Quantum Algorithms: Fourier Transform

Recap

Last Week

We saw our first two quantum algorithms:

Deutsch-Josza: input is a constant function or balanced function.

Bernstein-Vazirani: input is a function of the form $f(x) = x \cdot s \bmod 2$ and we want to find s.

The quantum algorithm for both of these was the same!

"Hadamard, Query, Hadamard"

Hadamard followed by a (phase oracle) query gives the state:

$$\frac{1}{\sqrt{2^n}} \begin{bmatrix} \vdots \\ (-1)^{f(x)} \\ \vdots \end{bmatrix}$$

Deutsch-Josza

Bernstein-Vazirani

This is a multiple of the 0^n row of $H^{\otimes n}$, or orthogonal to it.

$$f(x) = x \cdot s \bmod 2$$

This is the s-th row of $H^{\otimes n}$.

Let's look at another example of this paradigm.

Simon's Algorithm

Setup

In Simon's algorithm are given oracle access to a function $f : \{0,1\}^n \to \{0,1\}^n$.

Although the output has more bits now, the oracle works in the same way. For $x, z \in \{0,1\}^n$

$$O_f|x\rangle|z\rangle = |x\rangle|z \oplus f(x)\rangle$$

where $z \oplus f(x) \in \{0,1\}^n$ is the bitwise XOR

$$(z \oplus f(x))_i = z_i \oplus f(x)_i$$

Setup

In Simon's algorithm we will be given oracle access to a function $f : \{0,1\}^n \to \{0,1\}^n$.

Now the promise on f is that there is an $s \in \{0,1\}^n$, not all zero, such that $f(x) = f(y)$ if and only if either

1) $x = y$

2) $x = y \oplus s$

Thus for every value z in the range of f there are exactly two distinct strings x, y such that $z = f(x) = f(y)$

The goal again is to determine s.

Example

Let's see an example of an $f : \{0,1\}^3 \to \{0,1\}^3$ satisfying the promise. We can think of the range as colors.

Say that $s = 011$.

x	f(x)
000	red
001	blue
010	blue
011	red
100	green
101	black
110	black
111	green

Poll

Classical Complexity

How could you solve this problem classically?

If you find $x \neq y$ with $f(x) = f(y)$ then $s = x \oplus y$.

A probabilistic argument (birthday problem) shows such a pair exists with const. prob. in a random set of size $O(2^{n/2})$.

You can also show that any successful randomized algorithm must query the oracle $\Omega(2^{n/2})$ times.

Deterministic Algorithm

There is a also a cute $O(2^{n/2})$ deterministic algorithm.

Baby steps: Query $f(x)$ for all $x = 0 \cdots 0 x_{n/2-1} \cdots x_1 x_0$.

Giant steps: Query $f(x)$ for all $x = x_{n-1} \cdots x_{n/2} 0 \cdots 0$.

Total of $2^{n/2+1}$ many queries.

Pair XORs to secret

Now consider the secret $s = s_{n-1} \cdots s_1 s_0$.

We queried $\ell = 0 \cdots 0 s_{n/2-1} \cdots s_1 s_0$.

We also queried $h = s_{n-1} \cdots s_{n/2} 0 \cdots 0$.

And $h \oplus \ell = s$ thus $f(h) = f(\ell)$.

This means that amongst the queries we will find a pair x, y with $f(x) = f(y)$ and can determine the secret s.

Quantum algorithm

The algorithm follows the same paradigm of Hadamard, Query, Hadamard.

Step 1: $(H^{\otimes n} \otimes I^{\otimes n})|0^n\rangle|0^n\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle|0^n\rangle$

Step 2: Apply O_f.

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle|f(x)\rangle$$

Step 3: Apply $H^{\otimes n} \otimes I^{\otimes n}$.

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle |f(x)\rangle = \frac{1}{2} \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (|x\rangle + |x \oplus s\rangle) |f(x)\rangle$$

Step 3: Apply $H^{\otimes n} \otimes I^{\otimes n}$.

For exposition, we analyze this instead by first measuring the second register and then applying $H^{\otimes n} \otimes I^{\otimes n}$ to the result.

The second register doesn't change so let's focus on

$$\frac{1}{\sqrt{2}} H^{\otimes n} (|x\rangle + |x \oplus s\rangle)$$

We will use the identity

$$(x \oplus s) \cdot y \bmod 2 = x \cdot y + s \cdot y \bmod 2$$

$$H^{\otimes n}(|x\rangle + |x \oplus s\rangle)$$

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

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

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

$$\frac{1}{\sqrt{2}} H^{\otimes n}(|x\rangle + |x \oplus s\rangle)$$

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

Note that the only $|y\rangle$ with nonzero amplitude are those satisfying $s \cdot y = 0 \bmod 2$.

When we measure this state, we will obtain a uniformly random element of the set

$$Y_s = \{y : s \cdot y = 0 \bmod 2\}$$

Repeat

Now we repeat this whole process k times to obtain $y_1, \ldots, y_k \in Y_s$.

Put the $y_1, \ldots, y_k$ as the rows of a matrix A.

$$\begin{bmatrix} \cdots & y_1 & \cdots \\ \cdots & y_2 & \cdots \\ & \vdots & \\ \cdots & y_k & \cdots \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{bmatrix} = \mathbf{0}_k$$

We know that $x = \mathbf{0}_n$ and $x = s$ are two solutions to this equation over $\mathbb{F}_2^n$.

Poll

Put the $y_1, \ldots, y_k$ as the rows of a matrix A.

$$\begin{bmatrix} \cdots & y_1 & \cdots \\ \cdots & y_2 & \cdots \\ & \vdots & \\ \cdots & y_k & \cdots \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{bmatrix} = \mathbf{0}_k$$

Claim: If the rank of A over $\mathbb{F}_2^n$ is $n-1$ then $\mathbf{0}_n$ and s are the only solutions to this equation.

In this case the kernel is one-dimensional and so only contains one non-zero element because we are over $\mathbb{F}_2$.

How many times?

Now we repeat this whole process k times to obtain $y_1, \ldots, y_k$ which span a space of dimension $n-1$.

Say that $y_1, \ldots, y_\ell$ are lin. ind. with $\ell < n-1$. What is the probability $y_1, \ldots, y_\ell, y_{\ell+1}$ are lin. ind. for a random

$$y_{\ell+1} \in_R \{y : y \cdot s = 0\}$$

Over $\mathbb{F}_2$, $y_1, \ldots, y_\ell$ span at most 2^ℓ many elements.

The probability $y_{\ell+1}$ is not in the span is at least

$$1 - \frac{2^\ell}{2^{n-1}} \geq \frac{1}{2}$$

How many times?

The expected number of trials to find $n-1$ lin. ind. strings is at most $2n$.

By doing $10n$ trials we find $n-1$ lin. ind. strings with prob. at least $4/5$ by Markov's inequality.

$$\begin{bmatrix} \cdots & y_1 & \cdots \\ \cdots & y_2 & \cdots \\ & \vdots & \\ \cdots & y_k & \cdots \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{bmatrix} = \mathbf{0}_k$$

Then we can find s classically by solving this linear system using Gaussian elimination over $\mathbb{F}_2$ in time $O(n^3)$.

Simon's Complexity

In each round we perform

$$(H^{\otimes n} \otimes I^{\otimes n}) O_f (H^{\otimes n} \otimes I^{\otimes n}) |0^n\rangle |0^n\rangle$$

and measure. Perform $2n+1$ gates in each round.

Number of rounds is at most $10n$.

Total number of quantum operations is $O(n^2)$.

Then we do an $O(n^3)$ time classical computation.

This compares with the $\Omega(2^{n/2})$ classical randomized complexity.

Shor's Algorithm

Overview

- Shor's period finding algorithm

  Similar to (and inspired by) Simon's algorithm but over a different group: $\mathbb{Z}_N$.

  For this we will introduce the Fourier transform over $\mathbb{Z}_N$.

- Efficient quantum circuit for the Fourier transform

- Diversion: Application to phase estimation

- Next time:

Integer Factorization = period finding + number theory

Period Finding

Like Simon's algorithm we are given a periodic function and the goal is to find its period.

Now $f : \mathbb{Z}_N \to [M]$ and we are promised there is an s such that

1) $f(0), \ldots, f(s-1)$ are all distinct.

2) $f(x) = f(y)$ if $x - y = 0 \bmod s$

3) $s | N$

Conditions (2) and (3) mean $f(x) = f(x+s)$ for all $x \in \mathbb{Z}_N$

Example:

input	output
0	2
1	5
2	7
3	2
4	5
5	7

Non-Example:

input	output
0	2
1	5
2	7
3	2
4	5
5	7
6	2

Does not satisfy condition (3).

Period Finding

The condition that that s divides N can make the problem easy classically.

If $N = 2^n$ this problem can be solved with $O(n)$ many queries to f.

Later we will look at a relaxation of the condition $s|N$ that makes the problem hard classically but for which the efficient quantum alg. still works.

Oracle

- $f(x) = f(x+s)$ for all $x \in \mathbb{Z}_N$
- $f(0), \ldots, f(s-1)$ are all distinct.

We will again work in the query model.

We assume access to an oracle O_f

$$O_f|x\rangle|z\rangle = |x\rangle|z \oplus f(x)\rangle$$

where $z \in \{0,1\}^m$ and we think of $f(x)$ written as an m bit string.

Fourier Transform over $\mathbb{Z}_N$

Fourier Transform

A Fourier transform can be defined over any finite abelian group.

We have already seen an example: $H^{\otimes n}$ is the Fourier transform over $\mathbb{Z}_2^n$.

For the period finding problem we will use the Fourier transform over $\mathbb{Z}_N$.

The Fourier transform is a change of basis that lets us see the pattern of a periodic function.

Fourier Transform

The Fourier transform is a basis change from the standard basis to a basis of multiplicative characters.

A multiplicative character $\chi : \mathbb{Z}_N \to \mathbb{C}$ has the property

$$\chi(x+y) = \chi(x)\chi(y) \quad \text{for all } x, y \in \mathbb{Z}_N$$

Question: What is a (non-zero) multiplicative character for $\mathbb{Z}_2$?

Characters

$$\chi(x+y) = \chi(x)\chi(y) \quad \text{for all} \quad x, y \in \mathbb{Z}_N$$

There are two non-zero characters over $\mathbb{Z}_2$.

$$\chi_0(0) = 1, \chi_0(1) = 1 \qquad \text{trivial character}$$

$$\chi_1(0) = 1, \chi_1(1) = -1 \qquad \text{sign character}$$

The truth tables of these characters form the columns of the Hadamard matrix.

Characters: Prop. 1

$$\chi(x+y) = \chi(x)\chi(y) \quad \text{for all } x, y \in \mathbb{Z}_N$$

We can deduce some general properties of a non-zero multiplicative character over $\mathbb{Z}_N$.

- $\chi(0) = 1$

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

Characters: Prop. 2

$$\chi(x+y) = \chi(x)\chi(y) \quad \text{for all } x, y \in \mathbb{Z}_N$$

We can deduce some general properties of a non-zero multiplicative character over $\mathbb{Z}_N$.

- $\chi(a) = \chi(1)^a$

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

A character is completely determined by $\chi(1)$.

Characters: Prop. 3

$$\chi(x+y) = \chi(x)\chi(y) \quad \text{for all} \quad x, y \in \mathbb{Z}_N$$

We can deduce some general properties of a non-zero multiplicative character over $\mathbb{Z}_N$.

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

$$\chi(1)^N = \chi(\overbrace{1 + \cdots + 1}^{N \text{ times}}) = \chi(0) = 1$$

$\chi(1)$ is an N^{th} root of unity.

All $\chi(a)$ are N^{th} roots of unity.

N characters

$$\chi(x+y) = \chi(x)\chi(y) \quad \text{for all } x, y \in \mathbb{Z}_N$$

A character is completely determined by $\chi(1)$, which is an N^{th} root of unity.

Let $\omega = e^{2\pi \mathrm{i}/N}$ be a primitive N^{th} root of unity.

We can define N characters $\chi_0, \cdots, \chi_{N-1}$, where

$$\chi_a(1) = \omega^a$$

N characters

$$\chi(x+y) = \chi(x)\chi(y) \quad \text{for all } x, y \in \mathbb{Z}_N$$

Let $\omega = e^{2\pi i/N}$ be a primitive N^{th} root of unity.

We can define N characters $\chi_0, \cdots, \chi_{N-1}$, where

$$\chi_a(1) = \omega^a$$

Check:

$$\chi_a(x+y) = \omega^{a\cdot(x+y)} = \omega^{ax} \cdot \omega^{ay} = \chi_a(x) \cdot \chi_a(y)$$

Exchanging arguments

A useful identity when manipulating characters is

$$\chi_a(x) = \omega^{a \cdot x} = \chi_x(a)$$

Exchanging the argument and subscript doesn't change the value.

$$\chi_a(x)\chi_b(x) = \chi_x(a)\chi_x(b) = \chi_x(a+b) = \chi_{a+b}(x)$$

The characters themselves form a group isomorphic to $\mathbb{Z}_N$ under pointwise multiplication.

Conjugating

We also make use of the fact that

$$\chi_a(x)\chi_a(-x) = \chi_a(0) = 1$$

Thus as $\chi_a(x)$ is a root of unity $\chi_a(-x) = \chi_a(x)^*$.

Key property

The following is a key property of characters we will use for Shor's algorithm and to show the FT is unitary.

Lemma: Let χ be a character of $\mathbb{Z}_N$. Let $r|N$ and suppose that $\chi(r) \neq 1$. Then

$$\sum_{\substack{a \in \mathbb{Z}_N \\ a=0 \bmod r}} \chi(a) = 0 \ .$$

Intuition

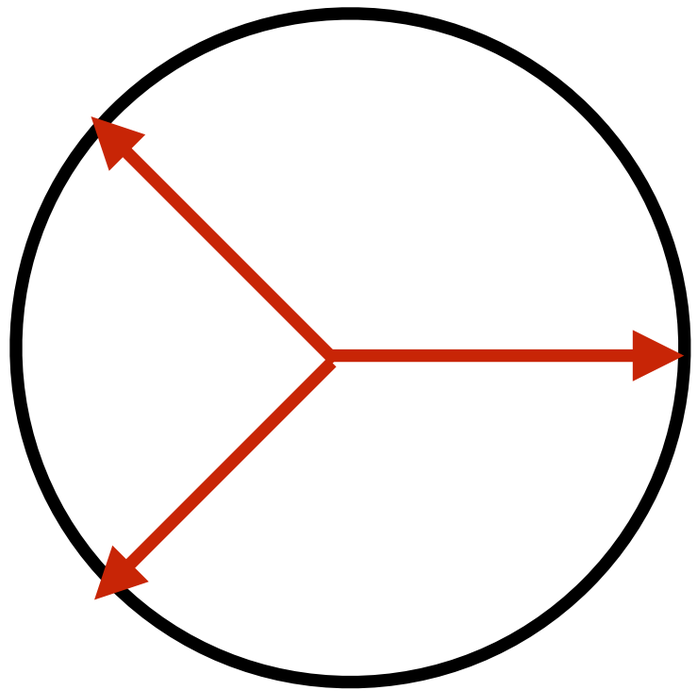
Lemma: Let χ be a character of $\mathbb{Z}_N$. Let $r|N$ and suppose that $\chi(r) \neq 1$. Then

$$\sum_{\substack{a \in \mathbb{Z}_N \\ a = 0 \bmod r}} \chi(a) = 0 \ .$$

Let $R = \{a : a = 0 \bmod r\}$.

The vectors $e^{2\pi i a/N}$ for $a \in R$ are balanced around the circle.

$N = 6, r = 2$



Lemma: Let χ be a character of $\mathbb{Z}_N$. Let $r|N$ and suppose that $\chi(r) \neq 1$. Then

$$\sum_{a \in R} \chi(a) = 0 \ .$$

$R = \{a : a = 0 \bmod r\}$

Proof:

$$r + R = \{r + a : a \in R\} = R$$

$$\chi(r) \sum_{a \in R} \chi(a) = \sum_{a \in r+R} \chi(a) = \sum_{a \in R} \chi(a)$$

As $\chi(r) \neq 1$ this gives the lemma.

Orthogonality

We next claim that the characters are orthogonal.

$$\sum_{x \in \mathbb{Z}_N} \chi_a(x)^* \chi_b(x) = \sum_{x \in \mathbb{Z}_N} \chi_x(-a)\chi_x(b)$$

$$= \sum_{x \in \mathbb{Z}_N} \chi_x(b-a)$$

$$= \sum_{x \in \mathbb{Z}_N} \chi_{b-a}(x)$$

By the previous lemma this is zero unless $b - a = 0$, in which case it is N.

Another basis

This means that $\frac{1}{\sqrt{N}}\chi_0, \ldots, \frac{1}{\sqrt{N}}\chi_{N-1}$,

viewed as N-dimensional vectors, form an orthonormal basis of $\mathbb{C}^N$.

The Fourier transform F_N over $\mathbb{Z}_N$ is the N-by-N change-of-basis matrix from the standard basis to the character basis above.

Given $f : \mathbb{Z}_N \to \mathbb{C}$ (viewed as a vector) $F_N f = \hat{f}$

$$f = \frac{1}{\sqrt{N}} \sum_{a \in \mathbb{Z}_N} \hat{f}(a)\chi_a$$

Change of basis

Let's step back and think about what a change of basis does.

Say we have a basis $\mathcal{A} = \{u_0, \ldots, u_{N-1}\}$ and a basis $\mathcal{B} = \{v_0, \ldots, v_{N-1}\}$.

Let $[z]_{\mathcal{A}} \in \mathbb{C}^N$ be the coordinates of $z \in \mathbb{C}^N$ in the basis $\mathcal{A}$, that is

$$z = ([z]_{\mathcal{A}})(0)u_0 + \cdots + ([z]_{\mathcal{A}})(N-1)u_{N-1}$$

Change of basis

Say we have a basis $\mathcal{A} = \{u_0, \ldots, u_{N-1}\}$ and a basis $\mathcal{B} = \{v_0, \ldots, v_{N-1}\}$.

The change of basis matrix $S_{\mathcal{A}\to\mathcal{B}}$ is such that for all z

$$S_{\mathcal{A}\to\mathcal{B}}[z]_{\mathcal{A}} = [z]_{\mathcal{B}}$$

This equation tells us what $S_{\mathcal{A}\to\mathcal{B}}$ is:

Since $[u_j]_{\mathcal{A}} = e_j$ we should have $S_{\mathcal{A}\to\mathcal{B}}\ e_j = [u_j]_{\mathcal{B}}$.

The LHS is the j^{th} column of $S_{\mathcal{A}\to\mathcal{B}}$.

Change of basis

Since $[u_j]_{\mathcal{A}} = e_j$ we should have $S_{\mathcal{A}\to\mathcal{B}}\, e_j = [u_j]_{\mathcal{B}}$.

$$S_{\mathcal{A}\to\mathcal{B}} = \begin{bmatrix} \vdots & \vdots & & \vdots \\ [u_0]_{\mathcal{B}} & [u_1]_{\mathcal{B}} & \cdots & [u_{N-1}]_{\mathcal{B}} \\ \vdots & \vdots & & \vdots \end{bmatrix}$$

Now how do we figure out what $[u_j]_{\mathcal{B}}$ is?

$$\begin{bmatrix} \vdots & \vdots & & \vdots \\ v_0 & v_1 & \cdots & v_{N-1} \\ \vdots & \vdots & & \vdots \end{bmatrix} x = u_j$$

Instantiation

In our case $\mathcal{A} = \{e_0, \ldots, e_{N-1}\}$ and $\mathcal{B} = \{\chi_0/\sqrt{N}, \ldots, \chi_{N-1}/\sqrt{N}\}$.

The j^{th} column of the Fourier transform is the solution to

$$\frac{1}{\sqrt{N}} \begin{bmatrix} \vdots & \vdots & & \vdots \\ \chi_0 & \chi_1 & \cdots & \chi_{N-1} \\ \vdots & \vdots & & \vdots \end{bmatrix} x = e_j$$

This is a unitary matrix!

Instantiation

The j^{th} column of the Fourier transform is

$$\frac{1}{\sqrt{N}} \begin{bmatrix} \cdots & \chi_0^* & \cdots \\ \cdots & \chi_1^* & \cdots \\ & \vdots & \\ \cdots & \chi_{N-1}^* & \cdots \end{bmatrix} e_j$$

That is, the Fourier transform over $\mathbb{Z}_N$ is the matrix

$$F_N = \frac{1}{\sqrt{N}} \begin{bmatrix} \cdots & \chi_0^* & \cdots \\ \cdots & \chi_1^* & \cdots \\ & \vdots & \\ \cdots & \chi_{N-1}^* & \cdots \end{bmatrix}$$

Fourier Transform

$$F_N = \frac{1}{\sqrt{N}} \begin{bmatrix} \cdots & \chi_0^* & \cdots \\ \cdots & \chi_1^* & \cdots \\ & \vdots & \\ \cdots & \chi_{N-1}^* & \cdots \end{bmatrix}$$

It will also be useful to think about this matrix by columns.

The j^{th} column is

$$\frac{1}{\sqrt{N}} \begin{bmatrix} \chi_0^*(j) \\ \chi_1^*(j) \\ \vdots \\ \chi_{N-1}^*(j) \end{bmatrix} = \frac{1}{\sqrt{N}} \begin{bmatrix} \chi_0(-j) \\ \chi_1(-j) \\ \vdots \\ \chi_{N-1}(-j) \end{bmatrix} = \frac{1}{\sqrt{N}} \chi_{-j}$$

Fourier Transform

Let's also look at it by entries:

For $j, k \in \{0, \ldots, N-1\}$ (indexing starts with 0)

$$F_N(j,k) = \frac{\chi_j(k)^*}{\sqrt{N}}$$
$$= \frac{\omega^{-j \cdot k}}{\sqrt{N}}$$

where $\omega = e^{2\pi i/N}$.

Inverse Fourier Transform

The change of basis matrix $S_{\mathcal{A}\to\mathcal{B}}$ is such that for all z

$$S_{\mathcal{A}\to\mathcal{B}}[z]_{\mathcal{A}} = [z]_{\mathcal{B}}$$

Thus $S_{\mathcal{B}\to\mathcal{A}} = S^{-1}_{\mathcal{A}\to\mathcal{B}}$.

The inverse Fourier transform changes from the character basis to the standard basis.

$$F_N^* = \frac{1}{\sqrt{N}} \begin{bmatrix} \vdots & \vdots & & \vdots \\ \chi_0 & \chi_1 & \cdots & \chi_{N-1} \\ \vdots & \vdots & & \vdots \end{bmatrix}$$

Example

$$F_6 = \frac{1}{\sqrt{6}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega^5 & \omega^4 & \omega^3 & \omega^2 & \omega \\ 1 & \omega^4 & \omega^2 & 1 & \omega^4 & \omega^2 \\ 1 & \omega^3 & 1 & \omega^3 & 1 & \omega^3 \\ 1 & \omega^2 & \omega^4 & 1 & \omega^2 & \omega^4 \\ 1 & \omega & \omega^2 & \omega^3 & \omega^4 & \omega^5 \end{bmatrix}.$$

$$F_6^* = \frac{1}{\sqrt{6}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega^1 & \omega^2 & \omega^3 & \omega^4 & \omega^5 \\ 1 & \omega^2 & \omega^4 & 1 & \omega^2 & \omega^4 \\ 1 & \omega^3 & 1 & \omega^3 & 1 & \omega^3 \\ 1 & \omega^4 & \omega^2 & 1 & \omega^4 & \omega^2 \\ 1 & \omega^5 & \omega^4 & \omega^3 & \omega^2 & \omega^1 \end{bmatrix}.$$

Notes

Our definition of F_N is what is usually called the discrete Fourier transform in the classical literature.

In quantum computing the "quantum Fourier transform" is usually defined to be the inverse F_N^* .

It does not make too much difference, but be aware of this when reading the literature.

Implementing the Fourier Transform

Outline

We will design an efficient quantum circuit to implement the Fourier transform over $\mathbb{Z}_N$.

We will assume that $N = 2^n$.

The circuit will have $O(n^2)$ one and two qubit gates.

We don't restrict our gate set.

Basis vectors

It suffices to show the circuit has the correct behavior on basis vectors $|x\rangle$ for $x \in \{0,1\}^n$.

$$F_N|x\rangle = \frac{1}{\sqrt{N}}\chi_{-x}$$

$$= \frac{1}{\sqrt{N}} \sum_{a\in[N]} \chi_a(x)^*|a\rangle$$

$$= \frac{1}{\sqrt{N}} \sum_{a\in[N]} \omega^{-ax}|a\rangle$$

Example

$$|x\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{a \in [N]} \omega^{-ax} |a\rangle$$

Consider the example $n = 4, N = 16$.

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

This is a product state! It equals

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

General Product

$$|x\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{a \in [N]} \omega^{-ax} |a\rangle$$

In general, the output is the tensor product

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

The coefficient of the state $|a_{n-1}a_{n-2}\cdots a_0\rangle$ is

$$\prod_{j:a_j=1} \omega^{-2^j x} = \omega^{-x\sum_j a_j 2^j} = \omega^{-ax}$$

as desired.

General Product

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

Let's simplify a general term

$$|z_k\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + \omega^{-2^k x}|1\rangle\right)$$

in this tensor product.

$$\begin{aligned}\omega^{-2^k x} &= \omega^{-2^k \sum_{j=0}^{n-1} x_j 2^j} \\ &= \prod_{j=0}^{n-1} \omega^{-x_j 2^{k+j}}\end{aligned}$$

$$|z_k\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + \omega^{-2^k x}|1\rangle\right)$$

Recalling that $\omega = e^{2\pi \mathrm{i}/2^n}$ we can continue to simplify

$$\omega^{-2^k x} = \prod_{j=0}^{n-1} \exp(2\pi \mathrm{i} x_j 2^{k+j}/2^n)$$

$$= \prod_{j=0}^{n-k-1} \exp(2\pi \mathrm{i} x_j 2^{k+j}/2^n)$$

Take away: $|z_k\rangle$ only depends on the $n-k$ least significant bits of x.

Building the circuit

$$|z_k\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + \prod_{j=0}^{n-k-1} \omega^{-x_j 2^{k+j}}|1\rangle\right)$$

This formula tells us how to build the circuit. How can we output $|z_k\rangle$?

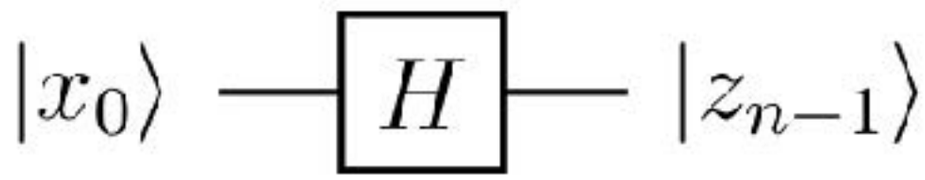
Let's work out the first couple cases:

$$\begin{aligned}
|z_{n-1}\rangle &= \frac{1}{\sqrt{2}}\left(|0\rangle + \omega^{-x_0 2^{n-1}}|1\rangle\right) \\
&= \frac{1}{\sqrt{2}}\left(|0\rangle + e^{-x_0 \pi \mathrm{i}}|1\rangle\right) \\
&= H|x_0\rangle
\end{aligned}$$

Building the circuit

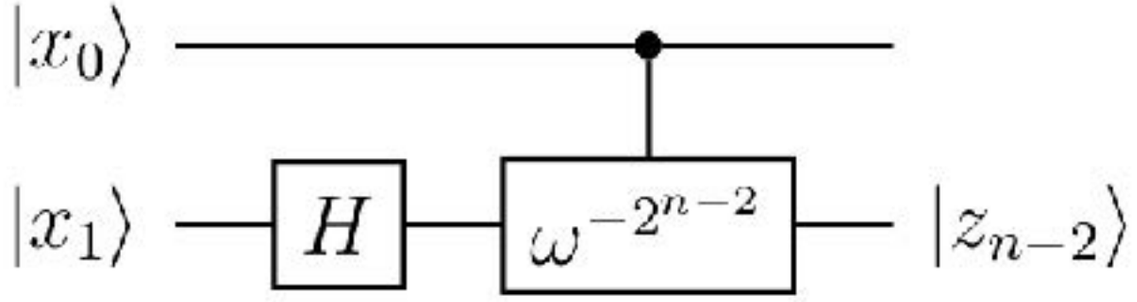
$$|z_k\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + \prod_{j=0}^{n-k-1} \omega^{-x_j 2^{k+j}} |1\rangle\right)$$

$$|z_{n-1}\rangle = H|x_0\rangle$$

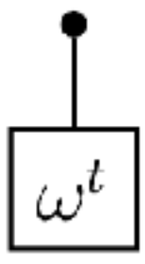


$$|z_k\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + \prod_{j=0}^{n-k-1} \omega^{-x_j 2^{k+j}} |1\rangle\right)$$

$$|z_{n-2}\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + (-1)^{x_0}\omega^{-x_1 2^{n-2}} |1\rangle\right)$$



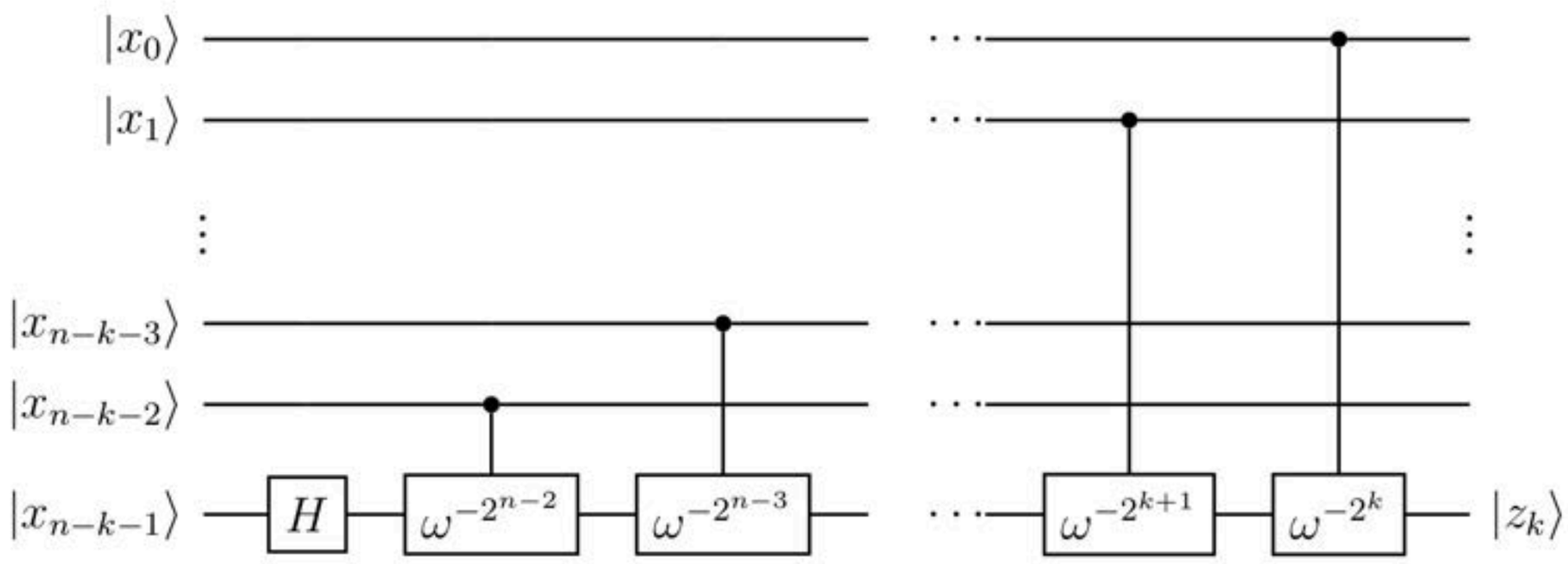
Where



$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \omega^t \end{bmatrix}$$

$$|z_k\rangle = \frac{1}{\sqrt{2}} \left(|0\rangle + \prod_{j=0}^{n-k-1} \omega^{-x_j 2^{k+j}} |1\rangle \right)$$

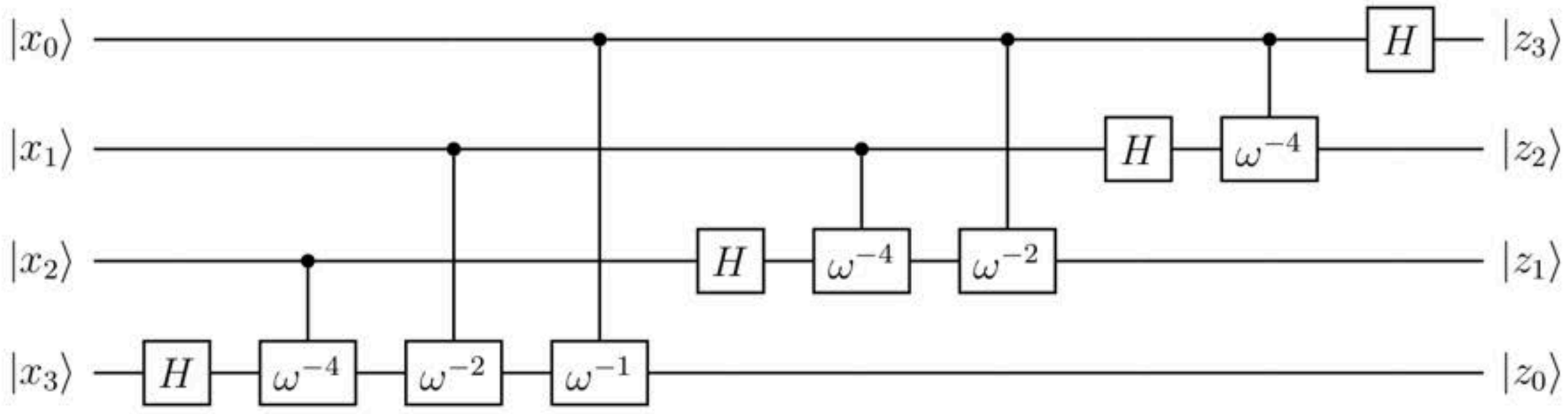
In general:



This uses $n-k$ gates.

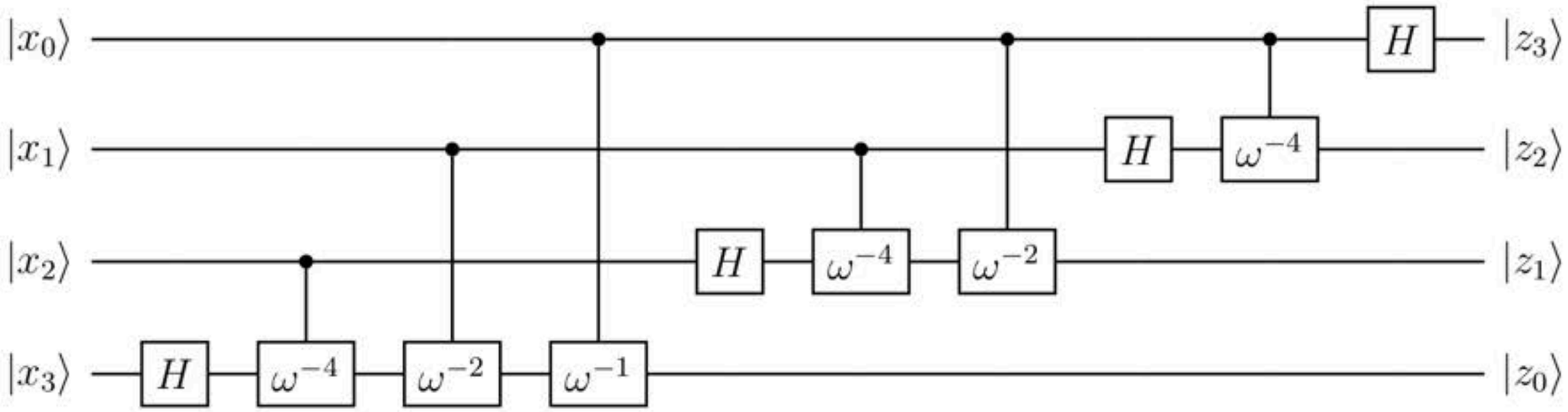
Building the circuit

We can create each $|z_k\rangle$ state sequentially, starting with $|z_0\rangle$.



This uses $\binom{n+1}{2}$ many gates.

But output qubits are in reverse order...



This uses $\binom{n+1}{2}$ many gates.

But output qubits are in reverse order...

We can fix that with $\lfloor n/2 \rfloor$ many swap gates.

The total number of gates is $O(n^2)$.

Further notes

• Using techniques described in first lecture we can approximate the circuit by one using our preferred universal gate set.

• Approximate Fourier Transform

We can approximate the Fourier transform to precision ε using only $n(\log(n) + \log(1/\varepsilon))$ gates [Coppersmith].

• We can also efficiently approximately implement F_N for any N, not just powers of two [Hales+Hallgren].

Phase Estimation

Phase Estimation

Let U be a unitary and $|v\rangle$ an eigenvector of U.

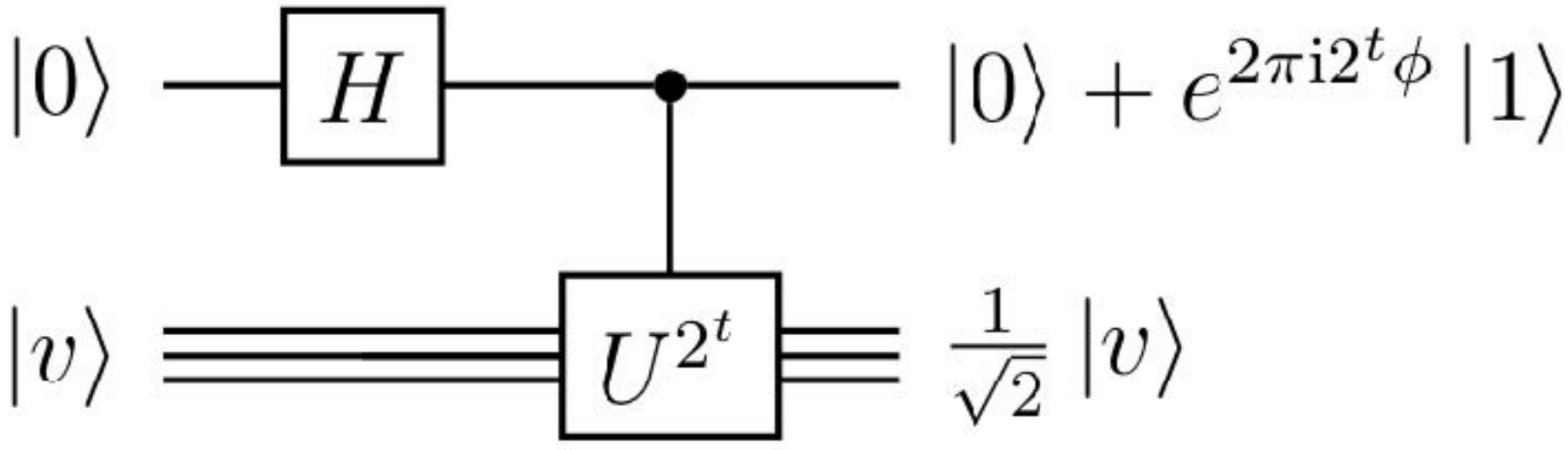
$$U|v\rangle = e^{2\pi \mathrm{i}\phi}|v\rangle$$

Let us assume that $\phi \in [0, 1)$ can be written exactly with n bits $\phi = 0.\phi_1 \dots \phi_n$.

Assumption: We are able to apply controlled U^m gates.

Goal: Output ϕ.

Building block

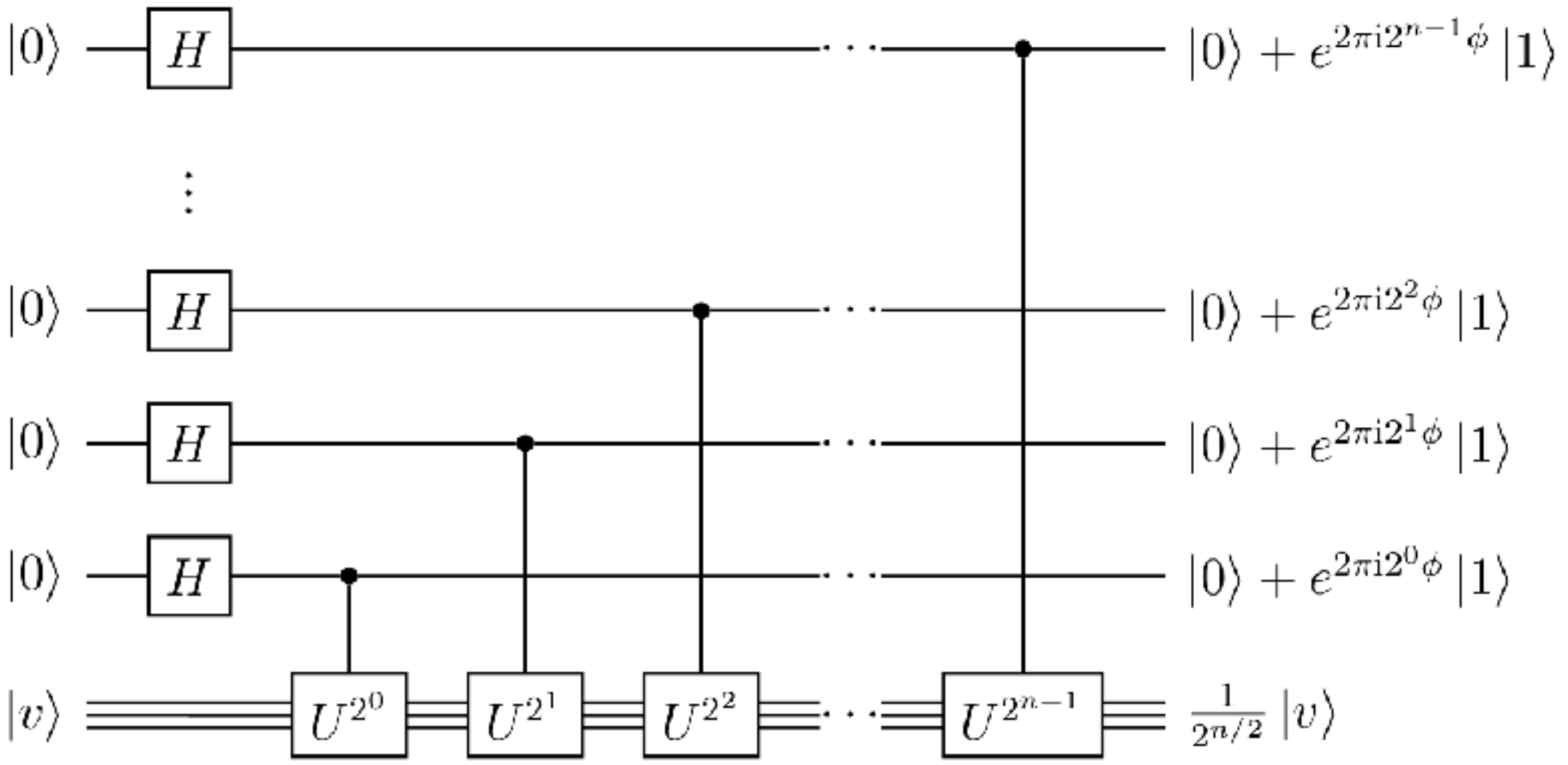


The controlled U^{2^t} maps

$$|0\rangle|v\rangle \mapsto |0\rangle|v\rangle$$

$$|1\rangle|v\rangle \mapsto e^{2\pi \mathrm{i} 2^t \phi}|1\rangle|v\rangle$$

Put these blocks together in the circuit



The final state is $|\psi\rangle|v\rangle$ where

$$|\psi\rangle = 2^{-n/2}(|0\rangle + e^{2\pi i 2^{n-1}\phi}|1\rangle) \otimes (|0\rangle + e^{2\pi i 2^{n-2}\phi}|1\rangle) \otimes \cdots \otimes (|0\rangle + e^{2\pi i 2^{0}\phi}|1\rangle)$$

$$= 2^{-n/2} \sum_{x=0}^{2^{n-1}} e^{2\pi i x\phi}|\mathrm{bin}(x)\rangle$$

We now have the state (labelling kets by $x \in \{0, \ldots, 2^{n-1}\}$)

$$|\psi\rangle = \sum_{x=0}^{2^{n-1}} e^{2\pi \mathrm{i} x\phi} |x\rangle$$

$$= \sum_{x=0}^{2^{n-1}} \omega^{x(2^n\phi)} |x\rangle \qquad \omega = e^{2\pi \mathrm{i}/2^n}$$

$$= F_{2n}^* |2^n\phi\rangle$$

Thus if we apply F_{2^n} we get the state $|2^n\phi\rangle$ with certainty.

If ϕ cannot exactly be written with n bits, the same procedure will with good probability output an n bit approximation of ϕ.