Quantum Computer Algorithms

Michele Mosca

Fields Institute Summer School in Quantum Information Processing

Crash Course on Computational Complexity

- Computational Complexity
- Computing Models
- Some notation
- Uniformity

Computational Complexity

- We usually measure the amount of resources (e.g. time, space, gates) used by an algorithm as a function of the input size.
- E.g. The grade-school algorithm for multiplying two n-bit integers uses $O(n^2)$ time steps. FFT methods use $O(n(\log n)(\log\log n))$ time steps. The best known lower bound is $\Omega(n)$ steps.

“polynomial” cost

- When we say an algorithm uses a polynomial amount of some resource (e.g. time, space, gates, energy), we mean that there is some polynomial $p(n)$ such that the amount of that resource used by the algorithm is in $O(p(n))$
- E.g. we can multiply n-bit numbers in polynomial time.

“polynomial” cost

- If the cost is not bounded above by a polynomial, we say its “super-polynomial”; sometimes people abuse the term “exponential” to mean super-polynomial
- E.g. the best rigorous probabilistic classical algorithm for factoring n-bit numbers uses time in $e^{O(\sqrt{n \log n})}$
- So there is no known polynomial time classical algorithm for factoring

What’s so special about polynomials?

- The Strong Church-Turing thesis states that a probabilistic Turing machine can simulate any reasonable algorithmic process with at most a polynomial overhead
- Using polynomial cost as our notion of “efficiency” is very convenient.

Computing Models

- Two commonly used models are the Turing machine model and the circuit model

Turing machines

- Turing machines can take inputs of any size.
- We measure the time complexity of a computation on a Turing machine by the number of steps taken before the TM stops
- The space complexity is the number of tape positions used for the computation
- We usually consider the *worst case* complexity for an input of a fixed size n.

Asymptotic Notation

- A function f(n) is in O(g(n)) if for some constant m there exists a positive constant c such that $f(n) \leq c\, g(n)$ for all $n \geq m$
- A function f(n) is in $\Omega(g(n))$ if for some constant m there exists a positive constant c such that $f(n) \geq c\, g(n)$ for all $n \geq m$
- A function f(n) is in $\Theta(g(n))$ if for some constant m there exists positive constants $c_1 \leq c_2$ such that $c_1\, g(n) \leq f(n) \leq c_2\, g(n)$ for all $n \geq m$

Circuits

- We usually measure the complexity of a circuit C_n by its size, $|C_n|$, which is the number of gates in it.
- We can also measure the depth (or time), and the space (or width).
- Circuits only take a fixed size input. So how can we fairly compare them to Turing machines?

Families of Circuits

- We consider families of circuits $\{C_n\}$ where C_n takes inputs of size n.
- We can, e.g., design a family of multiplication circuits where C_n has size $O(n^2)$ or $O(n \log n \log\log n)$.
- Recall that the description of a Turing machine is finite. Where do we keep an infinite family of circuits?

Families of Circuits

- We have a procedure (e.g. a Turing machine) that generates the circuit diagrams for us
- For the size of the circuit C_n to fairly reflect the complexity of solving a problem on an input of size n, the complexity of generating the circuit must be "reasonable"

Families of Circuits

- The definition of “reasonable” varies depending one what you are trying to prove, but as a bare minimum, we expect the time and space complexities of generating C_n to be at most polynomial in the size of C_n
- For most of the circuits we will encounter, it will be clear that we can efficiently generate C_n given the integer n

Families of Circuits

- A family of circuits that can be efficiently generated is a *uniform family* of circuits
- Non-uniform families of circuits can require exponential resources to construct. It is possible to hide valuable information in the circuit C_n that we might not be able to compute from scratch using poly($|C_n|$) resources. It is not appropriate to use $|C_n|$ as a measure of the complexity of solving a problem "from scratch"

Uniform Families of Acyclic Quantum Circuits

- The computing model we will use for most of this course is *uniform families of acyclic circuits*
- The word "circuit" seems to refer to particular physical implementation of a computer. We will often use the terms "network" or "array of gates" instead.

Quantum Algorithms Overview

- Eigenvalue Estimation lets us factor integers
- Eigenvalue 'kick-back' turns eigenvalue estimation problem into phase estimation problem
- Quantum Fourier Transform and Phase Estimation
- Generalization to finding hidden subgroups
- Finding Hidden Affine Functions

Integer Factorization

- The security of many public key cryptosystems used in industry today relies on the difficulty of factoring large numbers into smaller factors.
- Factoring the integer N into smaller factors can be reduced to the following task:

Given integer a, find the smallest positive integer r so that $a^r \equiv 1 \bmod N$

Simple operator

Since we know how to efficiently multiply by a mod N, we can efficiently implement

$$U_a|x\rangle = |ax\rangle$$

Note that $U_a^r|x\rangle = |a^r x\rangle = |x\rangle$

i.e. $U_a^r = I$

Interesting eigenvalues

If $U_a^r = I$ then the eigenvalues of U_a are of the form $e^{2\pi i \frac{k}{r}}$

$$U_a|\psi_k\rangle = e^{i2\pi\frac{k}{r}}|\psi_k\rangle$$

$$|\psi_k\rangle = \sum_{j=0}^{r-1} e^{i2\pi j\frac{k}{r}}|a^j\rangle$$

Checking the eigenvalue

$$U_a|\psi_k\rangle = \sum_{j=0}^{r-1} e^{-i2\pi j\frac{k}{r}} U_a|a^j\rangle$$

$$= \sum_{j=0}^{r-1} e^{-i2\pi j\frac{k}{r}}|a^{j+1}\rangle = e^{i2\pi\frac{k}{r}}\left(\sum_{j=1}^{r} e^{-i2\pi j\frac{k}{r}}|a^j\rangle\right)$$

$$= e^{i2\pi\frac{k}{r}}\left(\sum_{j=0}^{r-1} e^{-i2\pi j\frac{k}{r}}|a^j\rangle\right) = e^{i2\pi\frac{k}{r}}|\psi_k\rangle$$

Finding r

For most integers k, a good estimate of $\frac{k}{r}$ (with error at most $\frac{1}{2r^2}$) allows us to determine r (even if we don't know k). (using continued fractions)

Where do we get $|\Psi_k\rangle$?

Since most k are good, a random $|\Psi_k\rangle$ suffices. Try $|1\rangle = \sum_{k=0}^{r-1} \frac{1}{\sqrt{r}} |\Psi_k\rangle$

Estimating Random Eigenvalue lets us Factor

In summary:
Factoring large numbers can be reduced to estimating a random eigenvalue of U_a

Must make the "global" phase a "relative" phase

A global phase has no physical significance.

In other words, states that differ only by a global phase are equivalent

$$U\left(\sum_x a_x |x\rangle\right) = \sum_x b_x |x\rangle$$

$$U\left(e^{i\theta}\sum_x a_x |x\rangle\right) = e^{i\theta}\sum_x b_x |x\rangle$$

so $$e^{i\theta}|\Phi\rangle \approx |\Phi\rangle$$

Must make the "global" phase a "relative" phase

A relative phase can affect outcome probabilities
E.g.

$$|0\rangle + e^{i\varphi}|1\rangle \xrightarrow{H} \left(\frac{1+e^{i\varphi}}{2}\right)|0\rangle + \left(\frac{1-e^{i\varphi}}{2}\right)|1\rangle$$

$$p_0 = \cos^2\left(\frac{\varphi}{2}\right)$$

Eigenvalue "kick-back"

We can also efficiently implement

$$c-U_a|0\rangle|x\rangle = |0\rangle|x\rangle$$

$$c-U_a|1\rangle|x\rangle = |1\rangle|ax\rangle$$

so

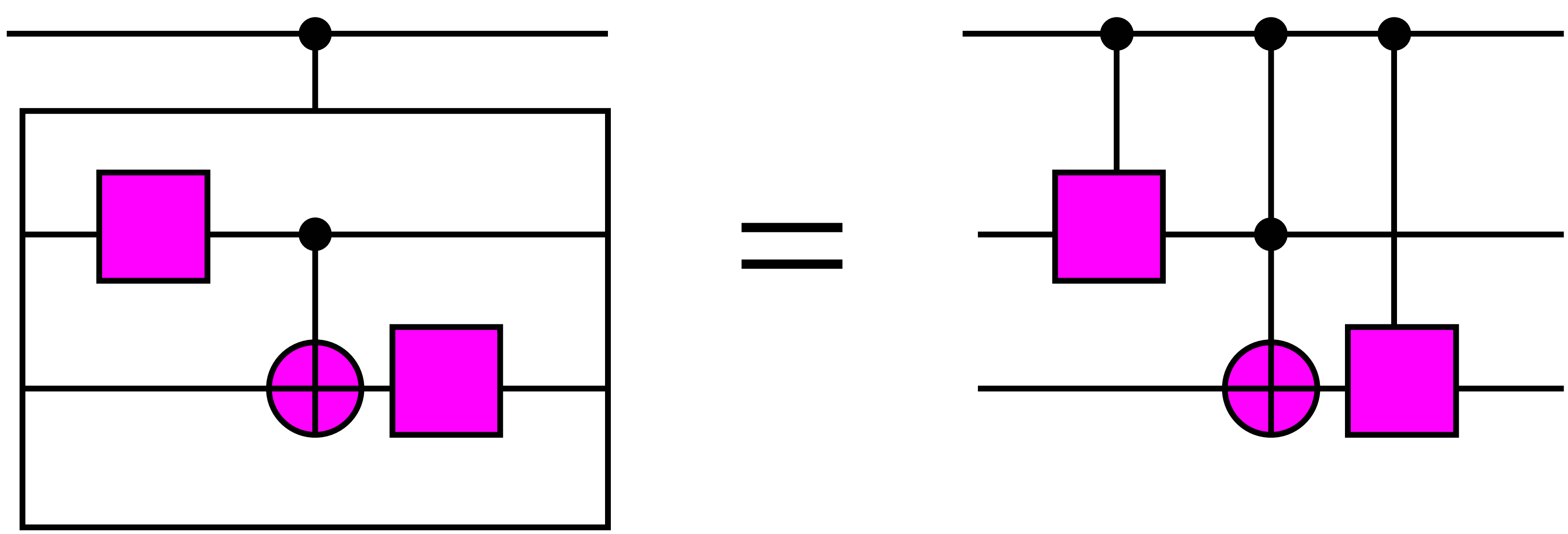
$$c-U_a|0\rangle|\Psi_k\rangle = |0\rangle|\Psi_k\rangle$$

$$c-U_a|1\rangle|\Psi_k\rangle = e^{2\pi i\frac{k}{r}}|1\rangle|\Psi_k\rangle$$

How do we implement c-U?

Replace every gate G in the circuit for with a c-G.

For example,

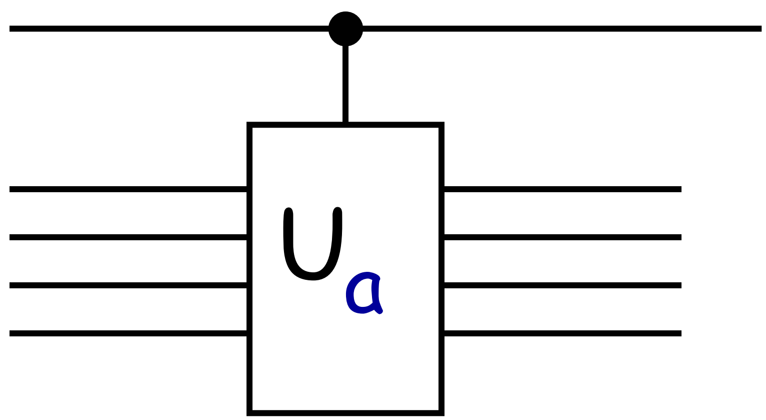


Eigenvalue kick-back

We can thus efficiently implement

$$|0\rangle + |1\rangle \qquad\qquad |0\rangle + e^{i2\pi\frac{k}{r}}|1\rangle$$

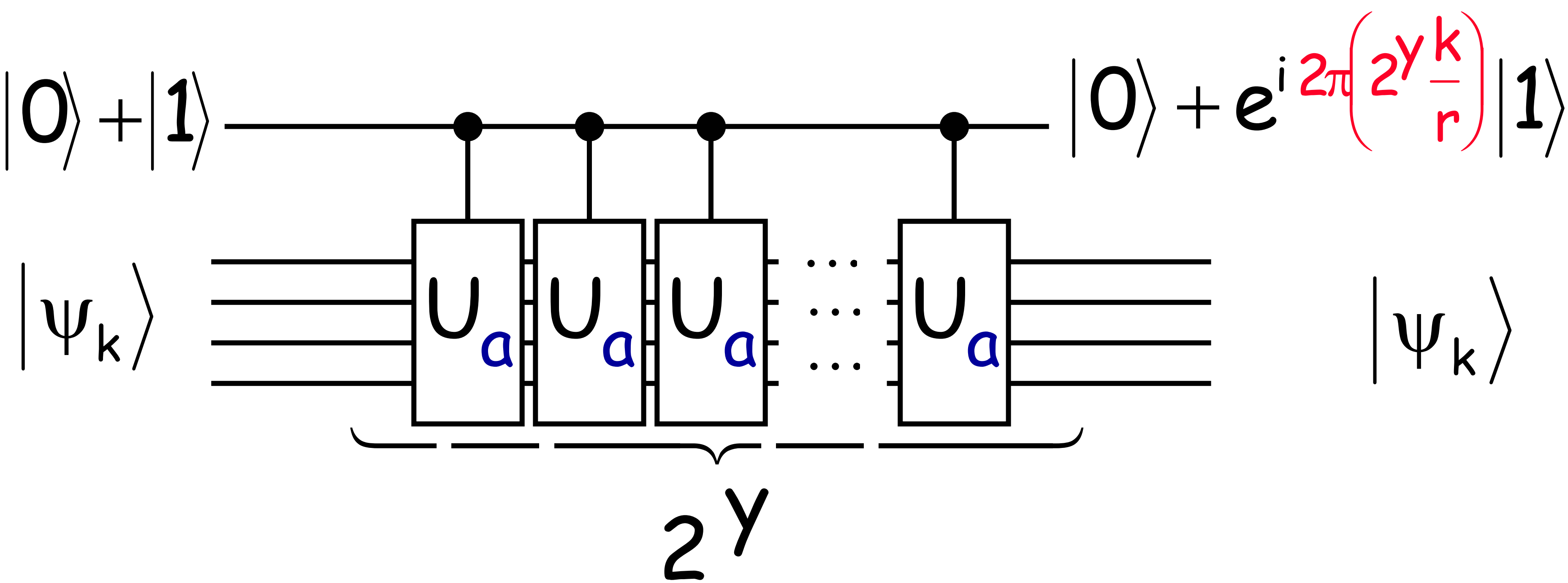
$$|\Psi_k\rangle \qquad\qquad |\Psi_k\rangle$$



This gives us a relative phase shift of $\varphi = 2\pi\frac{k}{r}$ in the control qubit

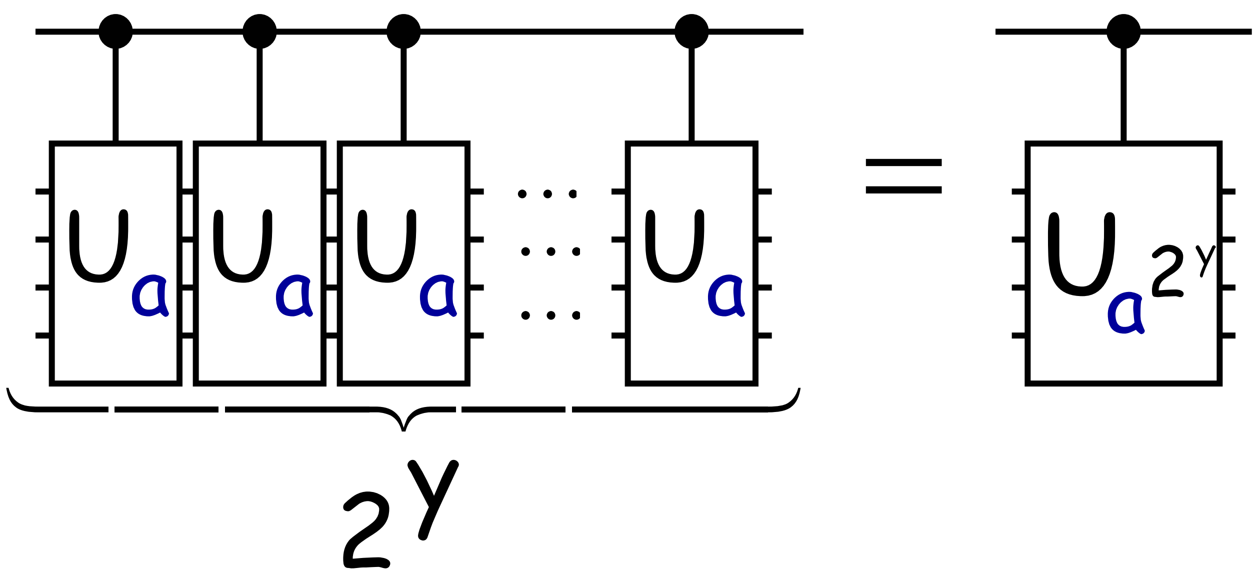
Inefficient exponentiation

We can effect a relative phase shift of $e^{i2\pi\left(2^y\frac{k}{r}\right)}$



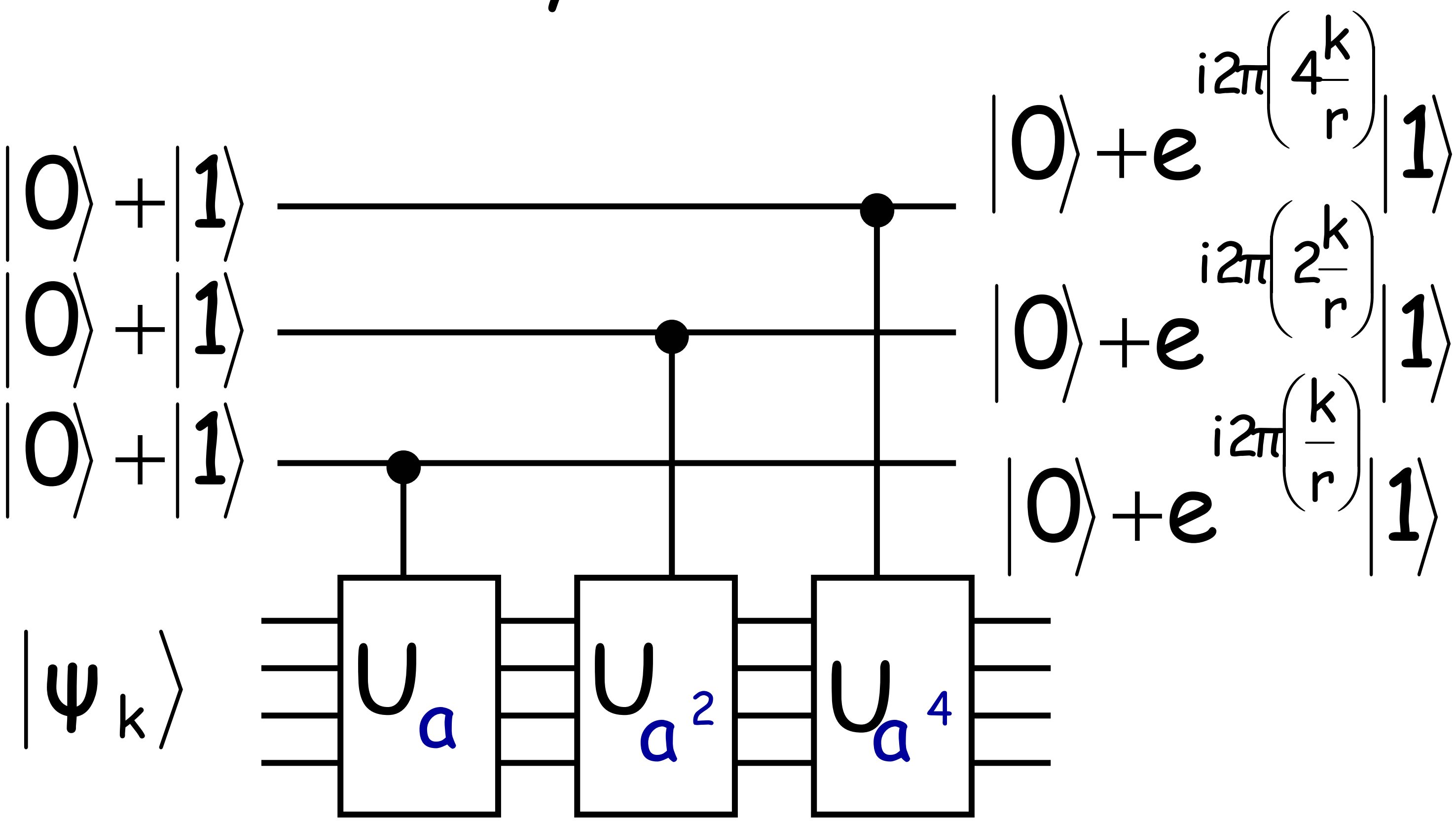
Efficient Exponentiation

But we can also do it **efficiently** by noticing that $U_a^{2^y} = U_{a^{2^y}}$



Reduction to phase estimation

We can efficiently construct



Phase Estimation

Given the qubits

$$\left(|0\rangle + e^{i2\pi\left(\frac{k}{r}\right)}|1\rangle\right)\left(|0\rangle + e^{i2\pi\left(2\frac{k}{r}\right)}|1\rangle\right)\cdots\left(|0\rangle + e^{i2\pi\left(2^{j}\frac{k}{r}\right)}|1\rangle\right)$$

Estimate $\frac{k}{r}$

Special Case

$$\left(|0\rangle + e^{i(\varphi)}|1\rangle\right) \quad \left(|0\rangle + e^{i(2\varphi)}|1\rangle\right) \quad \left(|0\rangle + e^{i(4\varphi)}|1\rangle\right)$$

Where

$$\frac{\varphi}{2\pi} = \frac{x}{8} = \frac{x_1x_2x_3}{8} = \frac{4x_1 + 2x_2 + x_3}{8} = 0.x_1x_2x_3$$

Since $e^{i2\pi} = 1$ then we have the state

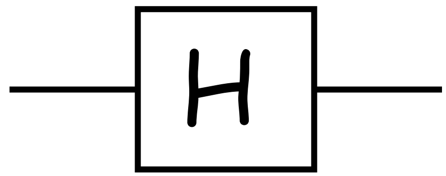
$$\left(|0\rangle + e^{i2\pi(0.x_1x_2x_3)}|1\rangle\right)\left(|0\rangle + e^{i2\pi(0.x_2x_3)}|1\rangle\right)\left(|0\rangle + e^{i2\pi(0.x_3)}|1\rangle\right)$$

Recall Hadamard transform

$$|b\rangle \xleftrightarrow{H} \left(|0\rangle + (-1)^b|1\rangle\right) = \left(|0\rangle + e^{i2\pi(0.b)}|1\rangle\right)$$

Obvious Phase Estimation Algorithm

$$\left(|0\rangle + e^{i2\pi(0.x_3)}|1\rangle\right)$$



$$|x_3\rangle$$

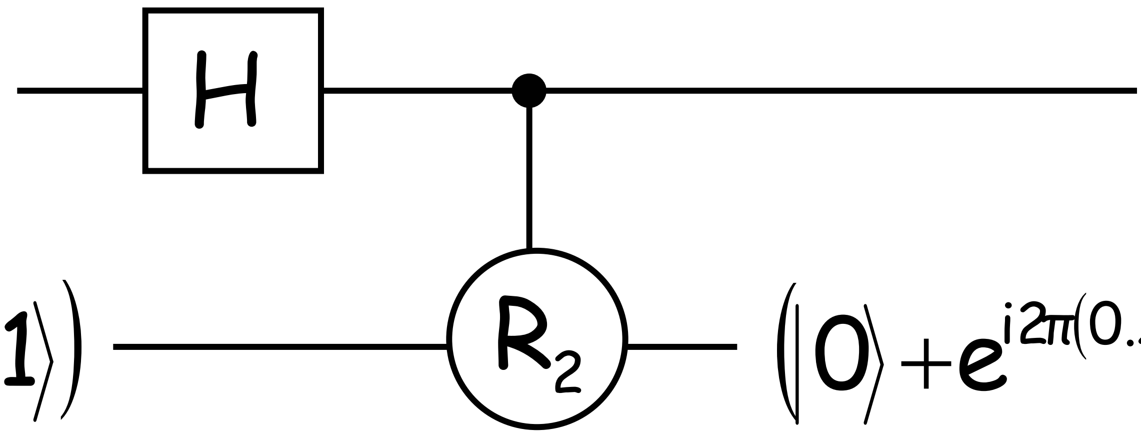
$$\left(|0\rangle + e^{i2\pi(0.x_2 x_3)}|1\rangle\right)$$

$$\left(|0\rangle + e^{i2\pi(0.x_1 x_2 x_3)}|1\rangle\right)$$

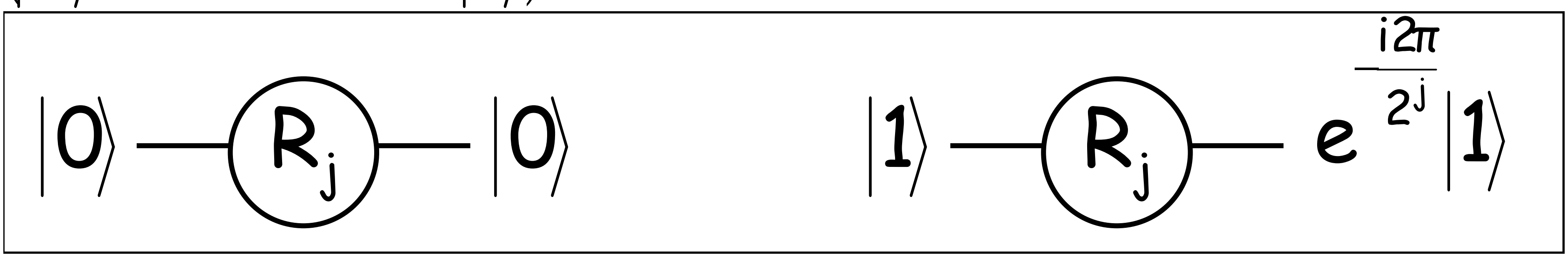
Phase Estimation

$$\left(|0\rangle + e^{i2\pi(0.x_3)}|1\rangle\right) \rightarrow |x_3\rangle$$

$$\left(|0\rangle + e^{i2\pi(0.x_2x_3)}|1\rangle\right) \rightarrow \left(|0\rangle + e^{i2\pi(0.x_2)}|1\rangle\right)$$



$$\left(|0\rangle + e^{i2\pi(0.x_1x_2x_3)}|1\rangle\right)$$

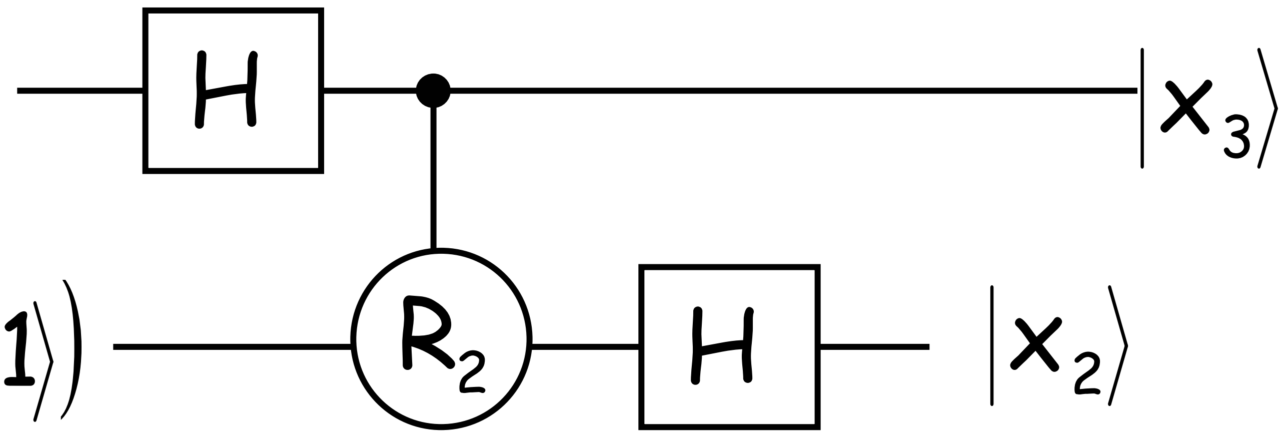


Natural Phase Estimation

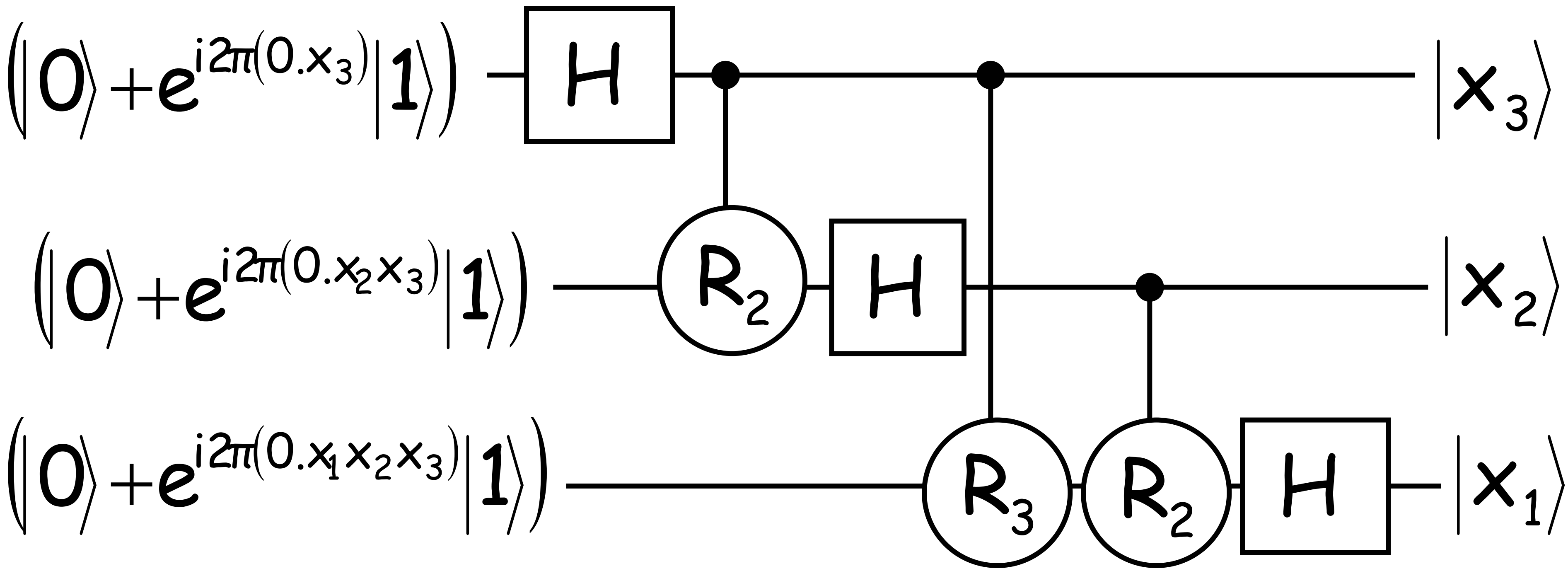
$$\left(|0\rangle + e^{i2\pi(0.x_3)}|1\rangle\right)$$

$$\left(|0\rangle + e^{i2\pi(0.x_2x_3)}|1\rangle\right)$$

$$\left(|0\rangle + e^{i2\pi(0.x_1x_2x_3)}|1\rangle\right)$$

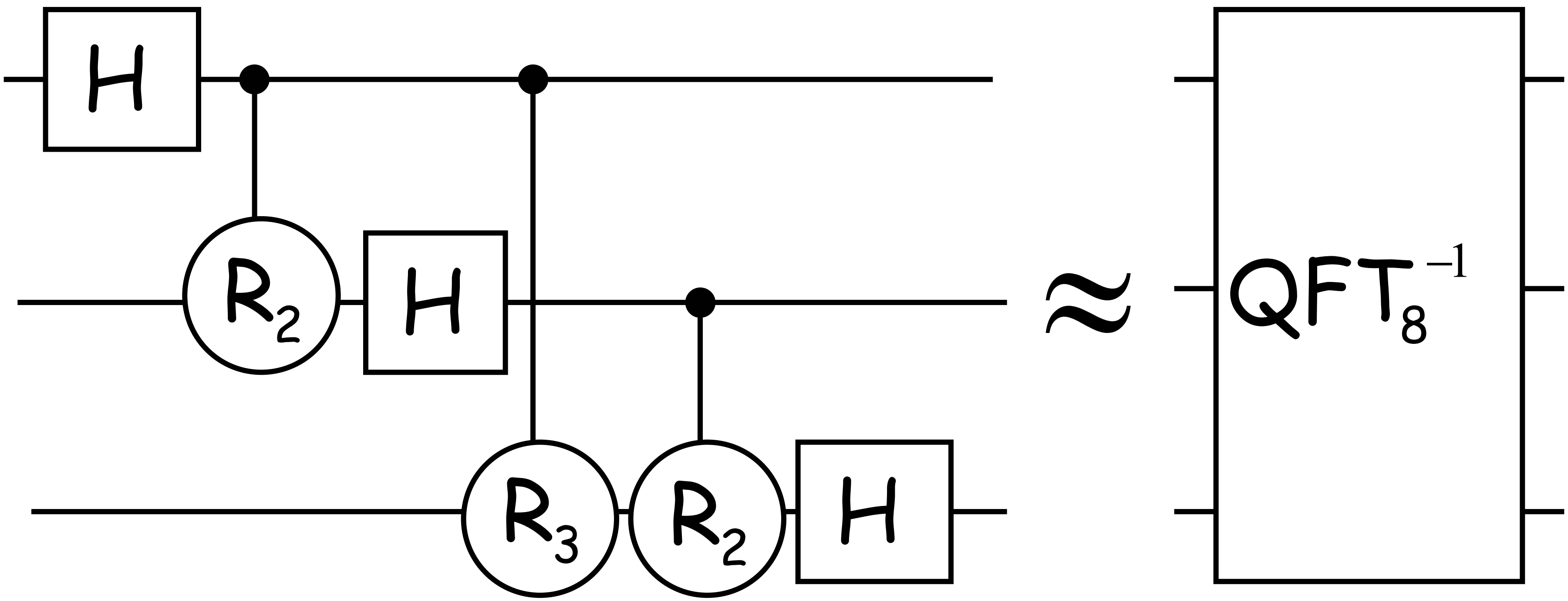


Phase Estimation



Inverse Quantum Fourier Transform

If we reorder the final qubits, we have



What is a (Q)FT?

$$FT_{2^n} : C^{2^n} \to C^{2^n}$$

$$e_j = (0,0,\cdots,1,\cdots,0,0)$$

$$\mapsto \frac{1}{\sqrt{2^n}}\left(1, e^{i2\pi\frac{j}{2^n}}, e^{i2\pi\left(2\frac{j}{2^n}\right)}, \ldots, e^{i2\pi\left((2^n-1)\frac{j}{2^n}\right)}\right)$$

$$FT_{2^n}^{-1} : \frac{1}{\sqrt{2^n}}\left(1, e^{i2\pi\frac{j}{2^n}}, e^{i2\pi\left(2\frac{j}{2^n}\right)}, \ldots, e^{i2\pi\left((2^n-1)\frac{j}{2^n}\right)}\right) \mapsto e_j$$

What is a (Q)FT?

$$FT_{2^n}^{-1}: \frac{1}{\sqrt{2^n}}\left(1, e^{i\varphi}, e^{i2\varphi}, \ldots, e^{i\left((2^n-1)\varphi\right)}\right)$$

$$\mapsto \left(a_0, a_1, \cdots, a_{2^n-1}\right)$$

$$\left|a_j\right| = \frac{\sin\left(2^n\left(\frac{\varphi}{2\pi} - \frac{j}{2^n}\right)\pi\right)}{2^n \sin\left(\left(\frac{\varphi}{2\pi} - \frac{j}{2^n}\right)\pi\right)}$$

What is a (Q)FT?

$$QFT_{2^n} : H_{2^n} \to H_{2^n}$$

$$|j\rangle \mapsto \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n} e^{i2\pi\left(x\frac{j}{2^n}\right)} |x\rangle$$

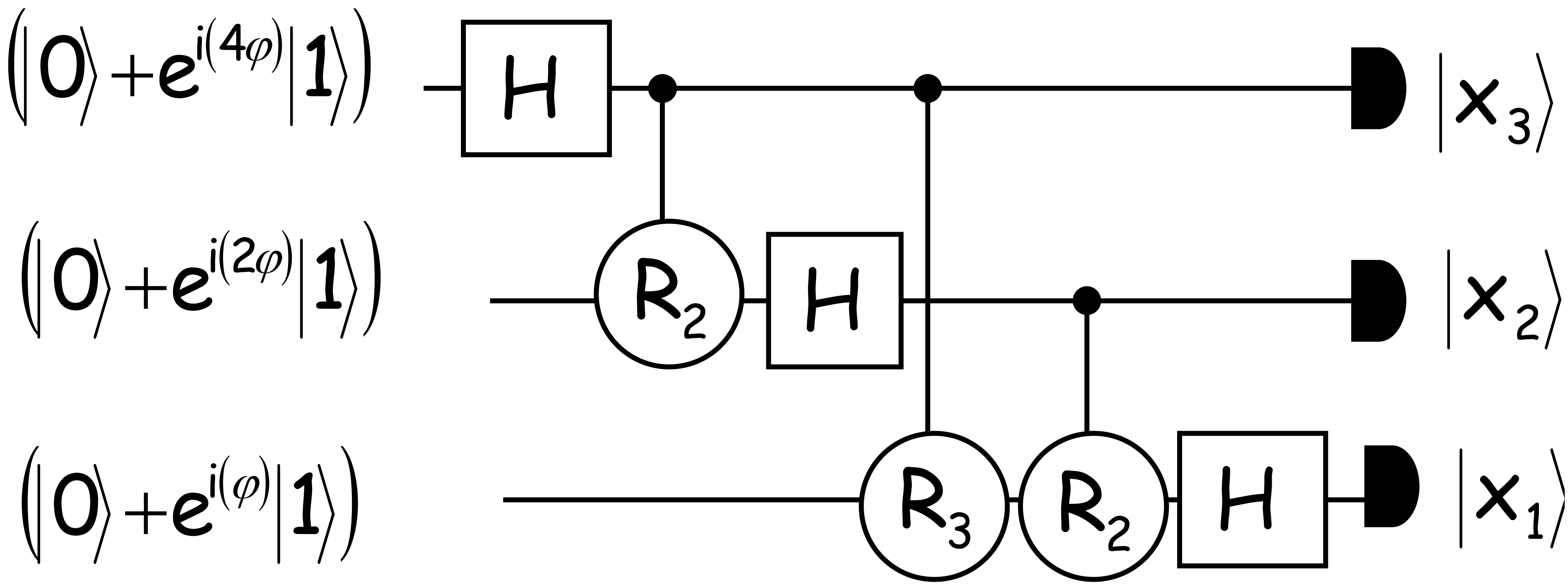
$$QFT_{2^n}^{-1} : \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n} e^{i2\pi\left(x\frac{j}{2^n}\right)} |x\rangle \mapsto |j\rangle$$

What is a (Q)FT?

$$QFT_{2^n}^{-1}: \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n} e^{ix\varphi} |x\rangle \mapsto \sum_j a_j |j\rangle$$

$$|a_j| = \frac{\sin\left(2^n\left(\frac{\varphi}{2\pi} - \frac{j}{2^n}\right)\pi\right)}{2^n \sin\left(\left(\frac{\varphi}{2\pi} - \frac{j}{2^n}\right)\pi\right)}$$

Phase Estimation: Arbitrary φ

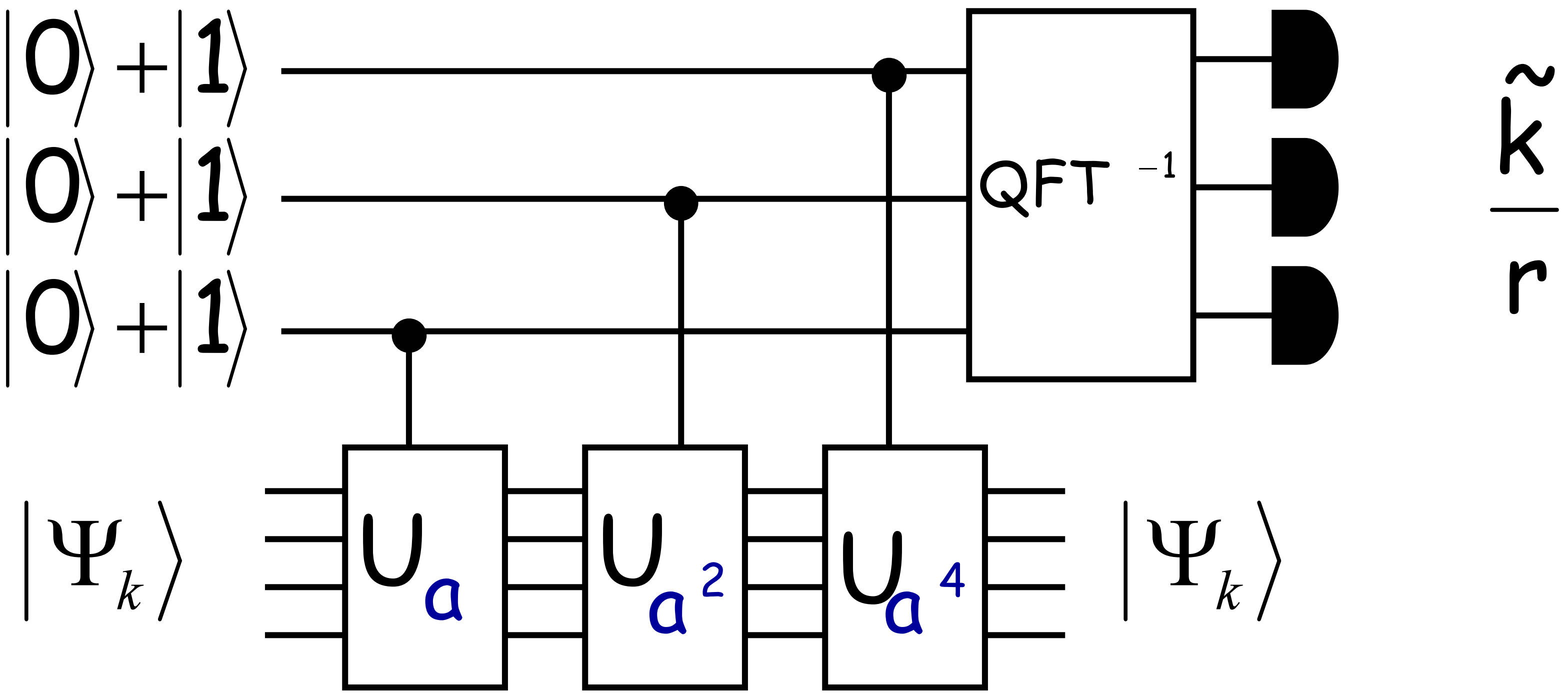


$$\tilde{\varphi} = 2\pi \frac{(4x_1 + 2x_2 + x_3)}{8}$$

$$P\left(\left| \frac{\tilde{\varphi}}{2\pi} - \frac{\varphi}{2\pi} \right| \leq \frac{1}{8} \right) \geq \frac{8}{\pi^2}$$

Quantum Factoring

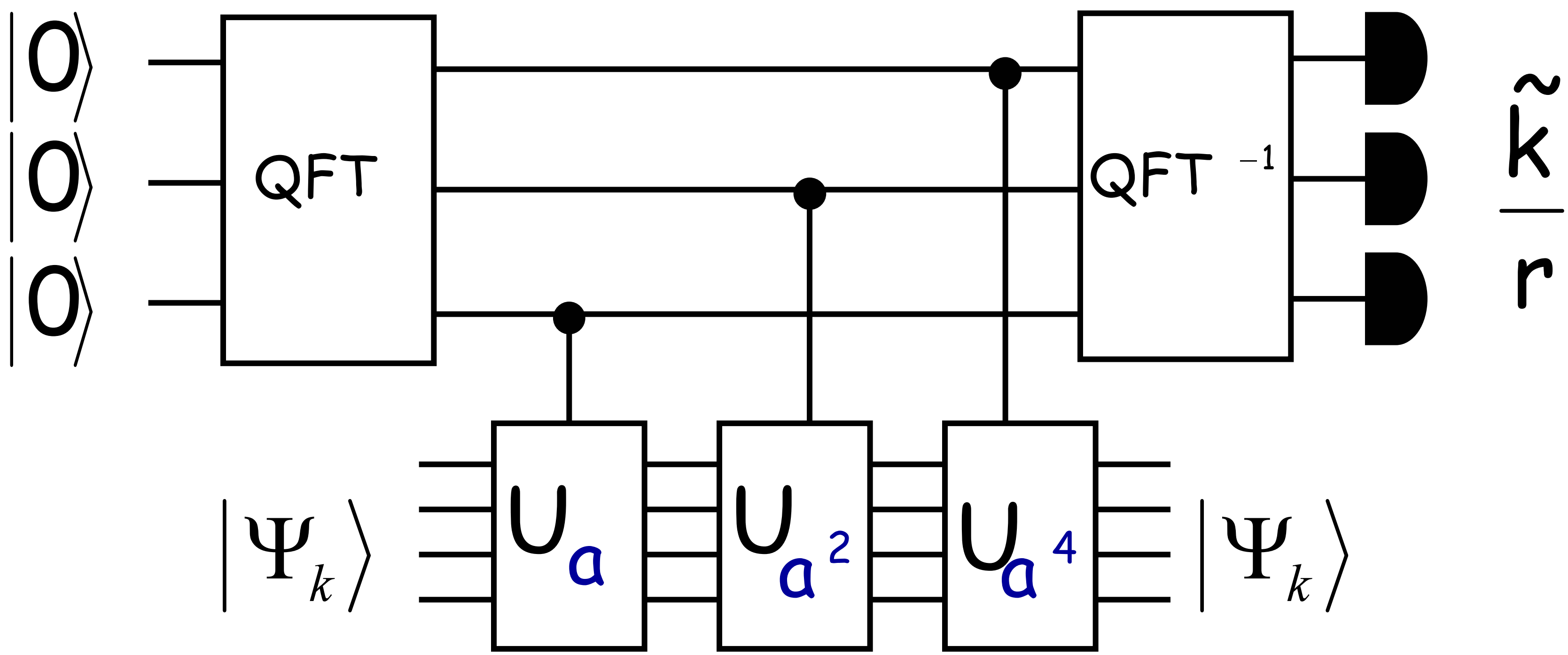
We can efficiently estimate $\frac{k}{r}$



[Kitaev95, CEMM98]

Factoring Network

We are effectively studying the behaviour of the controlled-U_a in a 'very quantum' basis.



Factoring Network

The given network maps

$$|000\rangle|\Psi_k\rangle \mapsto \left|\frac{\widetilde{k}}{r}\right\rangle|\Psi_k\rangle$$

And therefore

$$|000\rangle|1\rangle = \frac{1}{\sqrt{r}}\sum_k |000\rangle|\Psi_k\rangle \mapsto \frac{1}{\sqrt{r}}\sum_k \left|\frac{\widetilde{k}}{r}\right\rangle|\Psi_k\rangle$$

Partial measurements

$$\sum_k \frac{1}{\sqrt{r}} \left| \frac{\tilde{k}}{r} \right\rangle |\psi_k\rangle$$

What do we get when we measure the first register?
In general, we can rewrite

$$\sum_{xy} a_{xy} |x\rangle|y\rangle = \sum_x |x\rangle \left(\sum_y a_{xy} |y\rangle \right)$$

$$= \sum_x b_x |x\rangle |\Phi_x\rangle$$

$$|\Phi_x\rangle = \sum_y \frac{a_{xy}}{b_x} |y\rangle \qquad b_x = \sqrt{\sum_y |a_{xy}|^2}$$

Partial measurements

The probability of measuring x in the first register of $\sum_x b_x |x\rangle|\Phi_x\rangle$ is $b_x{}^2$

Partial measurements

Alternatively, we can rewrite

$$\sum_{xy} a_{xy} |x\rangle|y\rangle = \sum_{y}\left(\sum_{x} a_{xy} |x\rangle\right)|y\rangle$$

$$= \sum_{y} c_y \left|\Phi'_y\right\rangle|y\rangle$$

$$\left|\Phi'_y\right\rangle = \sum_{x} \frac{a_{xy}}{c_y}|x\rangle \qquad c_y = \sqrt{\sum_{x} \left|a_{xy}\right|^2}$$

Partial measurements

Measuring the first register of $\sum_y c_y |\Phi'_y\rangle |y\rangle$ is equivalent to performing a measurement on the state $|\Phi'_y\rangle$ with probability c_y^2

Partial measurements

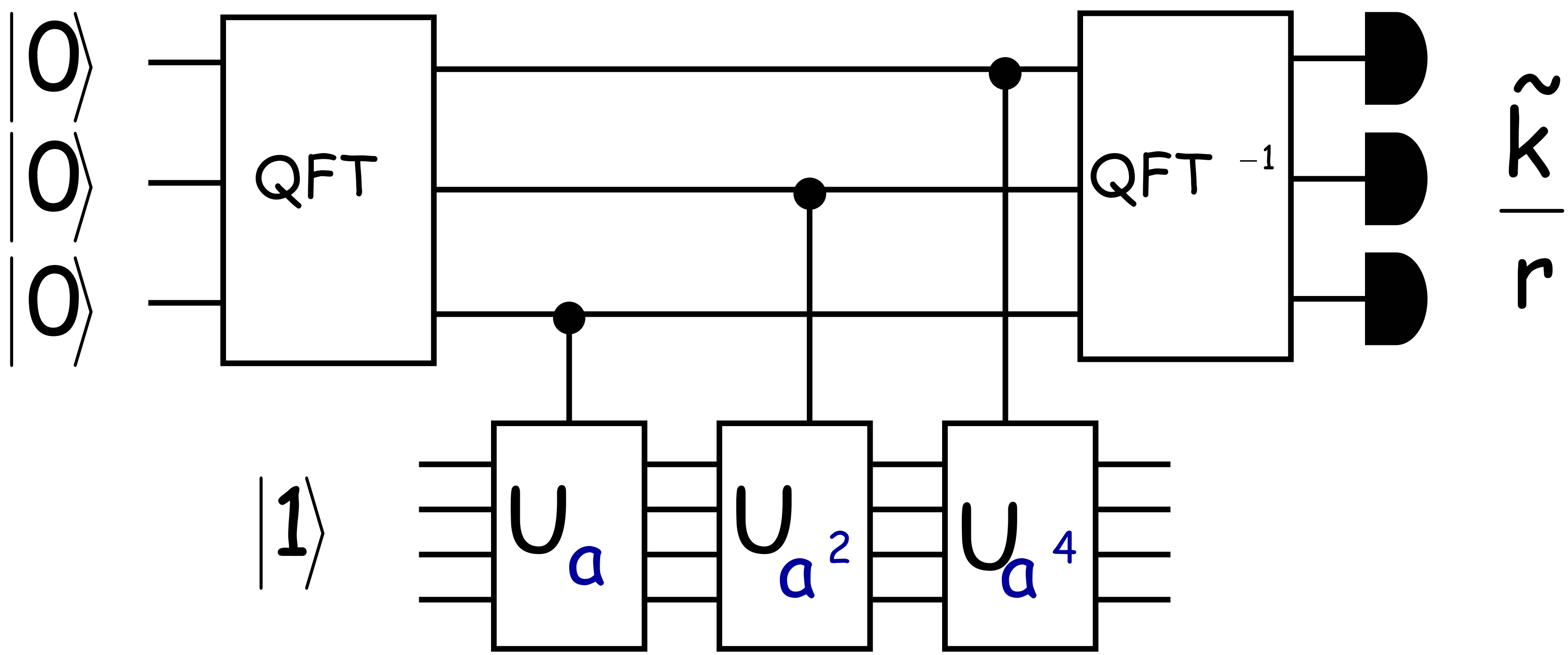
Measuring the first register of $\sum_k \frac{1}{\sqrt{r}} \left| \frac{\tilde{k}}{r} \right\rangle \left| \psi_k \right\rangle$

is equivalent to performing a measurement

on the state $\left| \frac{\tilde{k}}{r} \right\rangle$ with probability $\frac{1}{r}$

Factoring Network

We are effectively studying the behaviour of the controlled-U_a in a 'very quantum' basis.



[CEMM98] show this is equivalent to [Shor94]

Complexity comparison

The best rigorous classical algorithms use

$$e^{O(\sqrt{\log(N)\log\log(N)})}$$ operations

The best heuristic classical algorithms use

$$e^{O((\log(N))^{\frac{1}{3}}\log\log(N)^{\frac{2}{3}})}$$ operations

The quantum algorithm uses poly(log(N))

$$= e^{O(\log\log(N))}$$ operations

Hidden Subgroup

This approach allows us to solve efficiently any "Abelian Hidden Subgroup Problem" (see [ME98],[M99],[NC00])

$$f : G \to X$$

$$K \leq G$$

$$f(x) = f(y) \iff x - y \in K$$

Find K

Hidden Affine Functions

Hidden Affine Functions:

$$f : Z_p^n \rightarrow Z_p^m$$

$$x \rightarrow Mx + b$$

Find M using only m evaluations of f (instead of n+1) (D,BV,CEMM,H,M)