7. Quantum algorithms (part 1)

(a) Quantum query complexity: (KLM 9.2*, 6.2*)
black box model, phase kick back

(d) Deutsch-Jozsa algorithm
(NC 1.4.2-1.4.5, KLM 6.3-6.4, M 2.2)

(e) Quantum fourier transform (I)
(NC 5.1, M 3.5, KLM p110-117)

(f) Simon's algorithm (M 2.5, KLM 6.5)

(g) Shor's factoring algorithm

(M 3.1-3.4,3.7-3.10, NC 5.3, 5.4.1-5.4.2,
KLM 7.1.2-7.1.3, 7.3.1-7.3.2, 7.3.4, 7.4)

(h) Hidden subgroup framework (NC 5.4.3, KLM 7.5)

Simon's algorithm

Simon's problem:

Given: a black box for a function $f: \{0,1\}^n \longrightarrow \{0,1\}^n$.

Promise (partial information about f):

$\exists\, s \in \{0,1\}^n$, $f(x) = f(y)$ iff $x=y$ or $x=y \oplus s$

Problem: determine s.

Simon's problem:

Given: a black box for a function $f: \{0,1\}^n \longrightarrow \{0,1\}^n$.

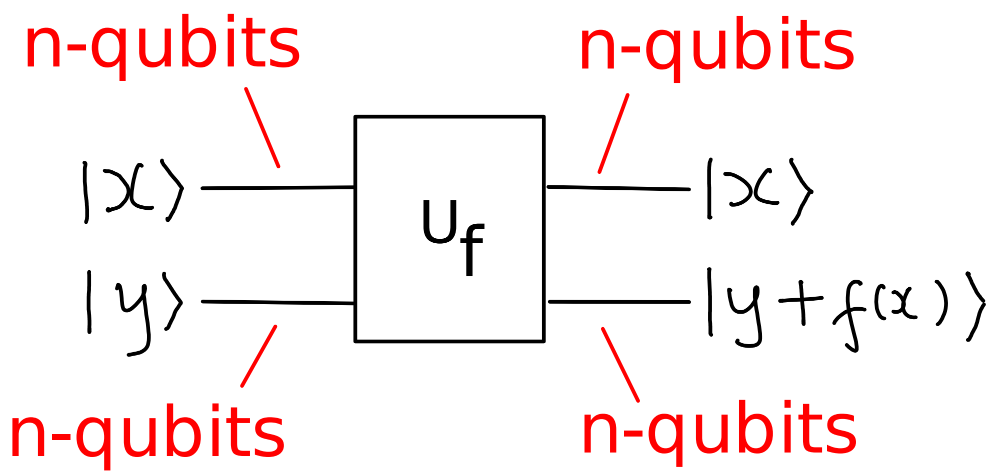
Promise (partial information about f):

$$\exists s \in \{0,1\}^n, \quad f(x) = f(y) \text{ iff } x=y \text{ or } x=y \oplus s$$

Problem: determine s.

Note: f is NOT boolean.

Quantum blackbox:



Simon's problem:

Given: a black box for a function $f: \{0,1\}^n \longrightarrow \{0,1\}^n$.

Promise (partial information about f):

$\exists s \in \{0,1\}^n$, $f(x) = f(y)$ iff $x=y$ or $x=y \oplus s$

Problem: determine s.

Example:

x	f(x)
000	011
001	101
010	000
011	010
100	101
101	011
110	010
111	000

Simon's problem:

Given: a black box for a function $f:\{0,1\}^n \longrightarrow \{0,1\}^n$.

Promise (partial information about f):

$\exists s \in \{0,1\}^n$, $f(x) = f(y)$ iff $x=y$ or $x=y \oplus s$

Problem: determine s.

Example:

x	f(x)
000	011
001	101
010	000
011	010
100	101
101	011
110	010
111	000

Question:
What is s in the example?

Idea: find (x,y) with f(x)=f(y)
take s = y-x

(a) s = 011
(b) s = 101
(c) s = 000
(d) s = 010

Simon's problem:

Given: a black box for a function $f: \{0,1\}^n \longrightarrow \{0,1\}^n$.

Promise (partial information about f):

$$\exists\, s \in \{0,1\}^n, \quad f(x) = f(y) \text{ iff } x=y \text{ or } x=y \oplus s$$

Problem: determine s.

Exponential quantum speed-up (in query complexity)
in BQP compared to BPP for a specific problem.

Simon's problem:

Given: a black box for a function $f: \{0,1\}^n \longrightarrow \{0,1\}^n$.

Promise (partial information about f):

$\exists\, s \in \{0,1\}^n$, $f(x) = f(y)$ iff $x=y$ or $x=y \oplus s$

Problem: determine s.

Exponential quantum speed-up (in query complexity) in BQP compared to BPP for a specific problem.

Classical: for any $2^{0.49n}$ queries, algorithm fails wp at least 99% on at least 99% of the functions.

(for large n, say, above 400)

Simon's problem:

Given: a black box for a function $f: \{0,1\}^n \longrightarrow \{0,1\}^n$.

Promise (partial information about f):

$$\exists s \in \{0,1\}^n, \quad f(x) = f(y) \text{ iff } x=y \text{ or } x=y \oplus s$$

Problem: determine s.

Exponential quantum speed-up (in query complexity) in BQP compared to BPP for a specific problem.

Classical: for any $2^{0.49n}$ queries, algorithm fails wp at least 99% on at least 99% of the functions.

Quantum: n+3 queries are sufficient to obtain s with probability at least 15/16.

Simon's problem:

Given: a black box for a function $f:\{0,1\}^n \longrightarrow \{0,1\}^n$.

Promise (partial information about f):

$$\exists\, s \in \{0,1\}^n, \quad f(x) = f(y) \text{ iff } x=y \text{ or } x=y \oplus s$$

Problem: determine s.

Exponential quantum speed-up (in query complexity) in BQP compared to BPP for a specific problem.

$\boxed{\text{Classical: for any } 2^{0.49n} \text{ queries, algorithm fails wp at least 99\% on at least 99\% of the functions.}}$

Quantum: n+3 queries are sufficient to obtain s with probability at least 15/16.

$\exists\, s \in \{0,1\}$,   f(x) = f(y) iff x=y or x=y $\oplus$ s

To specify f :

1. Specify any $s \neq 00...0$.

$\exists s \in \{0,1\}$, f(x) = f(y) iff x=y or $x = y \oplus s$

To specify f :

1. Specify any $s \neq 00...0$.
2. s pairs up the 2^n inputs into 2^{n-1} pairs.

Example:
s = 101

x	f(x)
000	011
001	101
010	000
011	010
100	101
101	011
110	010
111	000

There are 4 pairs
(000,101), (001,100),
(010,111), (011,110).

$\exists s \in \{0,1\}$, $f(x) = f(y)$ iff $x=y$ or $x=y \oplus s$

To specify f :

1. Specify any $s \neq 00...0$.

2. s pairs up the 2^n inputs into 2^{n-1} pairs.

Example:
s = 101

x	f(x)
000	011
001	101
010	000
011	010
100	101
101	011
110	010
111	000

There are 4 pairs
(000,101), (001,100),
(010,111), (011,110).

assign 011 to (000,101)
assign 101 to (001,100)
etc

3.

To get a valid f: give each pair a distinct function value.
For each s, there are $2^n (2^n-1) \cdots (2^n - 2^{n-1}+1)$ valid f's.

How does a classical algorithm fail?

For an algorithm with k distinct queries $x_1, \ldots, x_k$

How does a classical algorithm fail?

For an algorithm with k distinct queries $x_1, \ldots, x_k$

if (a) $f(x_1), f(x_2), \ldots, f(x_k)$ distinct and (b) $\binom{k}{2} \ll 2^n - 1$

then algorithm fails.

How does a classical algorithm fail?

For an algorithm with k distinct queries $x_1, \ldots, x_k$

if (a) $f(x_1), f(x_2), \ldots f(x_k)$ distinct and (b) $\binom{k}{2} \ll 2^n - 1$

then algorithm fails.

(a) means no luck getting a pair (x,y) with f(x) = f(y)

All we know is $\forall\ 1 \leq i, j \leq k,\ s \neq x_i - x_j$.

This eliminates $\binom{k}{2}$ possibilities for s, but there are $2^n - 1$ possibilities. (b) means we know little about s.

Qn: given distinct $x_1, \ldots, x_k$ how many s's does NOT lead to distinct $f(x_1), f(x_2), \ldots, f(x_k)$?

Qn: given distinct $x_1, \ldots, x_k$ how many s's does NOT lead to distinct $f(x_1), f(x_2), \ldots f(x_k)$?

Ans: this requires some $f(x_i) = f(x_j)$, or $s = x_i - x_j$ for some i,j. There are only $\binom{k}{2}$ $x_i - x_j$'s (which are not necessarily distinct).

So at most $\binom{k}{2}$ such possible s's.

Qn: given distinct $x_1, \ldots, x_k$ how many s's does NOT lead to distinct $f(x_1), f(x_2), \ldots f(x_k)$?

Ans: this requires some $f(x_i) = f(x_j)$, or $s = x_i - x_j$ for some i,j. There are only $\binom{k}{2}$ $x_i - x_j$'s (which are not necessarily distinct).

So at most $\binom{k}{2}$ such possible s's.

So, the fraction of s (and also the fraction of f) withOUT distinct $f(x_1), f(x_2), \ldots f(x_k)$ is at most $\dfrac{\binom{k}{2}}{2^n - 1} \lesssim \dfrac{k^2}{2^n}$.

Qn: given distinct $x_1, \ldots, x_k$ how many s's does NOT lead to distinct $f(x_1), f(x_2), \ldots f(x_k)$?

Ans: this requires some $f(x_i) = f(x_j)$, or $s = x_i - x_j$ for some i,j. There are only $\binom{k}{2}$ $x_i - x_j$'s (which are not necessarily distinct).

So at most $\binom{k}{2}$ such possible s's.

So, the fraction of s (and also the fraction of f) withOUT distinct $f(x_1), f(x_2), \ldots f(x_k)$ is at most $\dfrac{\binom{k}{2}}{2^n - 1} \lesssim \dfrac{k^2}{2^n}$.

If $k < 2^{0.49n}$, $\dfrac{k^2}{2^n} < 2^{-0.02n} < 1\%$ for large n (say, $n \geq 400$).

For an algorithm with k distinct queries $x_1, \ldots, x_k$

if (a) $f(x_1), f(x_2), \ldots, f(x_k)$ distinct and (b) $\binom{k}{2} \ll 2^n - 1$

then algorithm fails.

holds for at least 99% of the functions

99% of the s's are not eliminated

For an algorithm with k distinct queries $x_1, \ldots, x_k$

if (a) $f(x_1), f(x_2), \ldots, f(x_k)$ distinct and (b) $\binom{k}{2} \ll 2^n - 1$

then algorithm fails.

holds for at least 99% of the functions

99% of the s's are not eliminated

$\therefore$ Classical: for any $2^{0.49n}$ queries, algorithm fails wp at least 99% on at least 99% of the functions.

Aside: argument similar to the birthday paradox implies

$\approx \sqrt{2^n} \approx 2^{\frac{n}{2}}$ queries are sufficient.

Simon's problem:

Given: a black box for a function $f:\{0,1\}^n \longrightarrow \{0,1\}^n$.

Promise (partial information about f):

$\exists\, s \in \{0,1\}^n, \quad f(x) = f(y) \text{ iff } x=y \text{ or } x=y \oplus s$

Problem: determine s.

Exponential quantum speed-up (in query complexity) in BQP compared to BPP for a specific problem.

Classical: for any $2^{0.49n}$ queries, algorithm fails wp at least 99% on at least 99% of the functions.

$\boxed{\text{Quantum: n+3 queries are sufficient to obtain s with probability at least 15/16.}}$

Simon's algorithm:

1. Prepare a superposition of inputs

$$H^{\otimes n} |0\rangle = \frac{1}{\sqrt{2^n}} \sum_x |x\rangle$$

Simon's algorithm:

1. Prepare a superposition of inputs

$$H^{\otimes n} |0\rangle = \frac{1}{\sqrt{2^n}} \sum_x |x\rangle$$

2. Query with out phase kick-back:

$$U_f \left(\frac{1}{\sqrt{2^n}} \sum_x |x\rangle \underbrace{|0\rangle}_{2^n\text{-dim}} \right) = \frac{1}{\sqrt{2^n}} \sum_x |x\rangle |f(x)\rangle$$

Simon's algorithm:

1. Prepare a superposition of inputs

$$H^{\otimes n}|0\rangle = \frac{1}{\sqrt{2^n}}\sum_x |x\rangle$$

2. Query with out phase kick-back:

$$U_f\left(\frac{1}{\sqrt{2^n}}\sum_x |x\rangle \underbrace{|0\rangle}_{2^n\text{-dim}}\right) = \frac{1}{\sqrt{2^n}}\sum_x |x\rangle|f(x)\rangle$$

3. Measure 2nd register !

For each pair $(x, x\oplus s)$, we put one of them in a set T.

Each x in T has a unique f(x). Each of these f(x) occurs as the measurement outcome with prob $1/2^{n-1}$.

Simon's algorithm:

1. Prepare a superposition of inputs

$$H^{\otimes n} |0\rangle = \frac{1}{\sqrt{2^n}} \sum_x |x\rangle$$

2. Query with out phase kick-back:

$$U_f \left(\frac{1}{\sqrt{2^n}} \sum_x |x\rangle \underbrace{|0\rangle}_{2^n\text{-dim}} \right) = \frac{1}{\sqrt{2^n}} \sum_x |x\rangle |f(x)\rangle$$

3. Measure 2nd register !

For each pair $(x, x \oplus s)$, we put one of them in a set T.

Each x in T has a unique f(x). Each of these f(x) occurs as the measurement outcome with prob $1/2^{n-1}$.

Corresponding postmeasurement state:

$$\frac{1}{\sqrt{2}} \left(|x\rangle + |x \oplus s\rangle \right) |f(x)\rangle$$

Example:

x	f(x)
000	011
001	101
010	000
011	010
100	101
101	011
110	010
111	000

s=101

The pairs (x, x$\oplus$s) are:

(000,101), (001,100), (010,111), (011,110)

T={000,001,010,011}

Measure the 2nd register of $\frac{1}{\sqrt{2^n}}\sum_x |x\rangle |f(x)\rangle$

in the computational basis.

Question: what is the prob of outcome 010

& what is the postmeas state (pms)?

(a) prob = 1/8, pms = $|010\rangle|000\rangle$

(b) prob = 1/4, pms = $\frac{1}{\sqrt{2}}\left(|011\rangle + |110\rangle\right)|010\rangle$

Example:

x	f(x)
000	011
001	101
010	000
011	010
100	101
101	011
110	010
111	000

s=101

The pairs (x, x ⊕ s) are:
(000,101), (001,100), (010,111), (011,110)

T={000,001,010,011}

Measure the 2nd register of $\frac{1}{\sqrt{2^n}} \sum_x |x\rangle |f(x)\rangle$

in the computational basis.

Outcome = 010 wp 1/4.

Postmeasurement state:

$$\frac{1}{\sqrt{2}}\Big(\underbrace{|011\rangle}_{x} + \underbrace{|110\rangle}_{x \oplus s}\Big)\underbrace{|010\rangle}_{f(x)}$$

Example:

x	f(x)
000	011
001	101
010	000
011	010
100	101
101	011
110	010
111	000

s=101

The pairs (x, x$\oplus$s) are:
(000,101), (001,100), (010,111), (011,110)

T={000,001,010,011}

Measure the 2nd register of $\frac{1}{\sqrt{2^n}}\sum_x |x\rangle|f(x)\rangle$
in the computational basis.

Outcome = 010 wp 1/4.

Postmeasurement state:

$$\frac{1}{\sqrt{2}}\Big(\underbrace{|011\rangle}_{x}+\underbrace{|110\rangle}_{x\oplus s}\Big)\underbrace{|010\rangle}_{f(x)}$$

How to access s from

$$\frac{1}{\sqrt{2}}\left(|x\rangle+|x\oplus s\rangle\right)|f(x)\rangle \; ?$$

How NOT to access s from $\frac{1}{\sqrt{2}}\left(|x\rangle + |x \oplus s\rangle\right)|f(x)\rangle$?

How NOT to access s from $\frac{1}{\sqrt{2}}\left(|x\rangle + |x \oplus s\rangle\right)|f(x)\rangle$?

Measure the first register in the computational basis:
gets x or x $\oplus$ s, both are random n-bit strings.

How NOT to access s from $\frac{1}{\sqrt{2}}\left(|x\rangle + |x \oplus s\rangle\right)|f(x)\rangle$?

Measure the first register in the computational basis:
gets x or x $\oplus$ s, both are random n-bit strings.

If we have many copies of $\frac{1}{\sqrt{2}}\left(|x\rangle + |x \oplus s\rangle\right)|f(x)\rangle$
measuring the first register gives x half of the time,
& x $\oplus$ s half of the time, together we can deduce s.
But we cannot clone to make many copies ...

How NOT to access s from $\frac{1}{\sqrt{2}}\left(|x\rangle + |x \oplus s\rangle\right)|f(x)\rangle$?

Measure the first register in the computational basis: gets x or x $\oplus$ s, both are random n-bit strings.

If we have many copies of $\frac{1}{\sqrt{2}}\left(|x\rangle + |x \oplus s\rangle\right)|f(x)\rangle$

measuring the first register gives x half of the time, & x $\oplus$ s half of the time, together we can deduce s.

But we cannot clone to make many copies ...

If we repeat steps 1-3, we get $\frac{1}{\sqrt{2}}\left(|x'\rangle + |x' \oplus s\rangle\right)|f(x')\rangle$

for $x' \neq x$ with very high probability, so, doesn't help.

How to access s from $\frac{1}{\sqrt{2}}\left(|x\rangle + |x \oplus s\rangle\right)|f(x)\rangle$?

How to access s from $\frac{1}{\sqrt{2}}\left(|x\rangle + |x \oplus s\rangle\right)|f(x)\rangle$?

For quantum algorithms ...

when in doubt, fourier transform !

4. Fourier transform the first register:

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

Recall from last lecture:

$$|x\rangle \xrightarrow{H^{\otimes n}} \frac{1}{\sqrt{2^n}}\sum_y |y\rangle (-1)^{x \cdot y}$$

4. Fourier transform the first register:

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

$$= \frac{1}{\sqrt{2^n}}\sum_y (-1)^{x\cdot y}\,|y\rangle + \frac{1}{\sqrt{2^n}}\sum_y (-1)^{(x\oplus s)\cdot y}\,|y\rangle$$

Recall from last lecture:

$$|x\rangle \xrightarrow{H^{\otimes n}} \frac{1}{\sqrt{2^n}}\sum_y |y\rangle (-1)^{x\cdot y}$$

4. Fourier transform the first register:

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

$$= \frac{1}{\sqrt{2^n}}\sum_y (-1)^{x\cdot y}\,|y\rangle + \frac{1}{\sqrt{2^n}}\sum_y (-1)^{(x\oplus s)\cdot y}\,|y\rangle$$

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

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

4. Fourier transform the first register:

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

$$= \frac{1}{\sqrt{2^n}} \sum_y (-1)^{x \cdot y} |y\rangle + \frac{1}{\sqrt{2^n}} \sum_y (-1)^{(x \oplus s)\cdot y} |y\rangle$$

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

$$= \frac{1}{\sqrt{2^n}} \sum_y (-1)^{x \cdot y} \underbrace{\left(1 + (-1)^{s \cdot y}\right)}_{\substack{0 \text{ if } s \cdot y = 1 \\ 2 \text{ '' '' } 0}} |y\rangle$$

5. Measuring gives a random y orthogonal to s !

Example:

x	f(x)
000	011
001	101
010	000
011	010
100	101
101	011
110	010
111	000

s=101

Postmeasurement state on the 1st register:

$$\frac{1}{\sqrt{2}}\left(\underset{x}{|011\rangle} + \underset{x\oplus s}{|110\rangle}\right)$$

$$\Big\downarrow H^{\otimes 3}$$

$$\frac{1}{4}\Big[\big(|0\rangle+|1\rangle\big)\big(|0\rangle-|1\rangle\big)\big(|0\rangle-|1\rangle\big) + \big(|0\rangle-|1\rangle\big)\big(|0\rangle-|1\rangle\big)\big(|0\rangle+|1\rangle\big)\Big]$$

Example:

x	f(x)
000	011
001	101
010	000
011	010
100	101
101	011
110	010
111	000

s=101

Postmeasurement state on the 1st register:

$$\frac{1}{\sqrt{2}}\left(\underset{x}{|011\rangle} + \underset{x\oplus s}{|110\rangle}\right)$$

$$\Big\downarrow H^{\otimes 3}$$

$$\frac{1}{4}\left[\begin{array}{l}(|0\rangle+|1\rangle)(|0\rangle-|1\rangle)(|0\rangle-|1\rangle)\\ +(|0\rangle-|1\rangle)(|0\rangle-|1\rangle)(|0\rangle+|1\rangle)\end{array}\right]$$

$$= \frac{1}{4}\left(|000\rangle + |010\rangle + |101\rangle + |111\rangle\right)$$

all orthogonal to s = 101 (inner product of binary strings)

no $|001\rangle, |011\rangle, |100\rangle, |110\rangle$

measuring gives a random y in 000, 010, 101, 111, orthogonal to s = 101

6. To reconstruct s, repeat steps 1-5 n+t times.

This gives $y_1, y_2, \ldots, y_{t+n} \in \{0,1\}^n$,

s.t. $\forall i,\ y_i \cdot s = 0$.

n+t linear equations in n variables

6. To reconstruct s, repeat steps 1-5 n+t times.

This gives $y_1, y_2, \ldots, y_{t+n} \in \{0,1\}^n$,

s.t. $\forall i,\ y_i \cdot s = 0$.

n+t linear equations in n variables

We need n-1 such equations that are linearly independent to find s (the 2 solutions for these n-1 equations are s and 0). What t suffices?

6. To reconstruct s, repeat steps 1-5 n+t times.

This gives $y_1, y_2, \ldots, y_{t+n} \in \{0,1\}^n$,

s.t. $\forall i,\ y_i \cdot s = 0$.

n+t linear equations in n variables

We need n-1 such equations that are linearly independent to find s (the 2 solutions for these n-1 equations are s and 0). What t suffices?

KLM appendix 3:
n+3 queries give prob of success at least 2/3.

6. To reconstruct s, repeat steps 1-5 n+t times.

This gives $y_1, y_2, \ldots, y_{t+n} \in \{0,1\}^n$,

s.t. $\forall i,\ y_i \cdot s = 0$.

n+t linear equations in n variables

We need n-1 such equations that are linearly independent to find s (the 2 solutions for these n-1 equations are s and 0). What t suffices?

KLM appendix 3:
n+3 queries give prob of success at least 2/3.

Mermin appendix G:
n+t queries give prob of success at least $1-2^{-(t+1)}$!

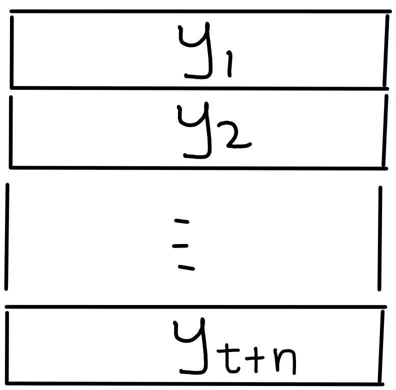
Mermin's proof: $y_1, y_2, \ldots, y_{t+n}$

come from an (n-1)-dim space S orthogonal to s.

Mermin's proof: $y_1, y_2, \ldots, y_{t+n}$

come from an (n-1)-dim space S orthogonal to s.

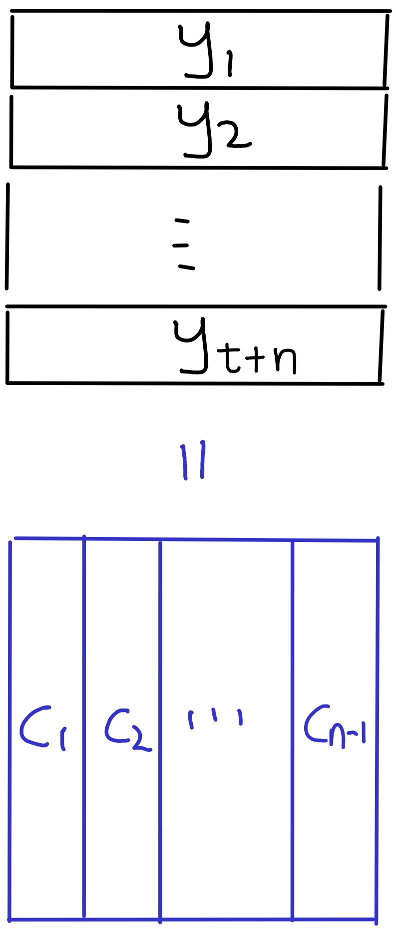
Take any basis for S. Represent each y_i (random) as an (n-1)-bit string in a row.



Mermin's proof: $y_1, y_2, \ldots, y_{t+n}$

come from an (n-1)-dim space S orthogonal to s.

Take any basis for S. Represent each y_i (random) as an (n-1)-bit string in a row.

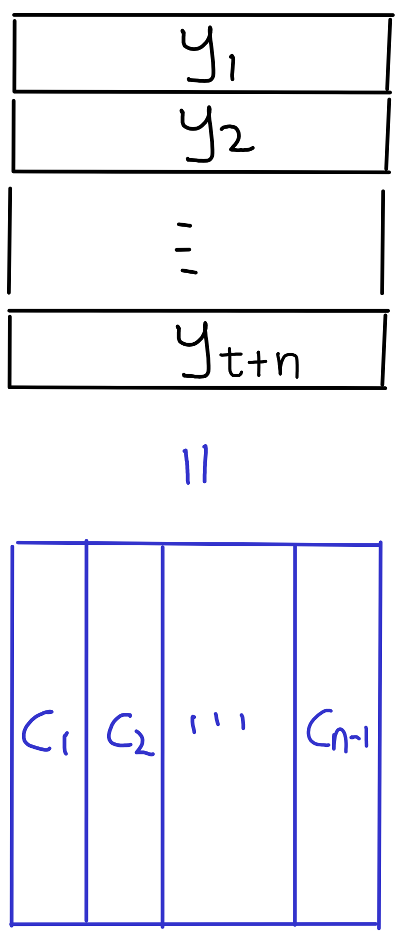


There are n-1 linearly independent rows
iff
the n-1 columns are linearly indep.
(row rank = column rank)

Mermin's proof: $y_1, y_2, \ldots, y_{t+n}$

come from an (n-1)-dim space S orthogonal to s.

Take any basis for S. Represent each y_i (random) as an (n-1)-bit string in a row.



There are n-1 linearly independent rows iff
the n-1 columns are linearly indep.
(row rank = column rank)

This happens with prob:

$$\Big(1-\underbrace{\frac{1}{2^{n+t}}}_{\Pr(c_1=0)}\Big)\Big(1-\underbrace{\frac{2}{2^{n+t}}}_{\Pr(c_2\in \mathrm{span}\{c_1\})}\Big)\cdots\Big(1-\underbrace{\frac{2^{n-2}}{2^{n+t}}}_{\Pr(c_{n-1}\in \mathrm{span}\{c_1, c_2, \ldots, c_{n-2}\})}\Big)$$

Can prove an arithmetic result (by induction):

if $0 \leqslant a+b+c \ldots \leqslant 1$,

then $(1-a)(1-b)(1-c)\ldots. \geqslant 1-(a+b+c\ldots)$

Can prove an arithmetic result (by induction):

if $0 \leqslant a+b+c \ldots \leqslant 1$,

then $(1-a)(1-b)(1-c)\ldots. \geqslant 1-(a+b+c\ldots)$

So, prob that there are n-1 linearly independent rows

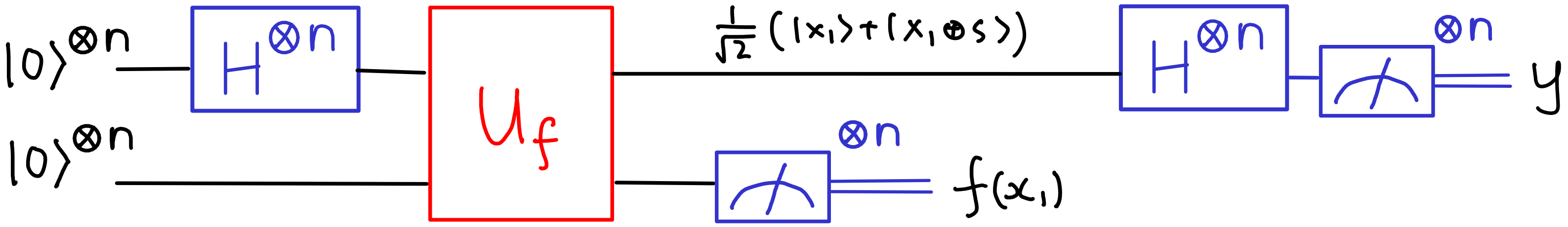
$$\left(1-\frac{1}{2^{n+t}}\right)\left(1-\frac{2}{2^{n+t}}\right) \cdots \left(1-\frac{2^{n-2}}{2^{n+t}}\right)$$

$$\geqslant 1-\left(\frac{1}{2^{n+t}}+\frac{1}{2^{n+t-1}}+\cdots+\frac{1}{2^{t+2}}\right)$$

$$\geqslant 1-\frac{1}{2^{t+1}}$$

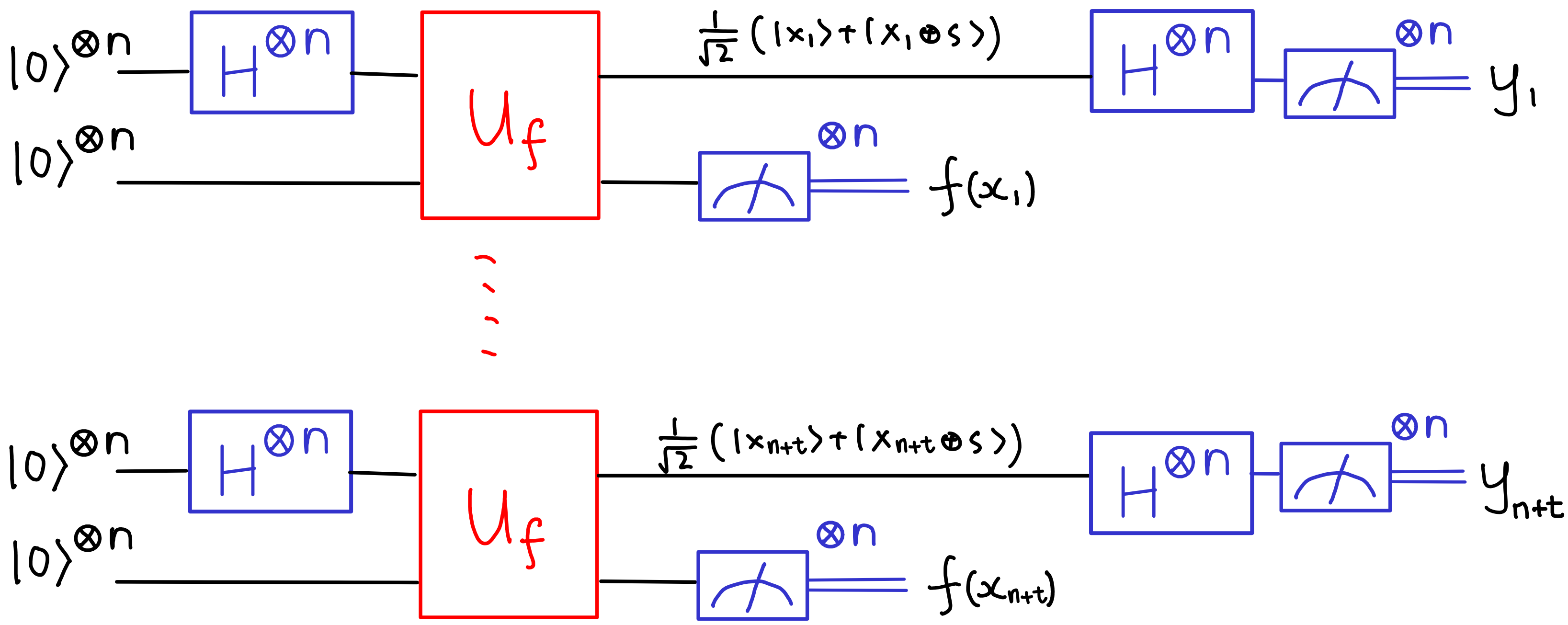
$\square$

Simon's algorithm altogether:



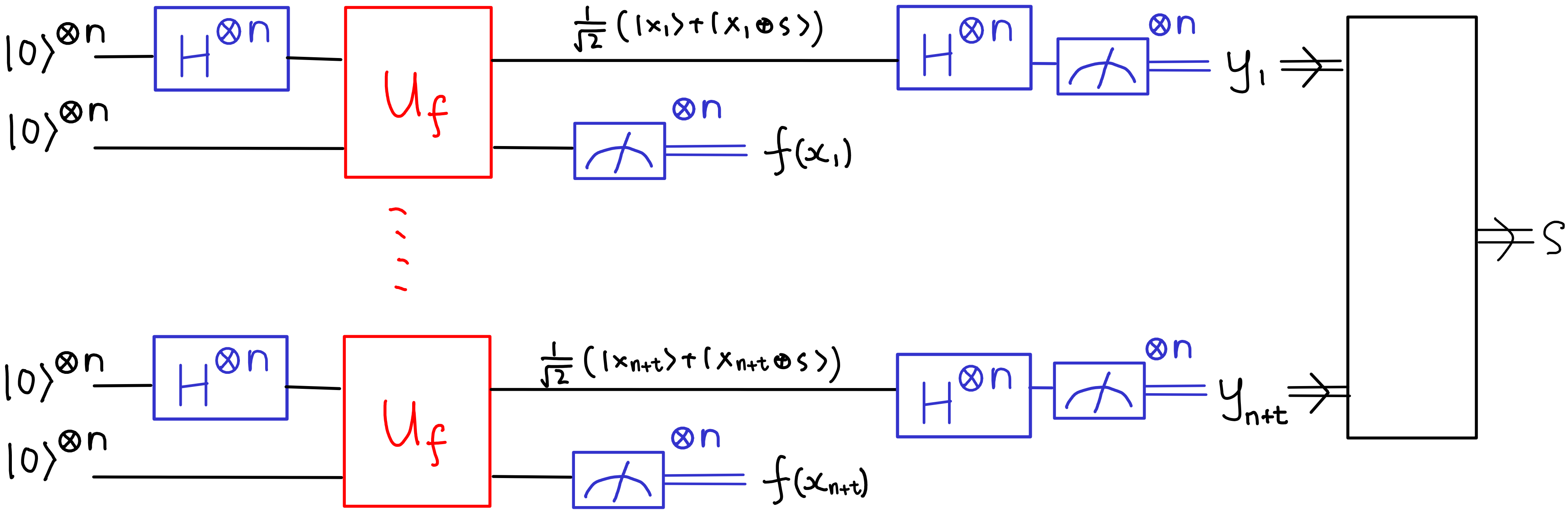
A quantum subroutine (steps 1-5) using 1 query to obtain one random y = y1 orthogonal to s.

Simon's algorithm altogether:



Repeat quantum subroutine n+t times to obtain n+t random $y_1, \ldots, y_{n+t}$ orthogonal to s.

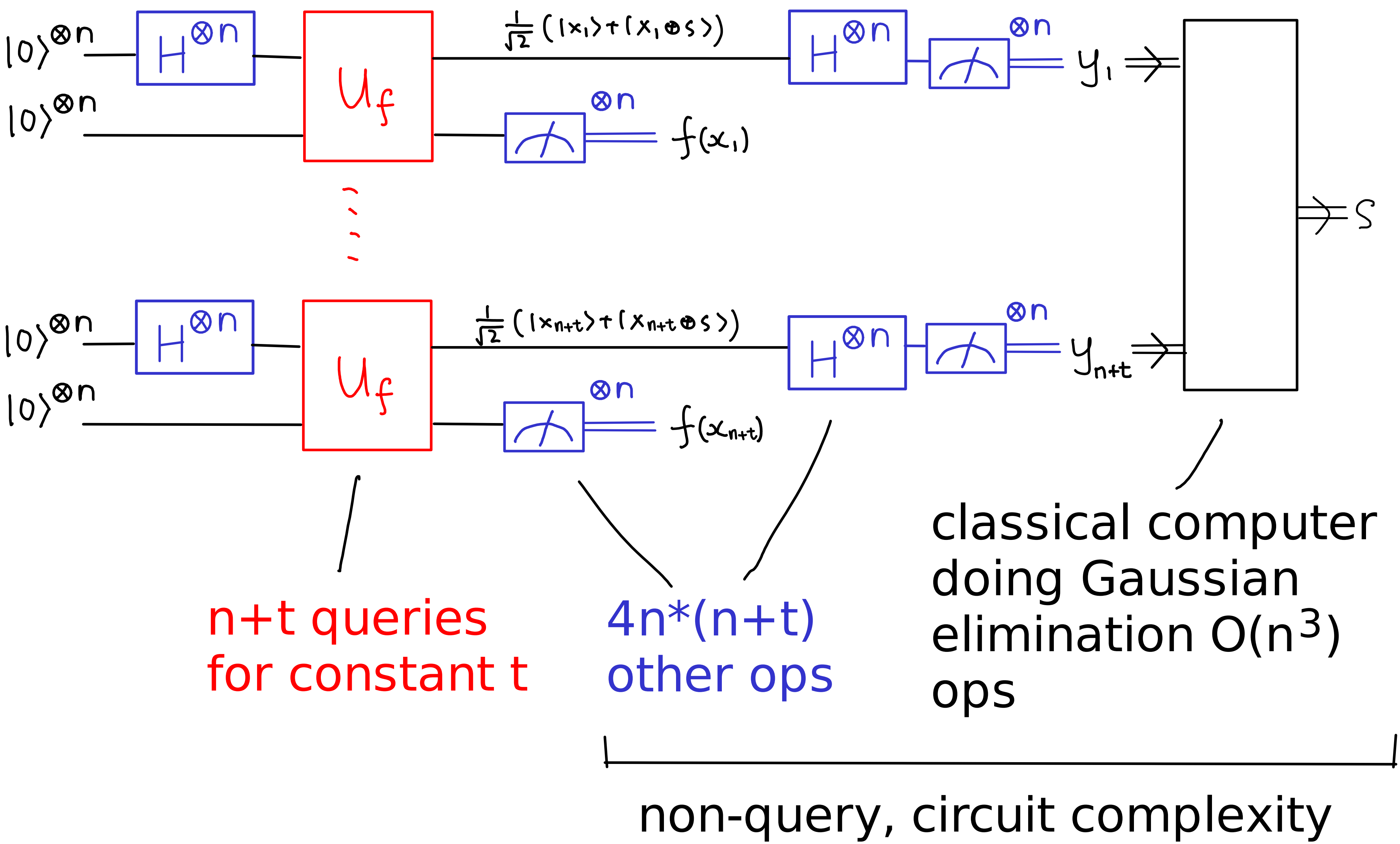
Simon's algorithm altogether:



Classically compute s from $y_1, \cdots, y_{n+t}$

With prob at least $1 - \frac{1}{2^{t+1}}$, s can be found.

Simon's algorithm altogether:



7. Quantum algorithms (part 1)

(a) Quantum query complexity: (KLM 9.2*, 6.2*)
black box model, phase kick back

(d) Deutsch-Jozsa algorithm
(NC 1.4.2-1.4.5, KLM 6.3-6.4, M 2.2)

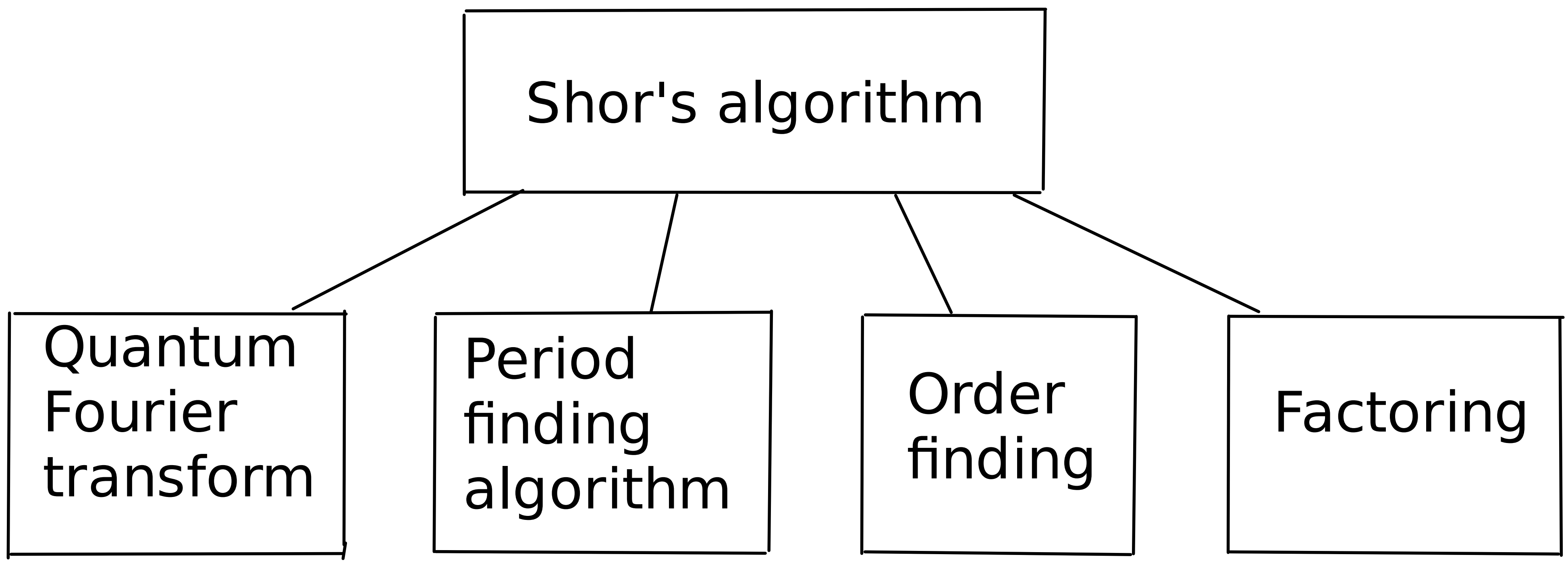
(e) Quantum fourier transform (I)
(NC 5.1, M 3.5, KLM p110-117)

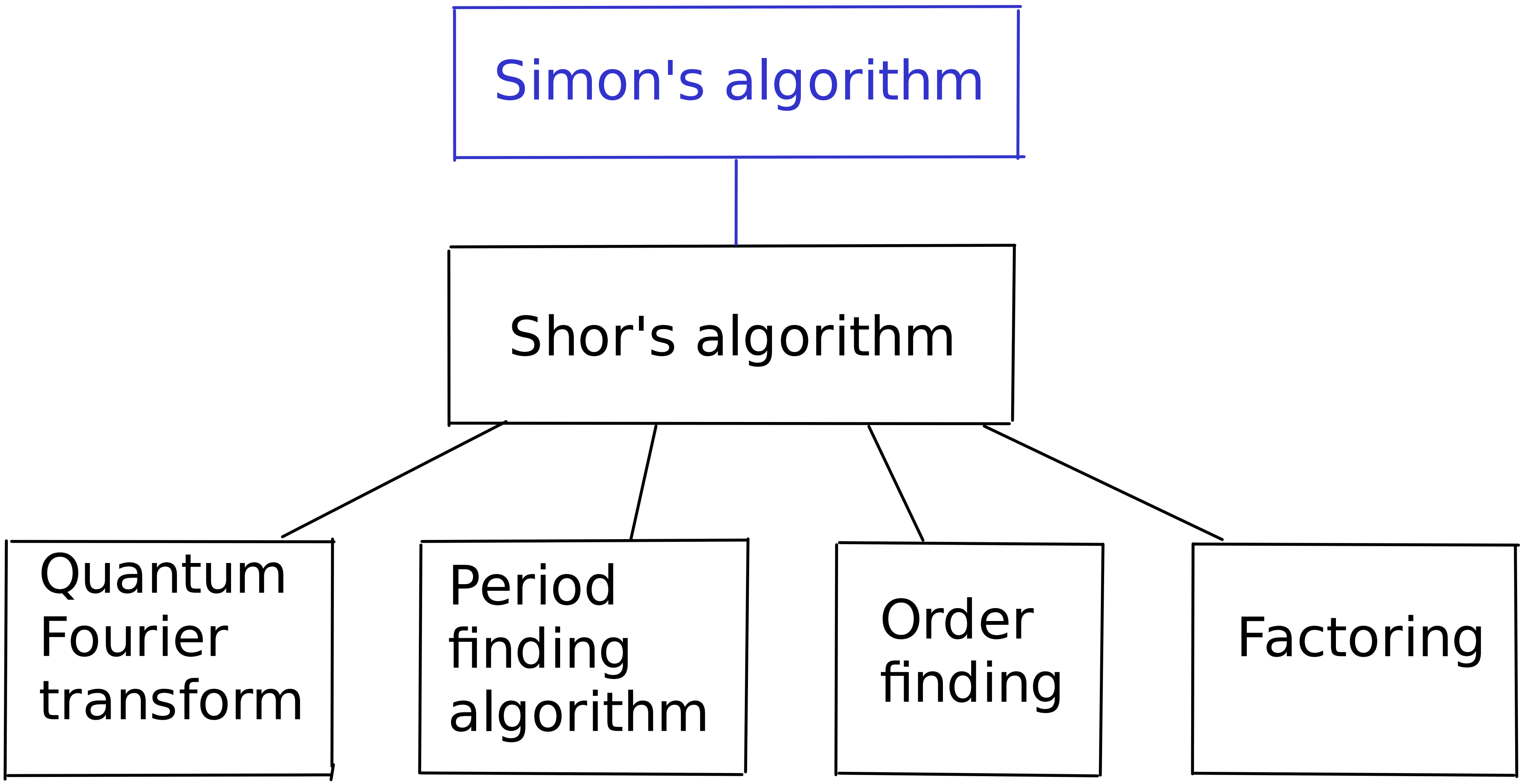
(f) Simon's algorithm (M 2.5, KLM 6.5)

(g) Shor's factoring algorithm
(M 3.1-3.4,3.7-3.10, NC 5.3, 5.4.1-5.4.2,
KLM 7.1.2-7.1.3, 7.3.1-7.3.2, 7.3.4, 7.4)

(h) Hidden subgroup framework (NC 5.4.3, KLM 7.5)

Deustch-Josza	Simon's	Shor's
speed up only if seeking exact solution	allows error in solution	allows error in solution
Quantum solution	Quantum subroutine	Quantum subroutine
	Classical processing	Heavy classical processing
	Analysis	Heavy Analysis
Concocted problem	Concocted problem	Critical problem for crypto
Speed-up in blackbox model	Speed-up in blackbox model	Natural model, no proof of speed-up


Shor's algorithm
Quantum Fourier transform
Period finding algorithm
Order finding
Factoring


Simon's algorithm
Shor's algorithm
Quantum Fourier transform
Period finding algorithm
Order finding
Factoring

Shor: key to Simon's algorithm was periodicity, and periodicity was closely connected with discrete log.

The problems by DJ and Simon are made to demonstrate quantum advantage using special properties of the Fourier transform $H^{\otimes n}$.

The natural (and interesting problem) discrete log or period finding requires new tools.

New tool (1): a new quantum Fourier transform.

Quantum Fourier transform over $(\mathbb{Z}_2)^n$

$$H^{\otimes n} |x\rangle = \frac{1}{\sqrt{2^n}} \sum_y (-1)^{x \cdot y} |y\rangle$$

$$|x_1 x_2 \cdots x_n\rangle \qquad x_1 y_1 \oplus \cdots \oplus x_n y_n \qquad |y_1 y_2 \cdots y_n\rangle$$

Quantum Fourier transform over $(\mathbb{Z}_2)^n$

$$H^{\otimes n}|x\rangle = \frac{1}{\sqrt{2^n}} \sum_y (-1)^{x \cdot y} |y\rangle$$

$|x_1 x_2 \cdots x_n\rangle$ $\quad$ $x_1 y_1 \oplus \cdots \oplus x_n y_n$ $\quad$ $|y_1 y_2 \cdots y_n\rangle$

Quantum Fourier transform over $\mathbb{Z}_d$

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

$x \in \mathbb{Z}_d$

primitive
dth root
of unity

multiplication
mod d

$y \in \mathbb{Z}_d = \{0, 1, \ldots, d-1\}$

$+, \times$ mod d

QFT over $(\mathbb{Z}_2)^n$:

e.g., n = 3, $H^{\otimes 3} =$

$$\frac{1}{\sqrt{8}}\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{pmatrix}$$

QFT over $(\mathbb{Z}_2)^n$:

e.g., n = 3, $H^{\otimes 3} =$

$$\frac{1}{\sqrt{8}}\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{pmatrix}$$

QFT over $\mathbb{Z}_d = \mathbb{Z}_{(2^n)}$:

e.g., d = 8, n = 3, $F =$

$$\frac{1}{\sqrt{8}}\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 & \omega^7 \\ 1 & \omega^2 & \omega^4 & \omega^6 & 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega & \omega^4 & \omega^7 & \omega^2 & \omega^5 \\ 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 \\ 1 & \omega^5 & \omega^2 & \omega^7 & \omega^4 & \omega & \omega^6 & \omega^3 \\ 1 & \omega^6 & \omega^4 & \omega^2 & 1 & \omega^6 & \omega^4 & \omega^2 \\ 1 & \omega^7 & \omega^6 & \omega^5 & \omega^4 & \omega^3 & \omega^2 & \omega \end{pmatrix}$$

QFT over $(\mathbb{Z}_2)^n$:

e.g., n = 3, $H^{\otimes 3} =$

$$\frac{1}{\sqrt{8}}\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{pmatrix}$$

Implementable with single qubit gates.

QFT over $\mathbb{Z}_d = \mathbb{Z}_{(2^n)}$:

e.g., d = 8, n = 3, $F =$

$$\frac{1}{\sqrt{8}}\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 & \omega^7 \\ 1 & \omega^2 & \omega^4 & \omega^6 & 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega & \omega^4 & \omega^7 & \omega^2 & \omega^5 \\ 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 \\ 1 & \omega^5 & \omega^2 & \omega^7 & \omega^4 & \omega & \omega^6 & \omega^3 \\ 1 & \omega^6 & \omega^4 & \omega^2 & 1 & \omega^6 & \omega^4 & \omega^2 \\ 1 & \omega^7 & \omega^6 & \omega^5 & \omega^4 & \omega^3 & \omega^2 & \omega \end{pmatrix}$$

How to implement this with a circuit?

(consider $d = 2^n$ only...)

Quantum Fourier transform (QFT) over $\mathbb{Z}_{(2^n)}$

Standard basis: $|x\rangle$, $x = x_{n-1} \cdots x_1 x_0 \in \{0,1\}^n$

Associate x with an integer: $x = \sum_{j=0}^{n-1} x_j 2^j$

Quantum Fourier transform (QFT) over $\mathbb{Z}_{(2^n)}$

Standard basis: $|x\rangle$, $x = x_{n-1} \cdots x_1 x_0 \in \{0,1\}^n$

Associate x with an integer: $x = \sum_{j=0}^{n-1} x_j 2^j$

Fourier basis: $|\tilde{x}\rangle = \frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} e^{\frac{2\pi i}{2^n} x y} |y\rangle$

both an integer & an n-bit string

multiplication of integers

Exercise: check that the Fourier basis is orthonormal, i.e., $\langle \tilde{x} | \tilde{w} \rangle = \delta_{\tilde{x}\tilde{w}}$

How to implement the QFT efficiently as a circuit?

$$|\tilde{x}\rangle = \frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} e^{\frac{2\pi i}{2^n} x y} |y\rangle$$

How to implement the QFT efficiently as a circuit?

$$|\tilde{x}\rangle = \frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} e^{\frac{2\pi i}{2^n} x y} |y\rangle$$

$$= \frac{1}{\sqrt{2^n}} \sum_{y_0=0}^{1} \cdots \sum_{y_{n-1}=0}^{1} e^{\frac{2\pi i}{2^n} x \sum_{j=0}^{n-1} 2^j y_j} |y_{n-1} y_{n-2} \cdots y_1 y_0\rangle$$

How to implement the QFT efficiently as a circuit?

$$|\tilde{x}\rangle = \frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} e^{\frac{2\pi i}{2^n} x y} |y\rangle$$

$$= \frac{1}{\sqrt{2^n}} \sum_{y_0=0}^{1} \cdots \sum_{y_{n-1}=0}^{1} e^{\frac{2\pi i}{2^n} x \sum_{j=0}^{n-1} 2^j y_j} |y_{n-1} y_{n-2} \cdots y_1 y_0\rangle$$

$$= \frac{1}{\sqrt{2^n}} \sum_{y_0=0}^{1} \cdots \sum_{y_{n-1}=0}^{1} \prod_{j=0}^{n-1} e^{\frac{2\pi i}{2^{n-j}} x y_j} |y_{n-1} y_{n-2} \cdots y_1 y_0\rangle$$

How to implement the QFT efficiently as a circuit?

$$|\tilde{x}\rangle = \frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} e^{\frac{2\pi i}{2^n} x y} |y\rangle$$

$$= \frac{1}{\sqrt{2^n}} \sum_{y_0=0}^{1} \cdots \sum_{y_{n-1}=0}^{1} e^{\frac{2\pi i}{2^n} x \sum_{j=0}^{n-1} 2^j y_j} |y_{n-1} y_{n-2} \cdots y_1 y_0\rangle$$

$$= \frac{1}{\sqrt{2^n}} \sum_{y_0=0}^{1} \cdots \sum_{y_{n-1}=0}^{1} \prod_{j=0}^{n-1} e^{\frac{2\pi i}{2^{n-j}} x y_j} |y_{n-1} y_{n-2} \cdots y_1 y_0\rangle$$

$$= \frac{1}{\sqrt{2}} \sum_{y_{n-1}=0}^{1} e^{\frac{2\pi i}{2^1} x y_{n-1}} |y_{n-1}\rangle \otimes \frac{1}{\sqrt{2}} \sum_{y_{n-2}=0}^{1} e^{\frac{2\pi i}{2^2} x y_{n-2}} |y_{n-2}\rangle \otimes \cdots$$

$$\cdots \otimes \frac{1}{\sqrt{2}} \sum_{y_1=0}^{1} e^{\frac{2\pi i}{2^{n-1}} x y_1} |y_1\rangle \otimes \frac{1}{\sqrt{2}} \sum_{y_0=0}^{1} e^{\frac{2\pi i}{2^n} x y_0} |y_0\rangle$$

Use the association: $X = \sum_{j=0}^{n-1} 2^j x_j$

* For the first qubit:

$$\frac{1}{\sqrt{2}} \sum_{y_{n-1}=0}^{1} e^{\frac{2\pi i}{2^1} x\, y_{n-1}} |y_{n-1}\rangle$$

Use the association: $X = \sum_{j=0}^{n-1} 2^j x_j$

* For the first qubit:

$$\frac{1}{\sqrt{2}} \sum_{y_{n-1}=0}^{1} e^{\frac{2\pi i}{2^1} x\, y_{n-1}} |y_{n-1}\rangle = \frac{1}{\sqrt{2}} \left(\underbrace{|0\rangle}_{y_{n-1}=0} + e^{\frac{2\pi i}{2^1} x} \underbrace{|1\rangle}_{y_{n-1}=1} \right)$$

Use the association: $X = \sum_{j=0}^{n-1} 2^j x_j$

* For the first qubit:

$$\frac{1}{\sqrt{2}} \sum_{y_{n-1}=0}^{1} e^{\frac{2\pi i}{2^1} X y_{n-1}} |y_{n-1}\rangle = \frac{1}{\sqrt{2}} \Big(\underbrace{|0\rangle}_{y_{n-1}=0} + e^{\frac{2\pi i}{2^1} x_0} \underbrace{|1\rangle}_{y_{n-1}=1} \Big)$$

$$e^{\frac{2\pi i}{2^1} X y_{n-1}} = e^{\frac{2\pi i}{2^1} x_0 y_{n-1}}$$

Use the association: $X = \sum_{j=0}^{n-1} 2^j x_j$

* For the first qubit: $e^{\frac{2\pi i}{2^1} x\, y_{n-1}} = e^{\frac{2\pi i}{2^1} x_0\, y_{n-1}}$

$$\frac{1}{\sqrt{2}} \sum_{y_{n-1}=0}^{1} e^{\frac{2\pi i}{2^1} x\, y_{n-1}} |y_{n-1}\rangle = \frac{1}{\sqrt{2}} \left(|0\rangle + \underbrace{e^{\frac{2\pi i}{2^1} x_0}}_{(-1)^{x_0}} |1\rangle \right)$$

Use the association: $X = \sum_{j=0}^{n-1} 2^j x_j$

* For the first qubit: $e^{\frac{2\pi i}{2^1} X y_{n-1}} = e^{\frac{2\pi i}{2^1} x_0 y_{n-1}}$

$$\frac{1}{\sqrt{2}} \sum_{y_{n-1}=0}^{1} e^{\frac{2\pi i}{2^1} X y_{n-1}} |y_{n-1}\rangle = \frac{1}{\sqrt{2}} \left(|0\rangle + \underbrace{e^{\frac{2\pi i}{2^1} x_0}}_{(-1)^{x_0}} |1\rangle \right)$$

$$= H |x_0\rangle$$

Use the association: $X = \sum_{j=0}^{n-1} 2^j x_j$

* For the first qubit: $e^{\frac{2\pi i}{2^1} X y_{n-1}} = e^{\frac{2\pi i}{2^1} x_0 y_{n-1}}$

$$\frac{1}{\sqrt{2}} \sum_{y_{n-1}=0}^{1} e^{\frac{2\pi i}{2^1} X y_{n-1}} |y_{n-1}\rangle = \frac{1}{\sqrt{2}} \left(|0\rangle + \underbrace{e^{\frac{2\pi i}{2^1} x_0}}_{(-1)^{x_0}} |1\rangle \right)$$

$$= H |x_0\rangle$$

* For the second qubit:

$$\frac{1}{\sqrt{2}} \sum_{y_{n-2}=0}^{1} e^{\frac{2\pi i}{2^2} X y_{n-2}} |y_{n-2}\rangle$$

Use the association: $X = \sum_{j=0}^{n-1} 2^j x_j$

* For the first qubit: $e^{\frac{2\pi i}{2^1} X y_{n-1}} = e^{\frac{2\pi i}{2^1} x_0 y_{n-1}}$

$$\frac{1}{\sqrt{2}} \sum_{y_{n-1}=0}^{1} e^{\frac{2\pi i}{2^1} X y_{n-1}} |y_{n-1}\rangle = \frac{1}{\sqrt{2}} \left(|0\rangle + \underbrace{e^{\frac{2\pi i}{2^1} x_0}}_{(-1)^{x_0}} |1\rangle \right)$$

$$= H |x_0\rangle$$

* For the second qubit: $e^{\frac{2\pi i}{2^2} X y_{n-2}} = e^{\frac{2\pi i}{2^2} x_0 y_{n-2} + \frac{2\pi i}{2^1} x_1 y_{n-2}}$

$$\frac{1}{\sqrt{2}} \sum_{y_{n-2}=0}^{1} e^{\frac{2\pi i}{2^2} X y_{n-2}} |y_{n-2}\rangle$$

Use the association: $X = \sum_{j=0}^{n-1} 2^j x_j$

* For the first qubit: $e^{\frac{2\pi i}{2^1} X y_{n-1}} = e^{\frac{2\pi i}{2^1} x_0 y_{n-1}}$

$$\frac{1}{\sqrt{2}} \sum_{y_{n-1}=0}^{1} e^{\frac{2\pi i}{2^1} X y_{n-1}} |y_{n-1}\rangle = \frac{1}{\sqrt{2}} \left(|0\rangle + \underbrace{e^{\frac{2\pi i}{2^1} x_0}}_{(-1)^{x_0}} |1\rangle \right)$$

$$= H |x_0\rangle$$

* For the second qubit: $e^{\frac{2\pi i}{2^2} X y_{n-2}} = e^{\frac{2\pi i}{2^2} x_0 y_{n-2} + \frac{2\pi i}{2^1} x_1 y_{n-2}}$

$$\frac{1}{\sqrt{2}} \sum_{y_{n-2}=0}^{1} e^{\frac{2\pi i}{2^2} X y_{n-2}} |y_{n-2}\rangle = \frac{1}{\sqrt{2}} \left(\underset{y_{n-2}=0}{|0\rangle} + e^{\frac{2\pi i}{2^2} x_0 + \frac{2\pi i}{2^1} x_1} \underset{y_{n-2}=1}{|1\rangle} \right)$$

Use the association: $X = \sum_{j=0}^{n-1} 2^j x_j$

* For the first qubit: $e^{\frac{2\pi i}{2^1} x y_{n-1}} = e^{\frac{2\pi i}{2^1} x_0 y_{n-1}}$

$$\frac{1}{\sqrt{2}} \sum_{y_{n-1}=0}^{1} e^{\frac{2\pi i}{2^1} x y_{n-1}} |y_{n-1}\rangle = \frac{1}{\sqrt{2}} \Big(|0\rangle + \underbrace{e^{\frac{2\pi i}{2^1} x_0}}_{(-1)^{x_0}} |1\rangle \Big)$$

$$= H |x_0\rangle$$

* For the second qubit: $e^{\frac{2\pi i}{2^2} x y_{n-2}} = e^{\frac{2\pi i}{2^2} x_0 y_{n-2} + \frac{2\pi i}{2^1} x_1 y_{n-2}}$

$$\frac{1}{\sqrt{2}} \sum_{y_{n-2}=0}^{1} e^{\frac{2\pi i}{2^2} x y_{n-2}} |y_{n-2}\rangle = \frac{1}{\sqrt{2}} \Big(|0\rangle + e^{\frac{2\pi i}{2^2} x_0 + \frac{2\pi i}{2^1} x_1} |1\rangle \Big)$$

$$= \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^2} \end{bmatrix}^{x_0} H |x_1\rangle$$

* For the (n-k)-th qubit:

$$e^{\frac{2\pi i}{2^{n-k}} x y_k} = e^{2\pi i \left(\frac{x_0}{2^{n-k}} + \frac{x_1}{2^{n-k-1}} + \cdots + \frac{x_{n-k-1}}{2}\right) y_k}$$

the $x_{n-k}, x_{n-k+1}, \cdots \; x_{n-1}$ terms

are multiplied to an integer * $2\pi i$

* For the (n-k)-th qubit:

$$e^{\frac{2\pi i}{2^{n-k}} x y_k} = e^{2\pi i \left(\frac{x_0}{2^{n-k}} + \frac{x_1}{2^{n-k-1}} + \cdots + \frac{x_{n-k-1}}{2}\right) y_k}$$

the $x_{n-k}, x_{n-k+1}, \cdots x_{n-1}$ terms

are multiplied to an integer * $2\pi i$

$$\frac{1}{\sqrt{2}} \sum_{y_k=0}^{1} e^{\frac{2\pi i}{2^{n-k}} x y_k} |y_k\rangle$$

$$= \frac{1}{\sqrt{2}} \sum_{y_k=0}^{1} e^{2\pi i \left(\frac{x_0}{2^{n-k}} + \frac{x_1}{2^{n-k-1}} + \cdots + \frac{x_{n-k-1}}{2}\right) y_k} |y_k\rangle$$

$$= \frac{1}{\sqrt{2}} \sum_{y_k=0}^{1} e^{2\pi i \left(\frac{x_0}{2^{n-k}} + \frac{x_1}{2^{n-k-1}} + \cdots + \frac{x_{n-k-1}}{2} \right) y_k} |y_k\rangle$$

$$= \frac{1}{\sqrt{2}} \sum_{y_k=0}^{1} e^{2\pi i \left(\frac{x_0}{2^{n-k}} + \frac{x_1}{2^{n-k-1}} + \cdots + \frac{x_{n-k-1}}{2} \right) y_k} |y_k\rangle$$

$$= \frac{1}{\sqrt{2}} \Big(\underset{y_k=0}{|0\rangle} + e^{2\pi i \left(\frac{x_0}{2^{n-k}} + \frac{x_1}{2^{n-k-1}} + \cdots + \frac{x_{n-k-1}}{2} \right)} \underset{y_k=1}{|1\rangle} \Big)$$

$$= \frac{1}{\sqrt{2}} \sum_{y_k=0}^{1} e^{2\pi i \left(\frac{x_0}{2^{n-k}} + \frac{x_1}{2^{n-k-1}} + \cdots + \frac{x_{n-k-1}}{2} \right) y_k} |y_k\rangle$$

$$= \frac{1}{\sqrt{2}} \left(|0\rangle + e^{2\pi i \left(\frac{x_0}{2^{n-k}} + \frac{x_1}{2^{n-k-1}} + \cdots + \frac{x_{n-k-1}}{2} \right)} |1\rangle \right)$$

$$= R_{n-k}^{x_0} R_{n-k-1}^{x_1} \cdots R_3^{x_{n-k-3}} R_2^{x_{n-k-2}} H |x_{n-k-1}\rangle$$

where $R_m = \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^m} \end{bmatrix}$

exp to 0 or 1
<- R_2 with
control x_{n-k-2}

* For the first qubit: $H\,|x_0\rangle$

* For the second qubit: $\begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^2} \end{bmatrix}^{x_0} H\,|x_1\rangle$

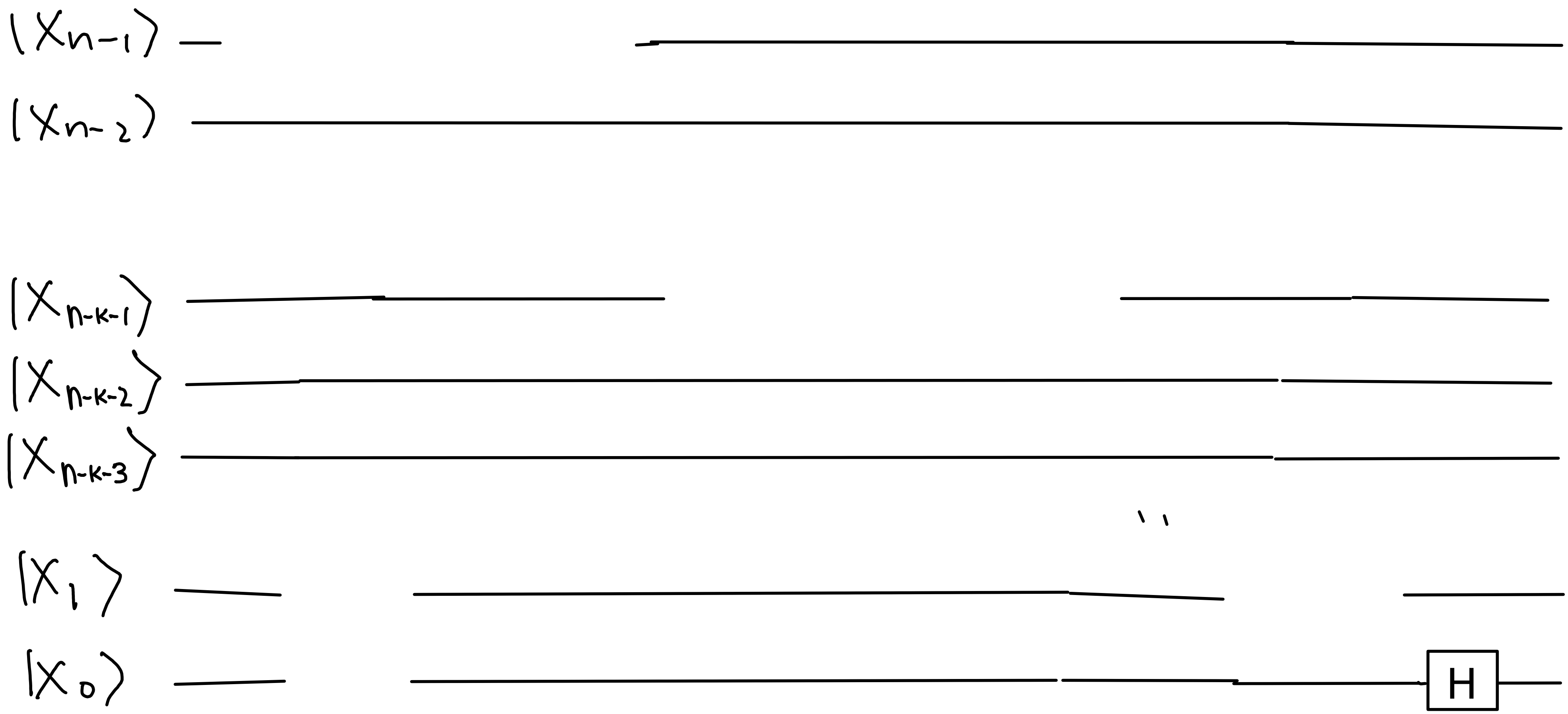
$\vdots$

* For the (n-k)-th qubit:

$$R_{n-k}^{x_0}\, R_{n-k-1}^{x_1} \cdots R_3^{x_{n-k-3}}\, R_2^{x_{n-k-2}}\, H\,|x_{n-k-1}\rangle$$

So $|x\rangle \longrightarrow |\tilde{x}\rangle$ can be implemented as

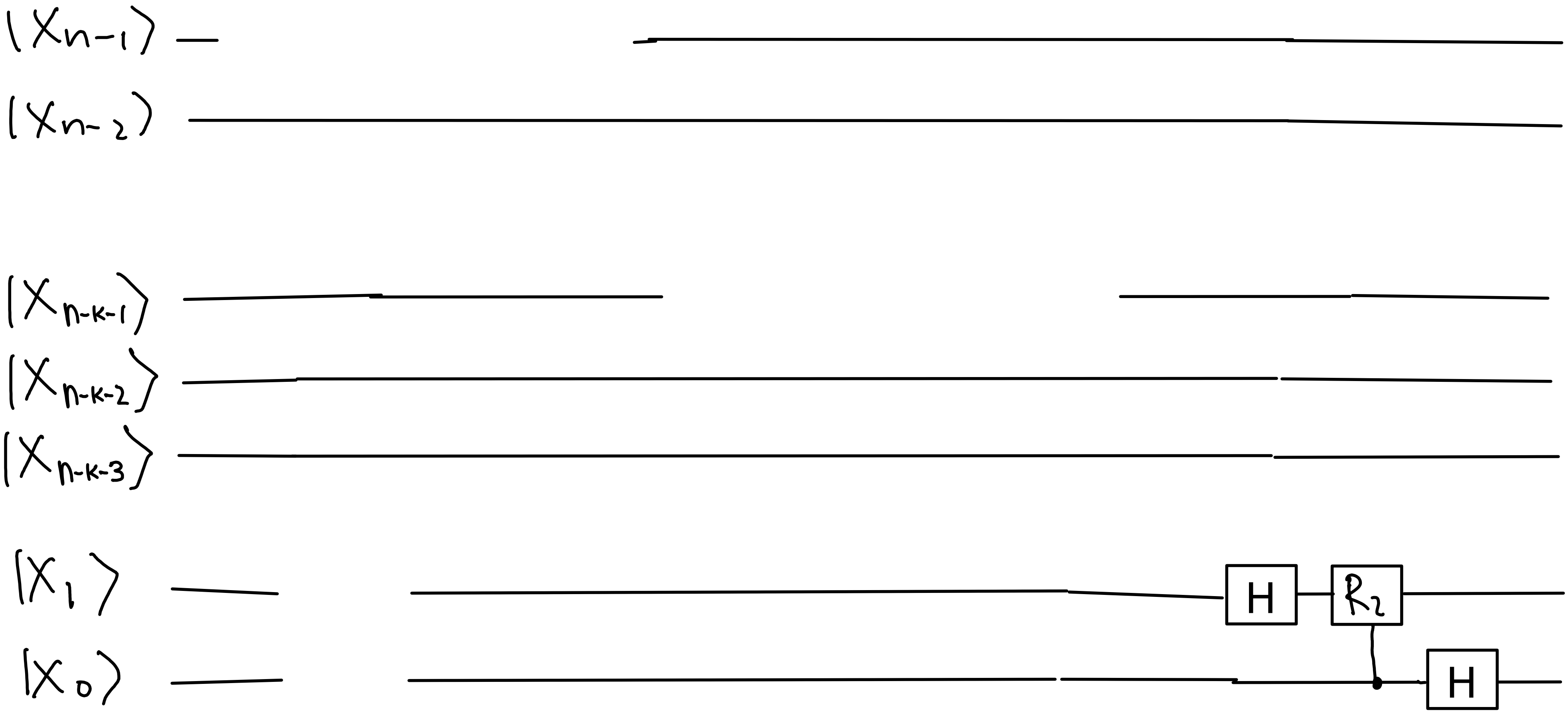
up to reversing the order of the output qubits



* For the first qubit: $H\,|x_0\rangle$

So $|x\rangle \longrightarrow |\tilde{x}\rangle$ can be implemented as

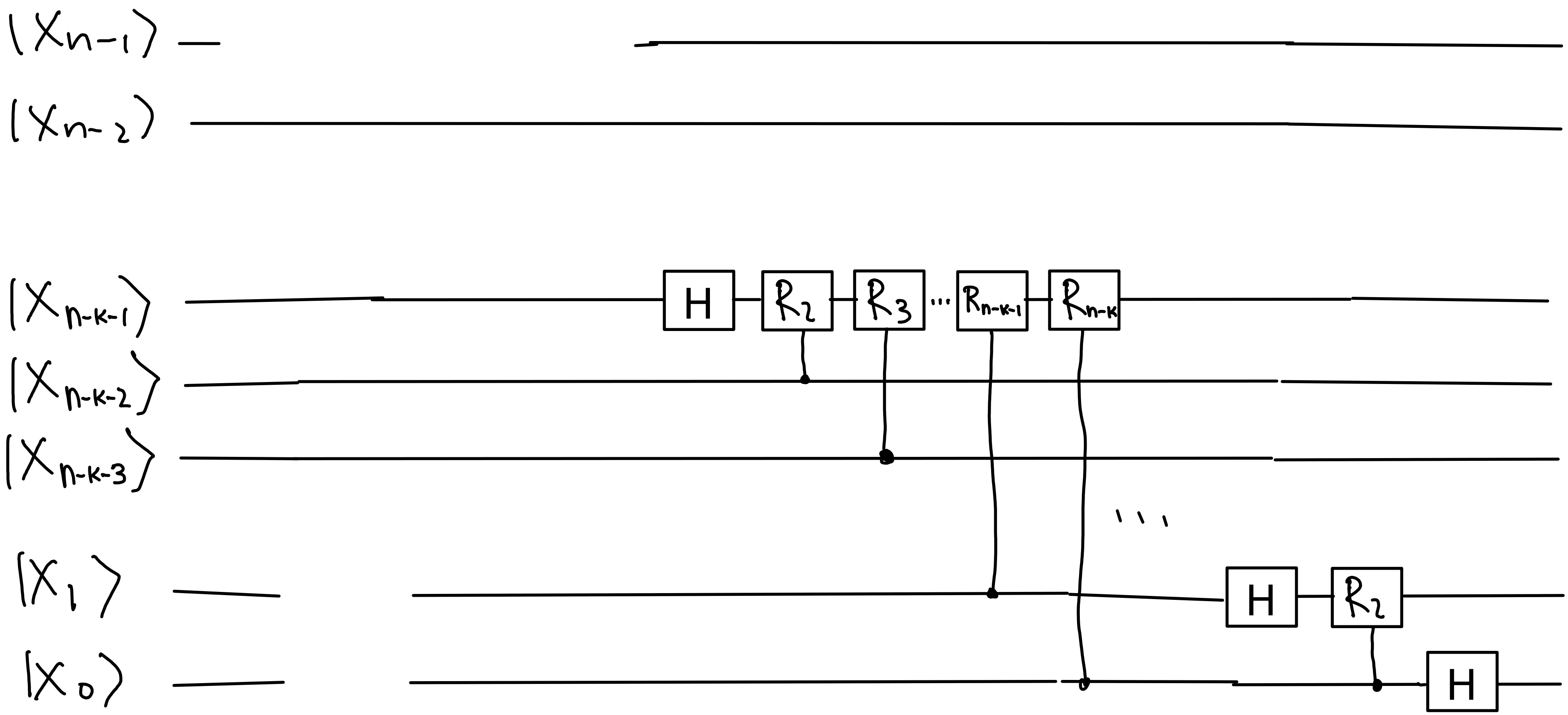
up to reversing the order of the output qubits



* For the second qubit: $\begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^2} \end{bmatrix}^{x_0}$ $H|x_1\rangle$

So $|x\rangle \rightarrow |\tilde{x}\rangle$ can be implemented as

up to reversing the order of the output qubits

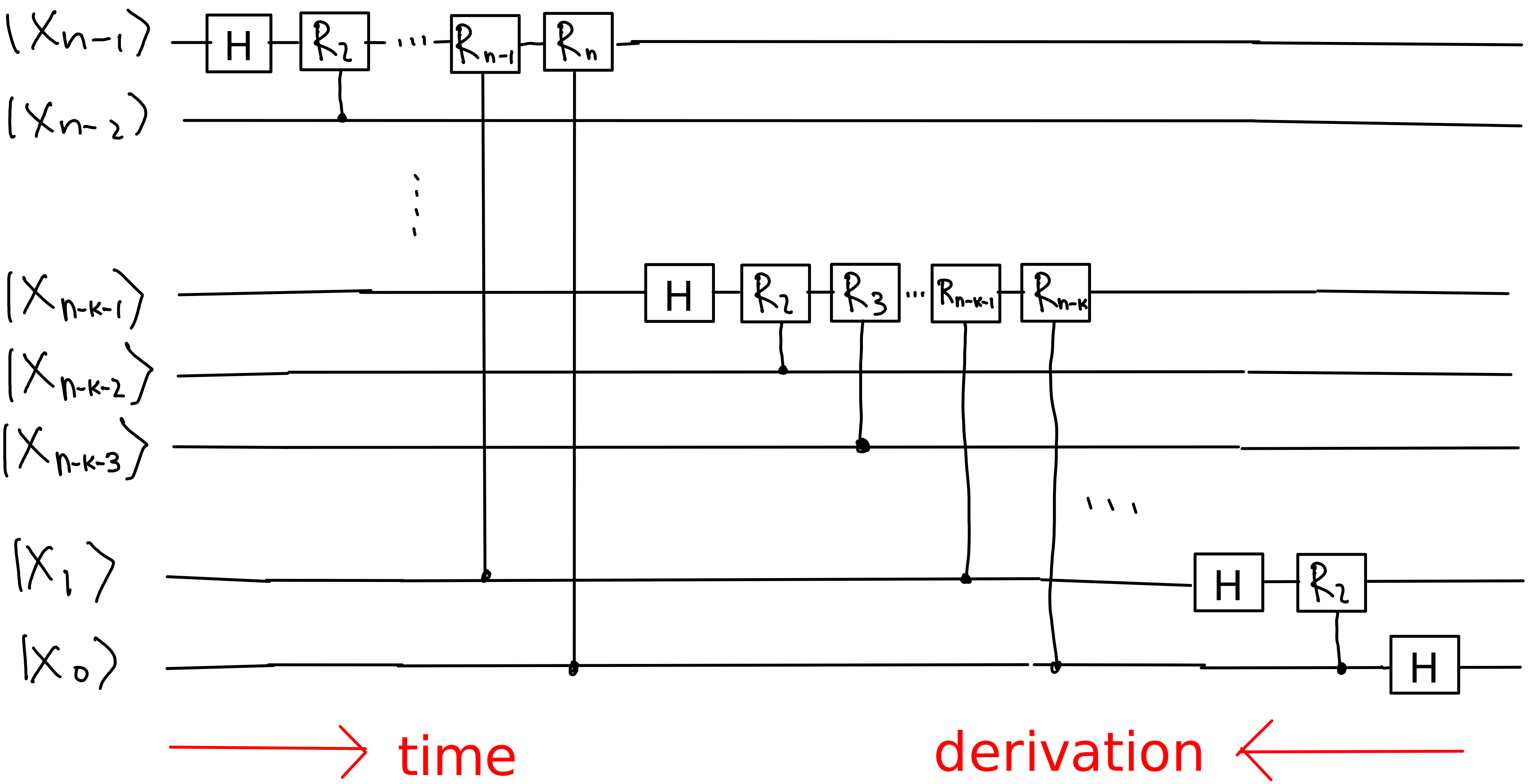


* For the (n-k)-th qubit:

$$R_{n-k}^{x_0} \; R_{n-k-1}^{x_1} \cdots R_3^{x_{n-k-3}} \; R_2^{x_{n-k-2}} \quad H|x_{n-k-1}\rangle$$

So $|x\rangle \longrightarrow |\tilde{x}\rangle$ can be implemented as

up to reversing the order of the output qubits

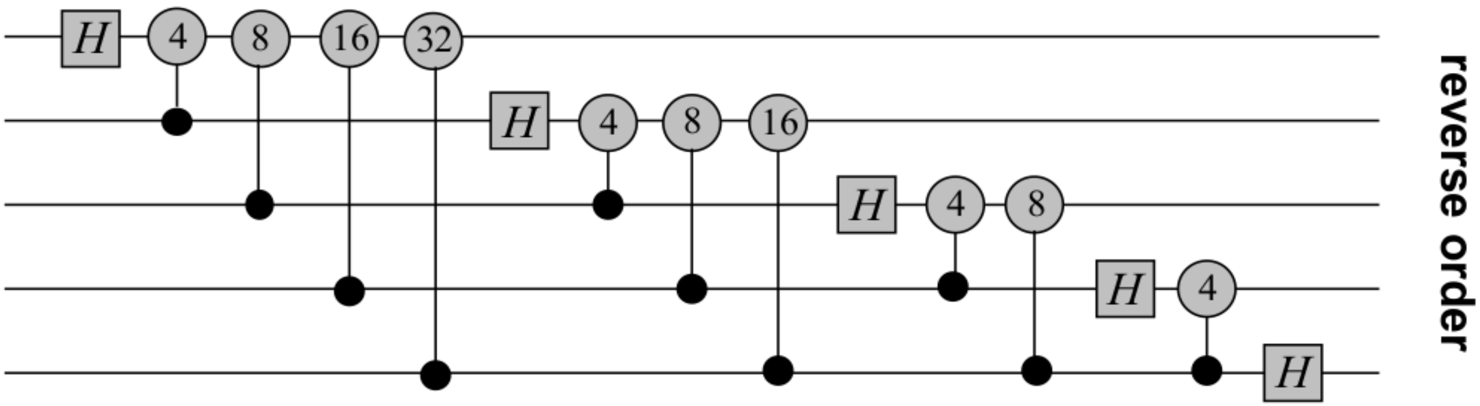


NB 1. alternative proof QFT is unitary

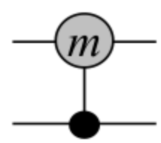
2. circuit size $O(n^2)$. A3 Q1: improve to $O(n \log n)$.

Computing the QFT for $m = 2^n$ (1)

Quantum circuit for F_{32}:



Gates: $-\boxed{H}- = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$



$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{2\pi i/m} \end{bmatrix}$$

For F_{2^n} costs $O(n^2)$ gates

From Professor Cleve lecture notes.

Exercise: work through the derivation of the QFT circuit for 3 or 4 qubits.