimes n} = \left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right)^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{s\in\{0,1\}^n} |s\rangle$$

$$H^{\otimes n}|x_1 \cdots x_n\rangle = (H|x_1\rangle)\otimes(H|x_2\rangle)\otimes \cdots \otimes (H|x_n\rangle)$$

$H|0\rangle = |+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$

$$= \left(\frac{1}{\sqrt{2}}|0\rangle + (-1)^{x_1}|1\rangle\right)\otimes \cdots \otimes\left(\frac{1}{\sqrt{2}}|0\rangle + (-1)^{x_n}|1\rangle\right)$$

$H|1\rangle = |-\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$

$$= \frac{1}{\sqrt{2^n}} \sum_{s\in\{0,1\}^n} (-1)^{x_1 \cdot s_1 + \cdots + x_n s_n} |s\rangle$$

So, the state at step 3 is

$$\frac{1}{\sqrt{2^{n+1}}} \sum_{s\in\{0,1\}^n} \left((-1)^{x^*\cdot s}|s\rangle + \frac{1}{\sqrt{2^{n+1}}} (-1)^{(x^*+L)\cdot s}|s\rangle \right)$$

$$= \frac{1}{\sqrt{2^{n+1}}} \sum_s (-1)^{x^*\cdot s}|s\rangle \underbrace{\left(1 + (-1)^{L\cdot s}\right)}$$

either $\begin{cases} 2 & \text{if } L\cdot s = 0 \bmod 2 \\ 0 & \text{if } L\cdot s = 1 \bmod 2 \end{cases}$

$$= \sqrt{\frac{2}{2^n}} \sum_{s:\, s\cdot L = 0} (-1)^{x^*\cdot s}|s\rangle$$

Half of all $s\in\{0,1\}^n$ satisfy $s\cdot L = 0 \bmod 2$
i.e. $\frac{2^n}{2}$ such string s in the sum

What happens if we measure this state now ?

We get a uniformly random $s\in\{0,1\}^n$ such that $s\cdot L = 0 \bmod 2$

Note: All the information about x^* went away!!

This is one bit of information about L
For example if $s = 0\cdots 1 0 \cdots 0$ had a single 1 coordinate
we learn that particular bit of L

In general, we get a linear equation $s \cdot L = 0 \mod 2$ for a random s

We know s explicitly e.g. $s = 1001110000$

$L = L_1 L_2 \ldots L_n$

$\Rightarrow \quad L_1 + L_4 + L_5 + L_6 = 0 \mod 2$

We can repeat this whole quantum subroutine T times and get T linear equations

$$s^{(1)} \cdot L = 0$$
$$s^{(2)} \cdot L = 0$$
$$\vdots$$
$$s^{(T)} \cdot L = 0$$

$\rightarrow$ Each equation reduces # of possible L's by $\frac{1}{2}$ and we can stop if there are exactly 2 solutions the true secret string L & 0

If these contain $n-1$ linearly independent equations, we know L exactly $\leftarrow$ Classical algorithm such as Gaussian Elimination

To summarize:
- Quantum subroutine gives us a random s satisfying $s \cdot L = 0$
- Collect T such strings which gives T linear equations (mod 2)
- Solve them classically

Claim $\quad \mathbb{P}[\text{first } n-1 \ s^{(1)}, \ldots s^{(n-1)} \text{ are linearly independent}] \geq \frac{1}{4}$

Start again if they are not

$$\mathbb{E}[\# \text{ applications of } U_f \text{ until we succeed}] \leq 4n$$

Proof of Claim $\quad$ Assume $s^{(1)}, \ldots, s^{(i)}$ are linearly independent

i.e. they span a subspace $\{\alpha_1 s^{(1)} + \alpha_2 s^{(2)} + \cdots + \alpha_i s^{(i)} \mid \alpha_1, \ldots \alpha_i \in \{0,1\}\}$ which has size 2^i

The next $s^{(i+1)}$ is independent if it is not in the span

$$\mathbb{P}[\underbrace{s^{(i+1)} \in \text{span}\{s^{(1)}, \ldots s^{(i)}\}}_{\text{bad event}}] = \frac{2^i}{2^{n-1}}$$

$$\mathbb{P}[\text{good}: \ s^{(i+1)} \notin \text{span}\{s^{(1)}, \ldots, s^{(i)}\}] = 1 - \frac{2^i}{2^{n-1}}$$

$$\mathbb{P}[\text{all } n\text{-1 are linearly independent}] = \left(1-\frac{1}{2^{n-1}}\right)\left(1-\frac{2}{2^{n-1}}\right)\left(1-\frac{4}{2^{n-1}}\right)\cdots\left(1-\frac{2^{n-2}}{2^{n-1}}\right)$$

$\mathbb{P}[s^{(1)} \notin \text{span}\{0\}]$

$\mathbb{P}[s^{(n-1)} \notin \text{span}\{s^{(1)}, \dots s^{(n-2)}\}]$

$$= \frac{1}{2}\cdot\frac{3}{4}\cdot\frac{7}{8}\cdots\left(1-\frac{4}{2^{n-1}}\right)\left(1-\frac{2}{2^{n-1}}\right)\left(1-\frac{1}{2^{n-1}}\right)$$

$$\geq \frac{1}{2}\left(1-\frac{1}{4}\right)\left(1-\frac{1}{8}\right)\left(1-\frac{1}{16}\right)\cdots$$

$$\geq \frac{1}{2}\left(1-\frac{1}{4}-\frac{1}{8}-\frac{1}{16}-\cdots\right)$$

$$= \frac{1}{2}\cdot\frac{1}{2} = \frac{1}{4} \qquad \blacksquare$$

NEXT: Buildup to Shor's Factoring Algorithm

Given N, find p, q s.t. $p \cdot q = N$

Shor's algorithm uses a similar subroutine over integers mod N called ORDER FINDING

ORDER FINDING: $f(x) = a^x \bmod N$ where a is uniform random number coprime to N

find L (dividing N) s.t. $f(x) = f(x+L) = f(x+2L) = f(x+3L) \cdots$

Main differences with Simon's problem:

1. Arithmetic mod N where N is a large number

2. No promise that L divides N, so need some results from number theory to deal with it

3. We can build the black-box ourselves $\longrightarrow$ This is what makes it practical!

The algorithm for order finding is similar to Simon's algorithm but we need an analog of $H^{\otimes n}$ that works mod N

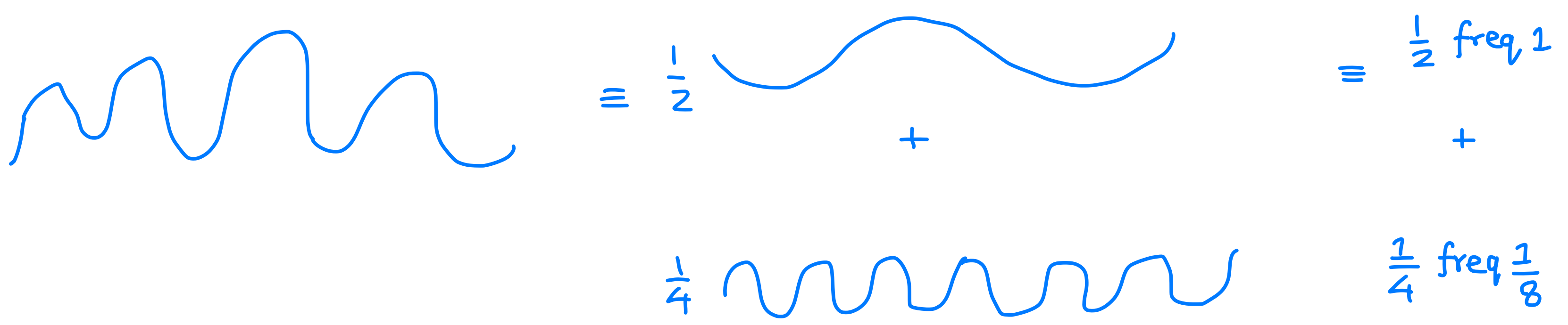
This is the Quantum Fourier Transform which we now introduce

Quantum Fourier Transform

Let us first talk about the classical discrete Fourier Transform

Useful in recovering periodic structure in data

E.g. continuous fourier transform allows



Time domain

Frequency domain

Discrete Fourier Transform (DFT)

Given $f : \mathbb{Z}_N \to \mathbb{C}$

$$\begin{array}{c} |0\rangle \\ \vdots \\ |N-1\rangle \end{array} \begin{bmatrix} f(0) \\ \vdots \\ f(N-1) \end{bmatrix} = \sum_{s=0}^{N-1} f(s)|s\rangle = \sum_{s=0}^{N-1} \hat{f}(s)|v_s\rangle$$

where $\{|v_0\rangle, \ldots |v_{N-1}\rangle\}$ is a different basis called the $\mathbb{Z}_N$- Fourier basis and $\hat{f}(i)$ are the Fourier coefficients

"TIME" domain
Standard basis

"FREQ" domain
Fourier basis

(Inverse) DFT matrix $\quad DFT_N = \sum_{s=0}^{N-1} |v_s\rangle\langle s| = \begin{bmatrix} | & | & & | \\ |v_0\rangle & |v_1\rangle & \cdots & |v_{N-1}\rangle \\ | & | & & | \end{bmatrix}$

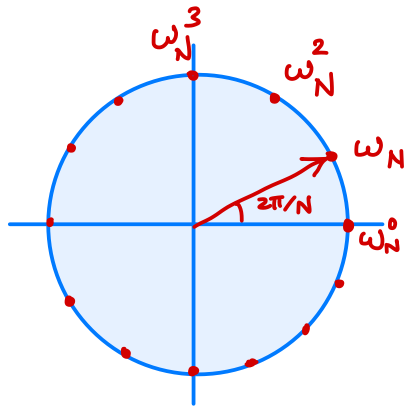
Unitary Matrix

$DFT_N^{-1} = DFT_N^{\dagger}$

E.g. $N=2$ $\quad DFT_2 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = H$

For general N, we need complex numbers

Let $\omega_N = e^{2\pi i/N}$ be the primitive N^{th}-root of unity

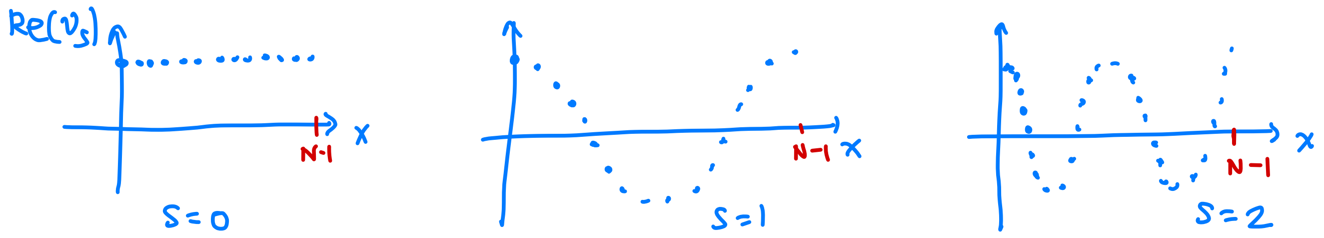


$$\omega_N = e^{\frac{2\pi i}{N}} = \cos\left(\frac{2\pi}{N}\right) + i\sin\left(\frac{2\pi}{N}\right)$$

$\boxed{\text{FACT}}$ $$1 + \omega_N + \omega_N^2 + \dots + \omega_N^{N-1} = \frac{1-\omega_N^N}{1-\omega_N} = 0$$

$$DFT_N = \frac{1}{\sqrt{N}} \begin{array}{cc} & s \\ x & \begin{bmatrix} & \vdots & \\ \cdots & \omega_N^{sx} & \\ & & \end{bmatrix} \end{array} \quad \text{so,} \quad |v_s\rangle = \begin{bmatrix} \omega_N^0 \\ \omega_N^s \\ \omega_N^{2s} \\ \vdots \\ \omega_N^{(N-1)s} \end{bmatrix}$$

Plotting Real parts of v_s the graph looks like a discrete cosine wave



E.g (N=4) $$DFT_4 = \frac{1}{\sqrt{4}} \begin{bmatrix} \omega_4^0 & \omega_4^0 & \omega_4^0 & \omega_4^0 \\ \omega_4^0 & \omega_4^1 & \omega_4^2 & \omega_4^3 \\ \omega_4^0 & \omega_4^2 & \omega_4^4 & \omega_4^6 \\ \omega_4^0 & \omega_4^3 & \omega_4^6 & \omega_4^9 \end{bmatrix}$$ → can express mod 4 since $\omega_4^4 = 1$

DFT_N^{-1} = Conjugate Transpose of DFT_4
= put negative signs in the exponent

One can compute discrete fourier transform of any vector in $\approx N\log N$ time classically

However, since DFT_N is a unitary matrix, one can applying it to a quantom state

NOTE The coefficients in standard and Fourier basis are encoded as amplitudes unlike the classical case where one can write the N coeffecients on a piece of paper

The advantage is that one can IMPLEMENT DFT_N for $N = 2^n$ with

$O(n^2)$ quantum gates (1 and 2 qubit gates)

$O(2^n \cdot n)$ time classically, so exponential savings but here we get a quantum state

Let's see how to do this by example, Say $N = 16$

We want to implement $|x\rangle \xrightarrow{DFT_{16}} \frac{1}{\sqrt{16}} \sum_{s=0}^{N-1} \omega_{16}^{sx} |s\rangle$ where $\omega_{16} = e^{\frac{2\pi i}{16}} := \omega$

$$DFT_{16} |x\rangle = \frac{1}{4} \left(|0000\rangle + \omega^{x} |0001\rangle + \omega^{2x} |0010\rangle + \omega^{3x} |0011\rangle + \dots + \omega^{15x} |1111\rangle \right)$$

Is this state entangled ? No!

$$= \underbrace{\left(\frac{|0\rangle + \omega^{8x} |1\rangle}{\sqrt{2}} \right)}_{|s_3\rangle} \otimes \underbrace{\left(\frac{|0\rangle + \omega^{4x} |1\rangle}{\sqrt{2}} \right)}_{|s_2\rangle} \otimes \underbrace{\left(\frac{|0\rangle + \omega^{2x} |1\rangle}{\sqrt{2}} \right)}_{|s_1\rangle} \otimes \underbrace{\left(\frac{|0\rangle + \omega^{x} |1\rangle}{\sqrt{2}} \right)}_{|s_0\rangle}$$

Compare this to the following step in Simon's algorithm:

$H^{\otimes n} |x\rangle = |+\rangle \otimes |-\rangle \otimes |+\rangle \otimes \dots$ output qubit i depends only on input qubit x

if $x_3 = 0$

if $x_2 = 1$

For DFT, each output qubit depends on all n-input qubits

We will do the transform qubit-by-qubit

It will be very convenient to reverse the order

least significant bit $|x_0\rangle$ ———— $|s_3\rangle$ most sig.

$|x_1\rangle$ ———— $|s_2\rangle$

$|x_2\rangle$ ———— $|s_1\rangle$

most $|x_3\rangle$ ———— $|s_0\rangle$ least sig

$x = x_3 x_2 x_1 x_0$

One can do $\frac{n}{2}$ SWAP gates to reverse the order at the end

To do the 0^{th} wire, we need to get $\frac{|0\rangle + \omega^{8x}|1\rangle}{\sqrt{2}}$ ← Seems like this depends on all 4 qubits of x

Notice, $\omega^8 = \omega_{16}^8 = (-1)$

So, $\omega^{8x} = (-1)^x$ and it only depends on whether x is even or odd, i.e. on x_0

So, we want $\frac{|0\rangle + (-1)^{x_0}|1\rangle}{\sqrt{2}} = H|x_0\rangle$

To do the 1^{st} wire, we need to get $\frac{|0\rangle + \omega^{4x}|1\rangle}{\sqrt{2}}$ ← Seems like this depends on all 4 qubits of x again

$\omega^4 = i$, so $\omega^{4x} = i^x$ ← only depends on x mod 4 i.e. x_0 and x_1

$$\omega^{4x} = \omega_{16}^{4(x_0 + 2x_1 + \cancel{4x_2} + \cancel{8x_3})}$$ since $16x_2, 32x_3 = 0$

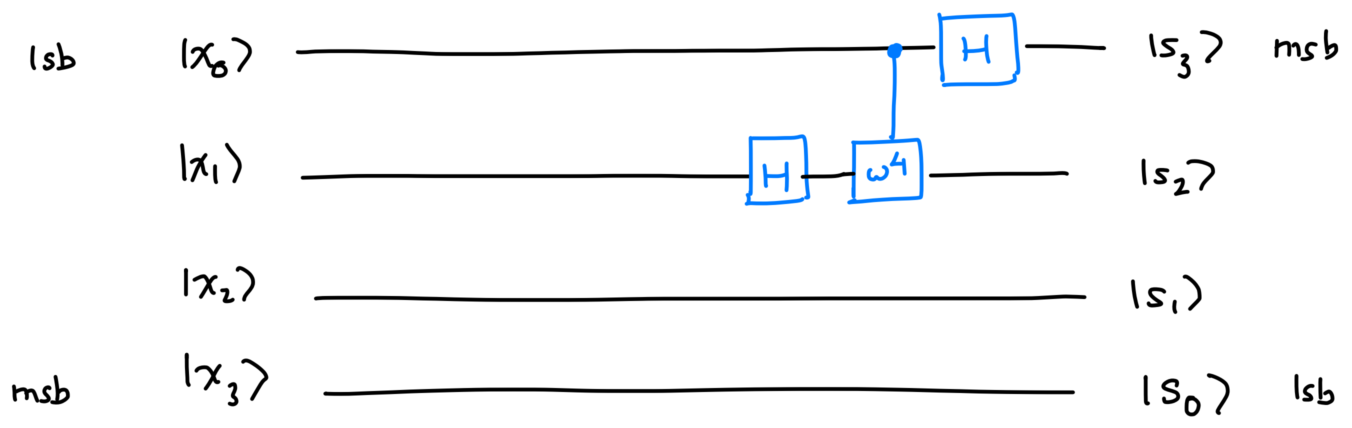
$$= \omega^{4x_0} \cdot \omega^{8x_1} = (\omega^4)^{x_0} (-1)^{x_1}$$

So, the $|1\rangle$ state should pick up phase (-1) if $x_1 = 1$ ← Hadamard
should also pick up phase ω^4 if $x_0 = 1$

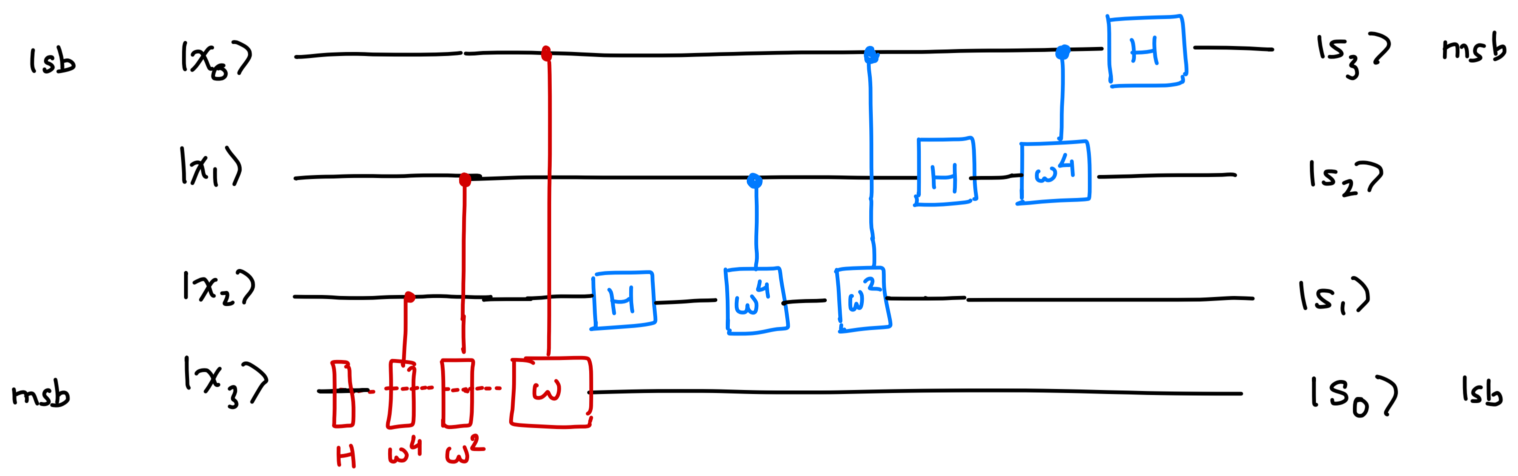
"controlled - ω^4" gate, control qubit $= x_0$

$|00\rangle \longrightarrow |00\rangle$ $\quad$ $|10\rangle \longrightarrow \omega^4|10\rangle$
$|01\rangle \longrightarrow |01\rangle$ $\quad$ $|11\rangle \longrightarrow \omega^4|11\rangle$

$$\begin{array}{c c} \begin{matrix} 00 \\ 01 \\ 10 \\ 11 \end{matrix} & \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & \omega^4 \end{bmatrix} \end{array}$$



Rest is similar, in the end we have



Total gates : $1+2+3+4+\dots+n = O(n^2)$

$\boxed{\text{Final Remarks}}$ For general n, say $n = 1000$ ω_{2^n} is the controlled 2^{1000}-th root of unity phase shift gate

We cannot build this accurately in practice

In general, not realistic for 2^k root of unity for $k \geq 30$

Luckily, it's not a problem!

$\boxed{\text{FACT}}$ Suppose we delete all gates where $k \geq \log\left(\frac{n}{\varepsilon}\right)$ E.g. $k=30$, $\varepsilon = 1\%$

Then, the resulting circuit

- "ε approximates" DFT_N $\longrightarrow$ success probability of Shor's algorithm only goes down by ε
- remaining gates can be built since they have large phases
- only $O(n \log(\frac{n}{\varepsilon}))$ gates remain $\leftarrow$ Near linear size? Way more efficient!

$\boxed{\text{NEXT TIME}}$ Buildup to Shor's Algorithm : Order Finding