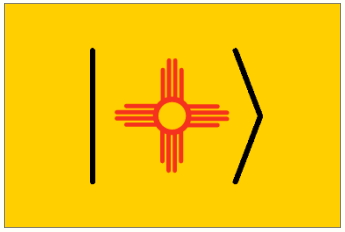
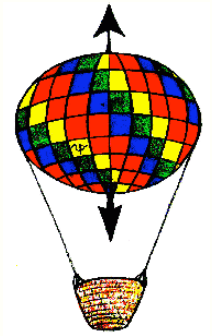


Short Course in Quantum Information Lecture 5

Quantum Algorithms



Prof. Andrew Landahl

University of New Mexico

Course Info

- All materials downloadable @ website

http://info.phys.unm.edu/~deutschgroup/DeutschClasses.html

- Syllabus

Lecture 1: Intro
Lecture 2: Formal Structure of Quantum Mechanics
Lecture 3: Entanglement
Lecture 4: Qubits and Quantum Circuits
Lecture 5: Algorithms
Lecture 6: Error Correction
Lecture 7: Physical Implementations
Lecture 8: Quantum Cryptography

Course Info

- All materials downloadable @ website

http://info.phys.unm.edu/~deutschgroup/DeutschClasses.html

- Syllabus

Lecture 1: Intro
Lecture 2: Formal Structure of Quantum Mechanics
Lecture 3: Entanglement
Lecture 4: Qubits and Quantum Circuits
Lecture 5: Algorithms
Lecture 6: Error Correction
Lecture 7: Physical Implementations
Lecture 8: Quantum Cryptography

Qubits and Quantum Gates: Review

Qubits:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad |\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

$$|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \qquad |\alpha|^2 + |\beta|^2 = 1$$

Rays in Hilbert space

Gates:

$$U = R_{\hat{n}}(\theta) = e^{-i\theta(\hat{n}\cdot\sigma)/2}$$

$$\hat{n}\cdot\sigma = n_x\sigma_x + n_y\sigma_y + n_z\sigma_z$$

Pauli matrices: $\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ $\sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$ $\sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$

Qubits and Quantum Gates: Review

Quantum circuit notation:

$$-\boxed{R_{\hat{n}}(\theta)}- \qquad ---------- = -R_x(2\pi) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$-\boxed{X}- = iR_x(\pi) \qquad -\boxed{H}- = R_y(\frac{\pi}{2}) = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

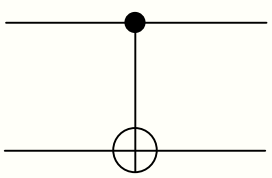
$$-\boxed{Y}- = iR_y(\pi) \qquad -\boxed{P}- = e^{i\pi/4}R_z(\frac{\pi}{2}) = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$$

$$-\boxed{Z}- = iR_z(\pi) \qquad -\boxed{T}- = R_z(\frac{\pi}{4}) = \begin{pmatrix} e^{-i\pi/8} & 0 \\ 0 & e^{i\pi/8} \end{pmatrix}$$

Qubits and Quantum Gates: Review

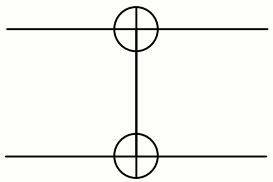
More gates:

CNOT

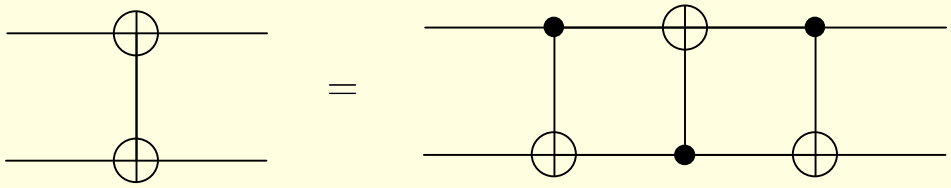


$$= \begin{matrix} |00\rangle|01\rangle|10\rangle|11\rangle \\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \\ \end{matrix} \begin{matrix} |00\rangle \\ |01\rangle \\ |10\rangle \\ |11\rangle \end{matrix}$$

SWAP



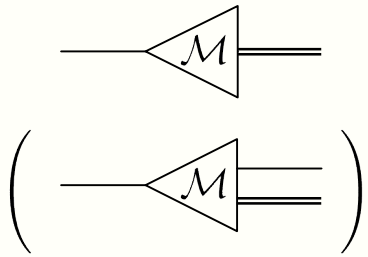
$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



C/C++ code:

```
a ^= b ^= a ^= b
```

Measure

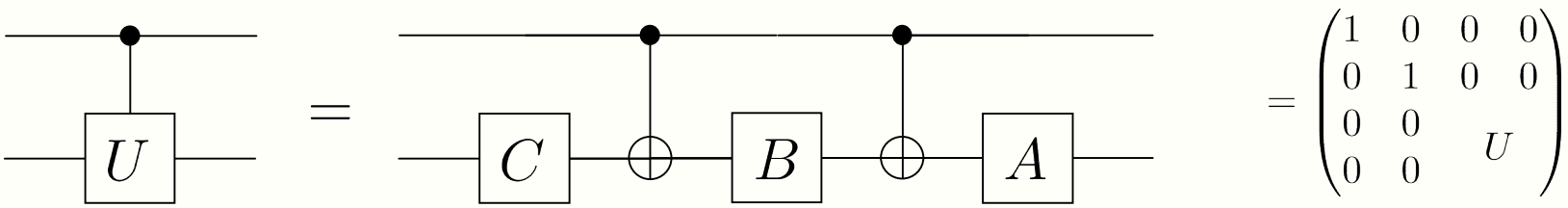


$$= \begin{cases} \dfrac{1}{|\alpha|} \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} & p = |\alpha|^2 \\ \dfrac{1}{|\beta|} \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} & p = |\beta|^2. \end{cases}$$

Qubits and Quantum Gates: Review

Controlled-U: $U = R_z(\alpha)R_y(\beta)R_z(\gamma)$

C_U



$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & & \\ 0 & 0 & & U \end{pmatrix}$$

$$A = R_z(\alpha)R_y(\beta/2)$$
$$B = R_y(-\beta/2)R_z(-(\alpha+\gamma)/2)$$
$$C = R_z((\gamma-\alpha)/2)$$

$$ABC = I$$
$$AXBXC = R_z(\alpha)R_y(\beta)R_z(\gamma)$$

Qubits and Quantum Gates

Universal gate bases:

Any $U \in SU(n)$ can be approximated to arbitrary precision by ε using $\mathcal{O}(f(n) \cdot 1/\varepsilon^c)$ gates from the basis.

Solovay-Kitaev Theorem proves that only $\mathcal{O}(f(n) \cdot \log^c 1/\varepsilon)$ gates are needed for this precision.

Examples:

$\{CNOT, \{U\}_1, |0\rangle, \mathcal{M}\}$ Simple: Good for building quantum algorithms.

$\{CNOT, H, T, |0\rangle, \mathcal{M}\}$ Discrete: Good for robust/computable implementations.

$\{U_{2,\text{generic}}, |0\rangle, \mathcal{M}\}$ Abstract: Good for existence proofs.

Other interesting universal gate sets exist, e.g., measurement-and-state only sets. *Current research area!*

Quantum Algorithms

Why this field exists.

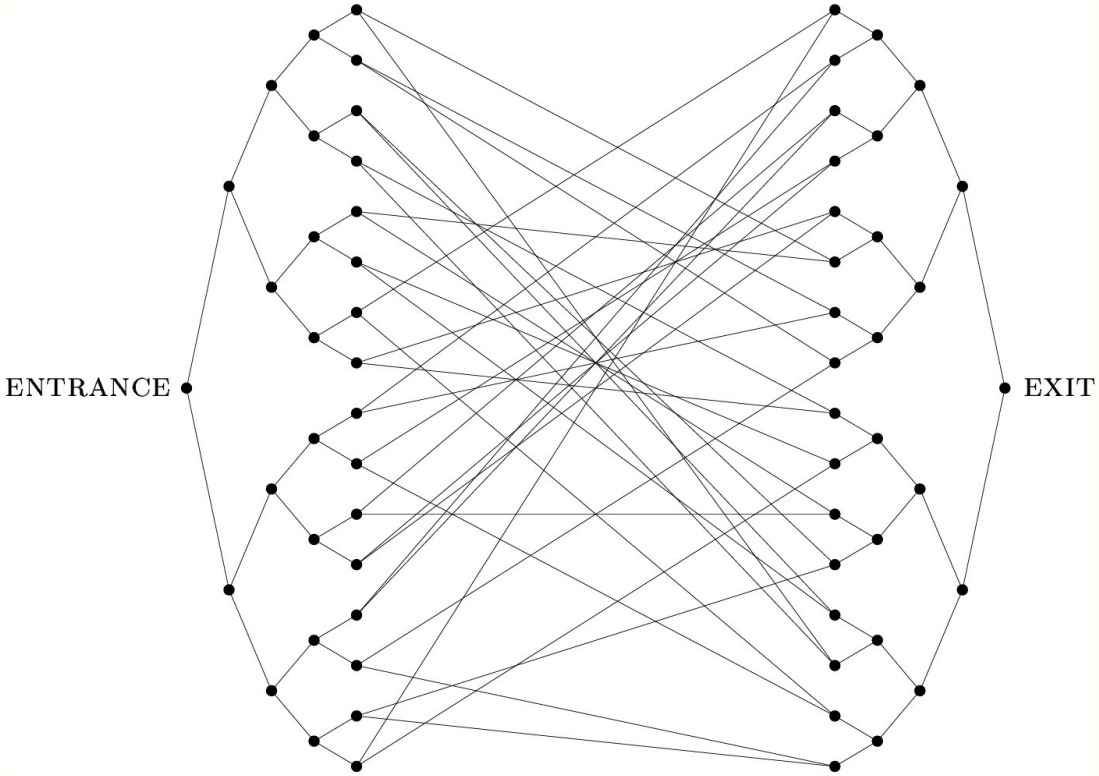
Approximate counting
Bernstein-Vazarani problem
Collision problem
Deutsch-Jozsa problem
Discrete logarithm
Element distinctness
Gauss sum approximation
Gradient estimation
Hidden shift problem
Hidden subgroup problem
Integer factoring
Jones polynomial evaluation
Matrix commutativity testing
Matrix multiplication verification
Maze solving
Mean estimation
Median estimation
Mode estimation
Order finding
Ordered search
Local Hamiltonian simulation
Parity evaluation
Pell's equation
Period finding
Phase estimation
Shifted Legendre symbol problem
Simon's problem
Sparse Hamiltonian simulation
Spatial search
Triangle finding
Unordered search

$$AB = C?$$

Quantum Algorithms

Why this field exists.

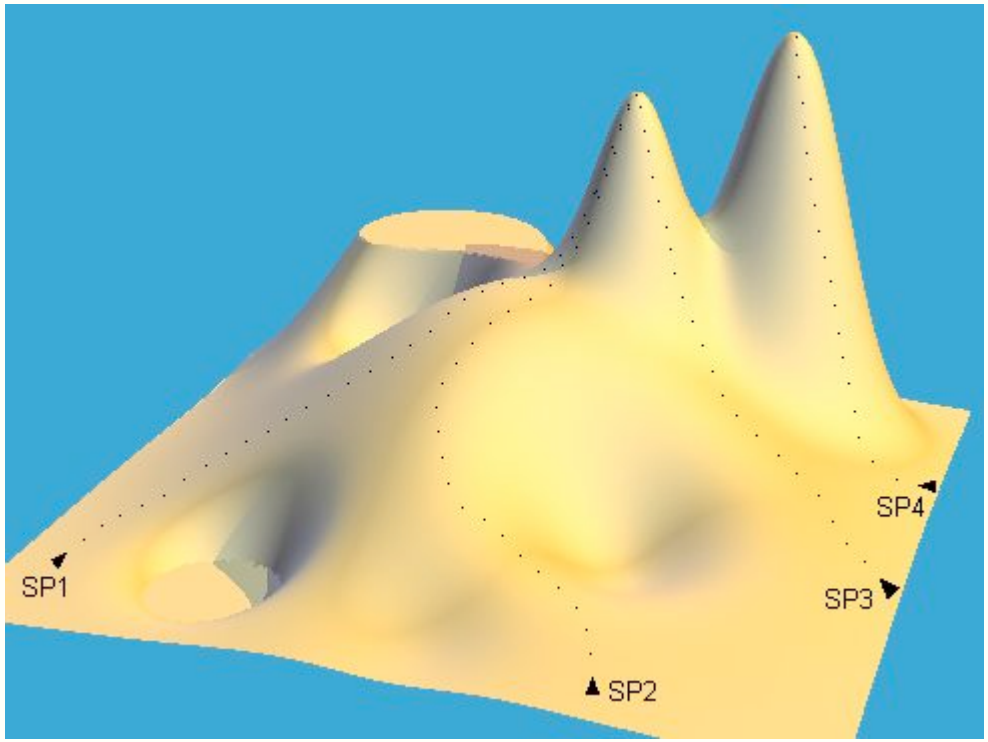
Approximate counting
Bernstein-Vazarani problem
Collision problem
Deutsch-Jozsa problem
Discrete logarithm
Element distinctness
Gauss sum approximation
Gradient estimation
Hidden shift problem
Hidden subgroup problem
Integer factoring
Jones polynomial evaluation
Matrix commutativity testing
Matrix multiplication verification
Maze solving
Mean estimation
Median estimation
Mode estimation
Order finding
Ordered search
Local Hamiltonian simulation
Parity evaluation
Pell's equation
Period finding
Phase estimation
Shifted Legendre symbol problem
Simon's problem
Sparse Hamiltonian simulation
Spatial search
Triangle finding
Unordered search



Quantum Algorithms

Why this field exists.

Approximate counting
Bernstein-Vazarani problem
Collision problem
Deutsch-Jozsa problem
Discrete logarithm
Element distinctness
Gauss sum approximation
Gradient estimation
Hidden shift problem
Hidden subgroup problem
Integer factoring
Jones polynomial evaluation
Matrix commutativity testing
Matrix multiplication verification
Maze solving
Mean estimation
Median estimation
Mode estimation
Order finding
Ordered search
Local Hamiltonian simulation
Parity evaluation
Pell's equation
Period finding
Phase estimation
Shifted Legendre symbol problem
Simon's problem
Sparse Hamiltonian simulation
Spatial search
Triangle finding
Unordered search



Quantum Algorithms

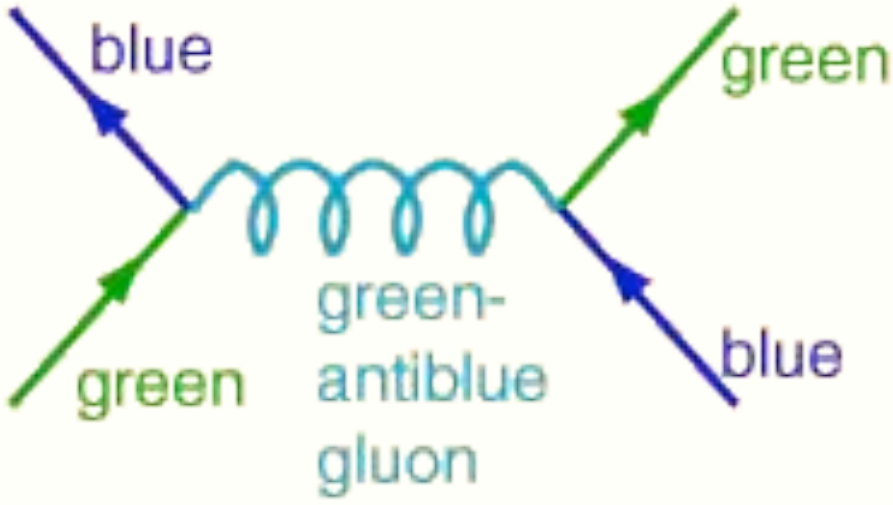
Why this field exists.

Approximate counting
Bernstein-Vazarani problem
Collision problem
Deutsch-Jozsa problem
Discrete logarithm
Element distinctness
Gauss sum approximation
Gradient estimation
Hidden shift problem
Hidden subgroup problem
Integer factoring
Jones polynomial evaluation
Matrix commutativity testing
Matrix multiplication verification
Maze solving
Mean estimation
Median estimation
Mode estimation
Order finding
Ordered search
Local Hamiltonian simulation
Parity evaluation
Pell's equation
Period finding
Phase estimation
Shifted Legendre symbol problem
Simon's problem
Sparse Hamiltonian simulation
Spatial search
Triangle finding
Unordered search

Quantum Algorithms

Why this field exists.

Approximate counting
Bernstein-Vazarani problem
Collision problem
Deutsch-Jozsa problem
Discrete logarithm
Element distinctness
Gauss sum approximation
Gradient estimation
Hidden shift problem
Hidden subgroup problem
Integer factoring
Jones polynomial evaluation
Matrix commutativity testing
Matrix multiplication verification
Maze solving
Mean estimation
Median estimation
Mode estimation
Order finding
Ordered search
Local Hamiltonian simulation
Parity evaluation
Pell's equation
Period finding
Phase estimation
Shifted Legendre symbol problem
Simon's problem
Sparse Hamiltonian simulation
Spatial search
Triangle finding
Unordered search



Feynman diagram for an interaction between quarks generated by a gluon.

Quantum Algorithms

Why this field exists.

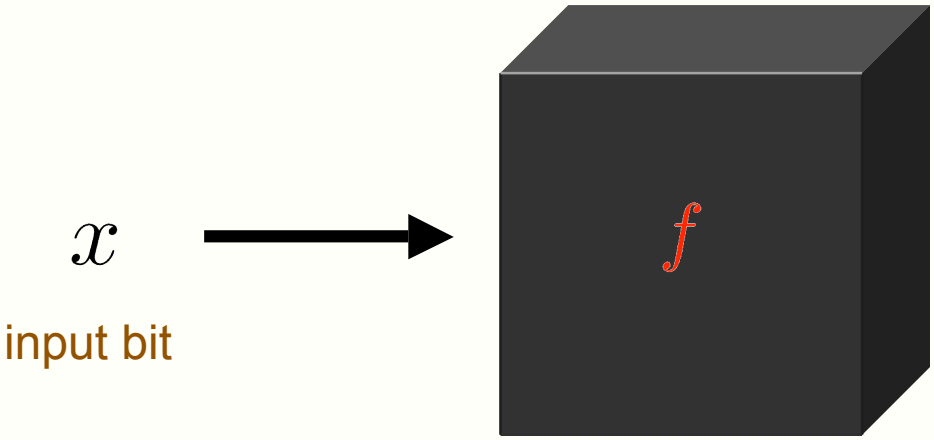
Approximate counting
Bernstein-Vazarani problem
Collision problem
Deutsch-Jozsa problem
Discrete logarithm
Element distinctness
Gauss sum approximation
Gradient estimation
Hidden shift problem
Hidden subgroup problem
Integer factoring
Jones polynomial evaluation
Matrix commutativity testing
Matrix multiplication verification
Maze solving
Mean estimation
Median estimation
Mode estimation
Order finding
Ordered search
Local Hamiltonian simulation
Parity evaluation
Pell's equation
Period finding
Phase estimation
Shifted Legendre symbol problem
Simon's problem
Sparse Hamiltonian simulation
Spatial search
Triangle finding
Unordered search

$$91 = 7 \cdot 13$$

(David) Deutsch's Problem

$\neq$

[D. Deutsch, 1985]



input bit

output bit

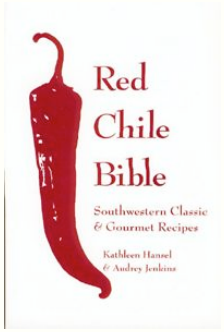
Question: Is f constant or balanced?

x	f_1	f_2	f_3	f_4
0	0	0	1	1
1	0	1	0	1

Deutsch's Problem

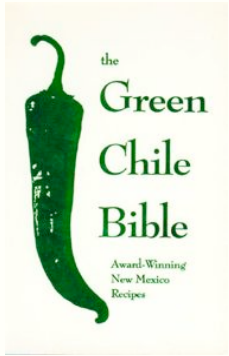
$x = 0$

$f(0) = 0$



$x =$ David or Ivan

$f =$ Red or Green



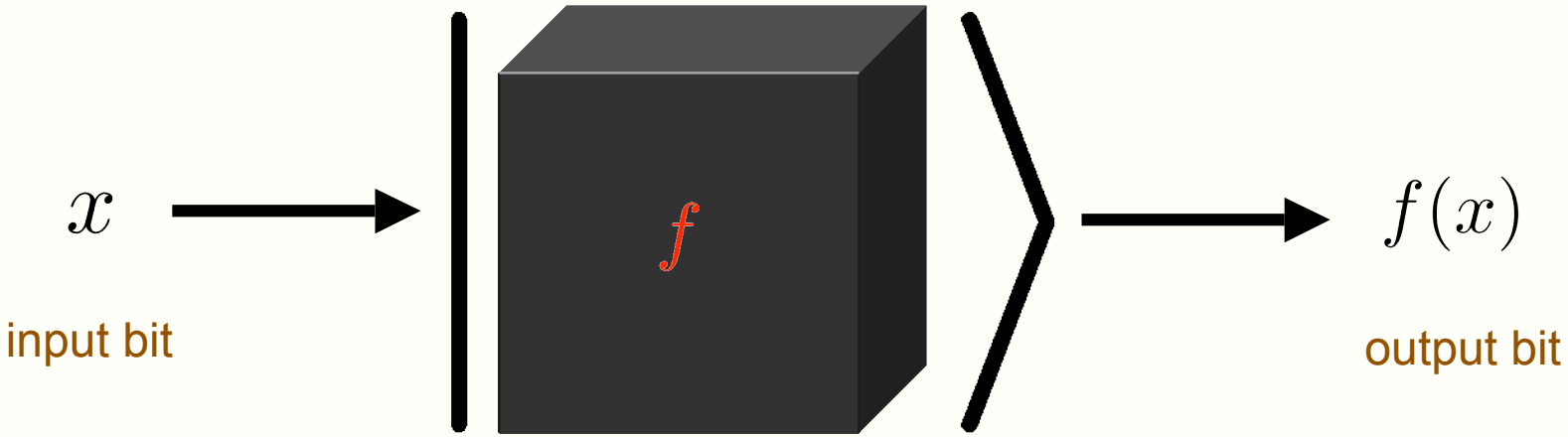
$x = 1$

$f(1) = 1$

Example: Do the Deutsch's like the same kind of chile or not?

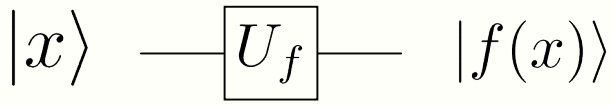
Problem: I only have time to query the function once.

Solution: Use a quantum black box!



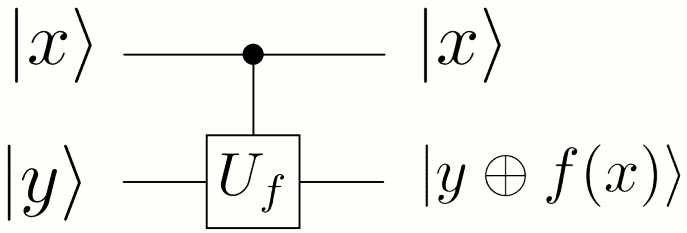
Deutsch's Problem

First attempt:



$$f(0) = f(1) \Rightarrow \text{Not unitary (noninvertible)}$$

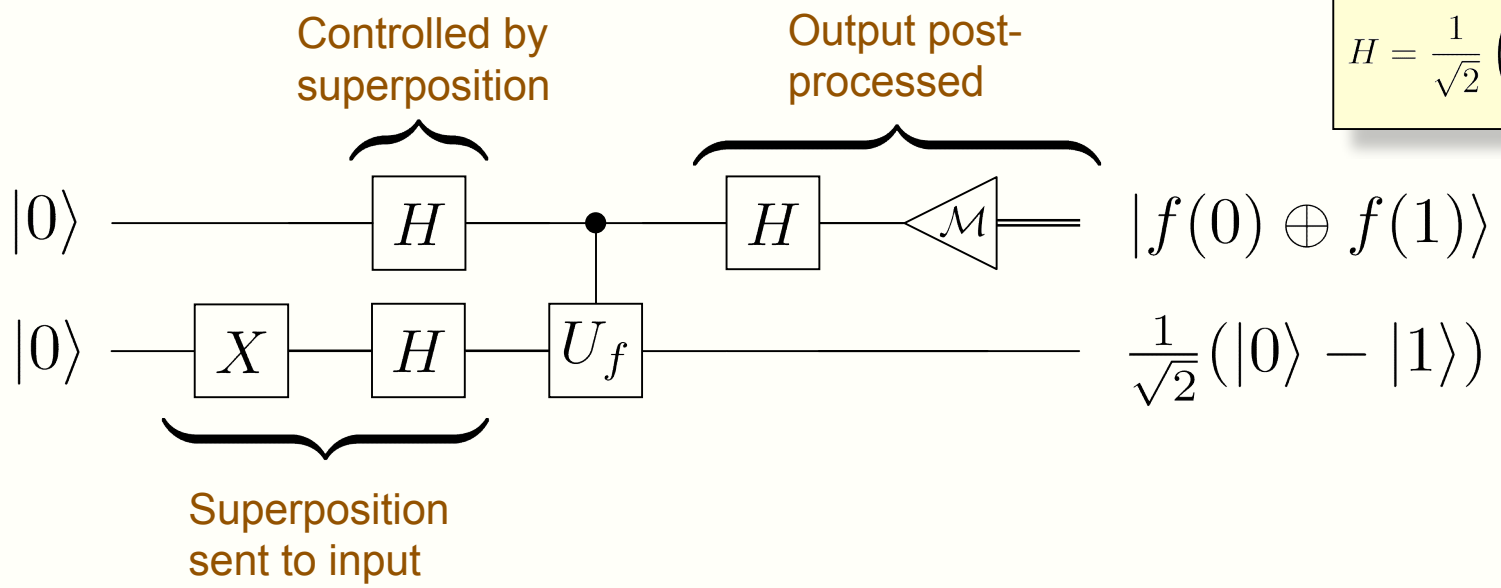
Second attempt:



Aha!

(David) Deutsch's Algorithm

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$



$$|0\rangle|0\rangle \mapsto |0\rangle|1\rangle$$

$$\mapsto \tfrac{1}{2}(|0\rangle + |1\rangle)(|0\rangle - |1\rangle)$$

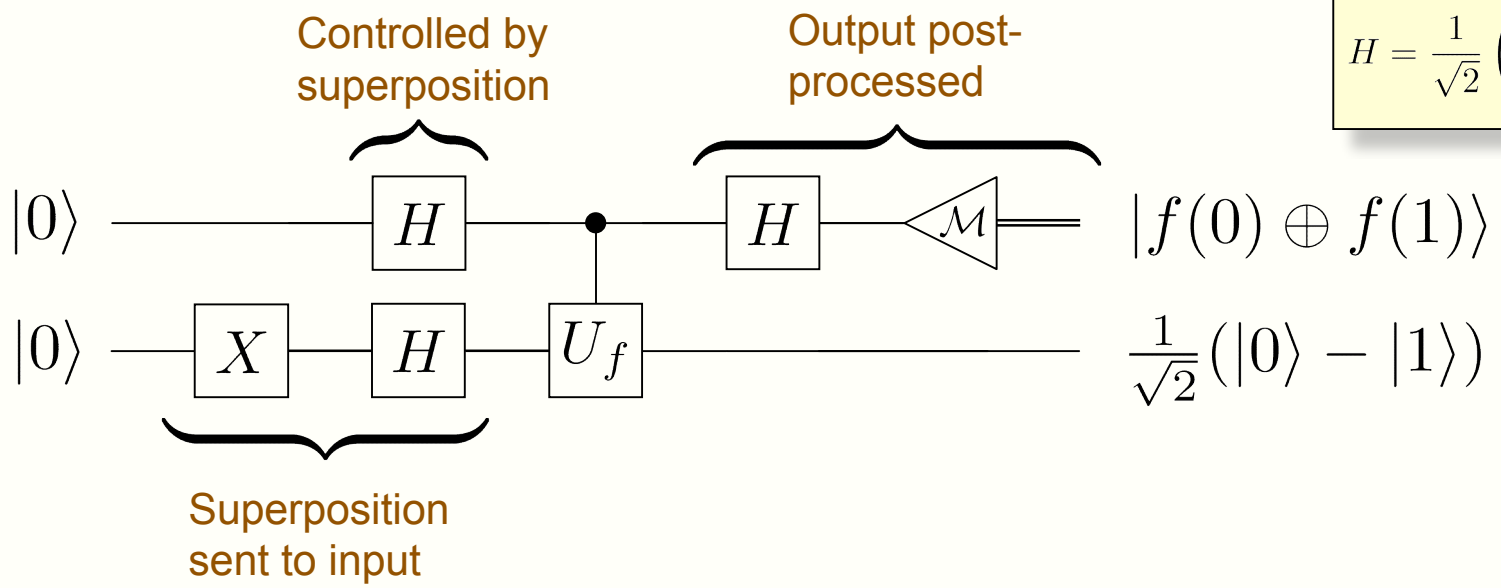
$$\mapsto \tfrac{1}{2}|0\rangle(|f(0)\rangle - |1 \oplus f(0)\rangle) + \tfrac{1}{2}|1\rangle(|f(1)\rangle - |1 \oplus f(1)\rangle)$$

$$= \tfrac{1}{2}(-1)^{f(0)}|0\rangle(|0\rangle - |1\rangle) + \tfrac{1}{2}(-1)^{f(1)}|1\rangle(|0\rangle - |1\rangle)$$

$$= (-1)^{f(0)}\tfrac{1}{2}(|0\rangle + (-1)^{f(0)\oplus f(1)}|1\rangle)(|0\rangle - |1\rangle)$$

(David) Deutsch's Algorithm

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$



$$\tfrac{1}{2}(|0\rangle + (-1)^{f(0)\oplus f(1)}|1\rangle)(|0\rangle - |1\rangle)$$

$$\mapsto \tfrac{1}{2^{3/2}}[(1 + (-1)^{f(0)\oplus f(1)})|0\rangle + (1 - (-1)^{f(0)\oplus f(1)})|1\rangle)](|0\rangle - |1\rangle)$$

$$= \begin{cases} |0\rangle\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) & \text{if } f(0) \oplus f(1) = 0 \\ |1\rangle\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) & \text{if } f(0) \oplus f(1) = 1. \end{cases}$$

Quantum transforms

What is the Hadamard transform doing?

$$H|b\rangle = \frac{1}{\sqrt{2}} \sum_{z=0}^{1} (-1)^{bz} |z\rangle$$

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

J. Hadamard
(1865-1963)

$$H^{\otimes n}|b_1, \ldots, b_n\rangle = \frac{1}{\sqrt{2^n}} \sum_{z_1,\ldots,z_n} (-1)^{b_1 z_1 + \cdots + b_n z_n} |z_1, \ldots, z_n\rangle$$

$$H^{\otimes n}|j\rangle = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} (-1)^{j \cdot k} |k\rangle$$

$$\tilde{x}_j = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} (-1)^{j \cdot k} x_k$$

Walsh-Hadamard Transform

(Fourier transform in square waves)

n steps: $2^n \times 2^n$ matrix transform

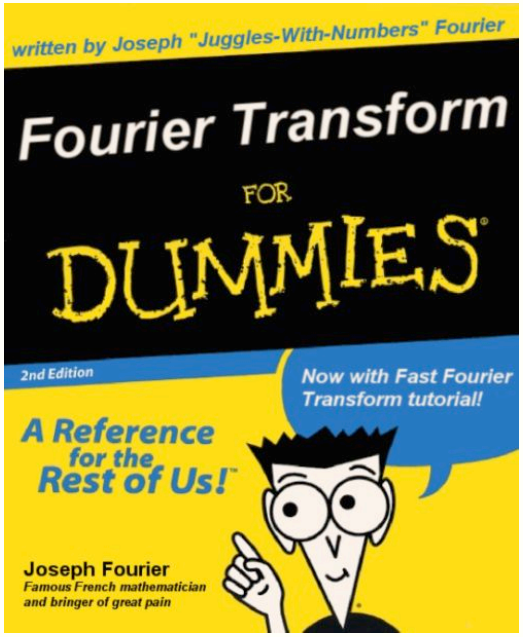
Quantum transforms

Discrete Fourier transform:

$$\tilde{x}_j \mapsto \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{2\pi i j k / 2^n} x_k$$

Naïve Fourier Transform: $\mathcal{O}(2^{2n})$

Fast Fourier Transform: $\mathcal{O}(n2^n)$



http://jan.moesen.nu/media/images/fun/

J. Fourier
(1768–1830)

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

Quantum Fourier Transform: $\mathcal{O}(n^2)$

Caveat: QFT is in the *amplitudes*.

Fast Fourier transform

$$\tilde{x}_j \mapsto \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{2\pi i j k/2^n} x_k$$

Must multiply j by k for 2^n values of k.

Fast Fourier Transform:

$$j = j_{n-1} \ldots j_0$$
$$k = k_{n-1} \ldots k_0$$

Binary expansion.:

$$\frac{jk}{2^n} = k_{n-1}(0.j_0) + k_{n-2}(0.j_1 j_0) + \cdots + k_0(0.j_{n-1} \ldots j_0)$$

n multiplications for 2^n terms in sum: $\mathcal{O}(n2^n)$

Example: $n = 3,\ j = 2,\ k = 3$

$$\begin{aligned} \frac{2 \cdot 3}{2^3} &= 0(0.1) + 1(0.10) + 1(0.010) \\ &= 0.11 \\ &= 1/2 + 1/4 \\ &= 3/4 \end{aligned}$$

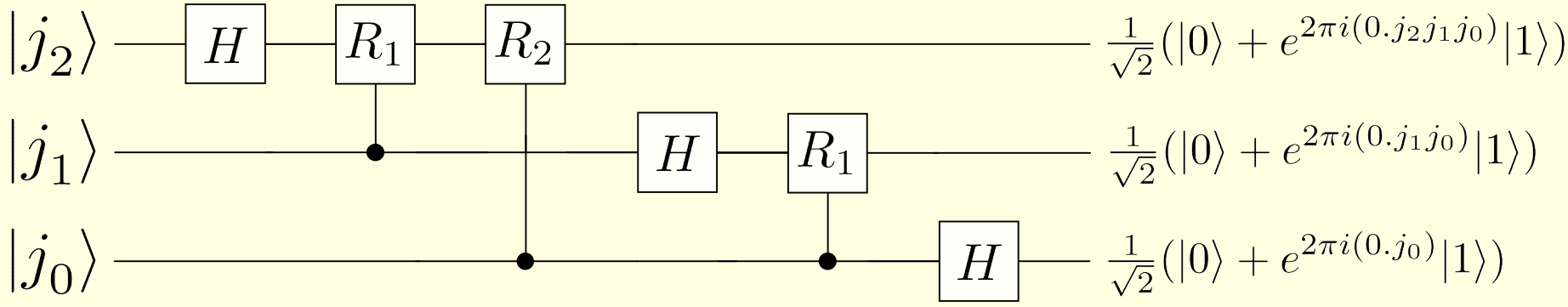
Quantum Fourier transform

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

$$= \frac{1}{\sqrt{2^n}} (|0\rangle + e^{2\pi i (0.j_0)} |1\rangle) \cdots (|0\rangle + e^{2\pi i (0.j_{n-1} \ldots j_0)} |1\rangle)$$

$$R_k \equiv e^{i\pi/2^{k+1}} R_z\left(\tfrac{\pi}{2^k}\right) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/2^k} \end{pmatrix}$$

Example: $n = 3$



N.B. It is possible to modify the circuit to use only single-qubit gates with adaptive computation.

Quantum Fourier transform

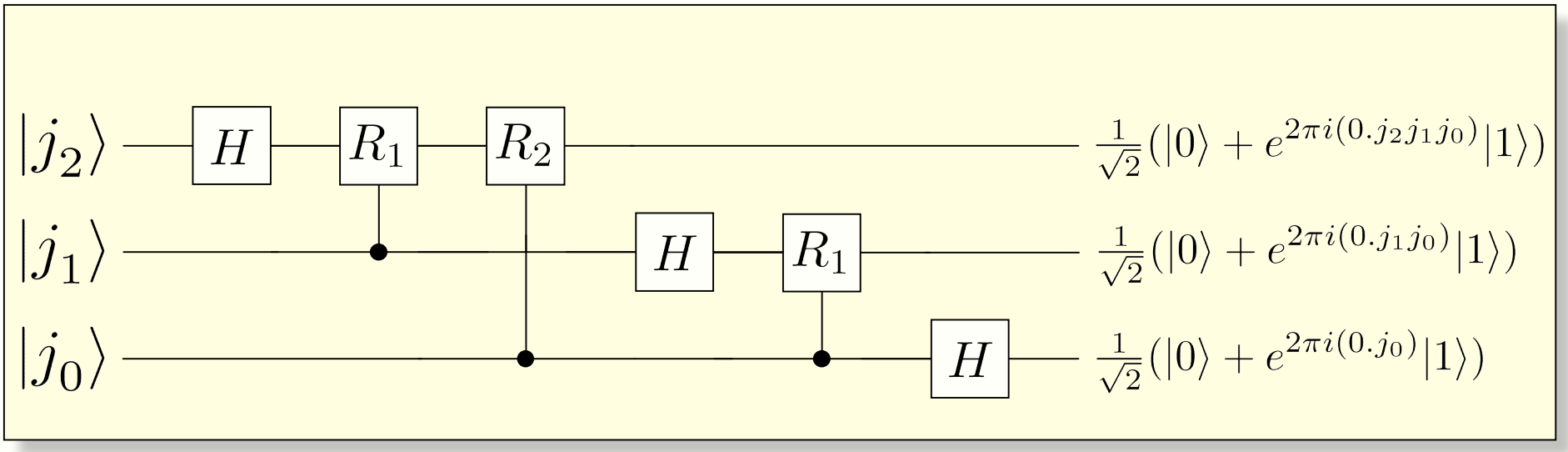
Complexity of QFT:

n Hadamard gates

$\frac{n(n-1)}{2}$ Controlled-R_k gates

$\frac{n}{2}$ SWAP gates

$\Big\}$ $\mathcal{O}(n^2)$ QFT "algorithm"



Phase estimation "algorithm"

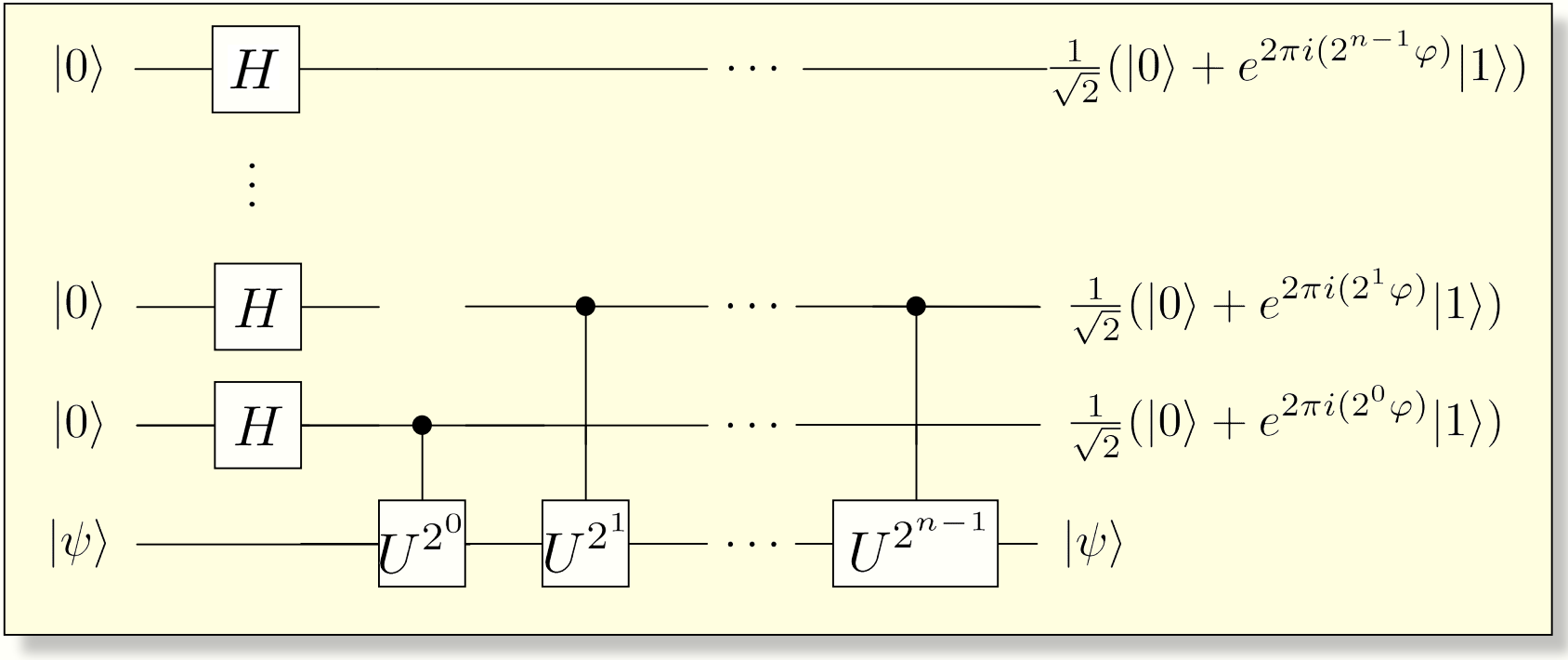
Given: Controlled-U^{2^k} gates, $C_{U^{2^k}}, k \in \{1, \ldots, n\}$

Eigenstate $|\psi\rangle$ of U such that $U|\psi\rangle = e^{2\pi i\varphi}|\psi\rangle$.

Problem: Estimate φ to n bits of precision.

A. Kitaev

Solution: Phase 1



Phase estimation "algorithm"

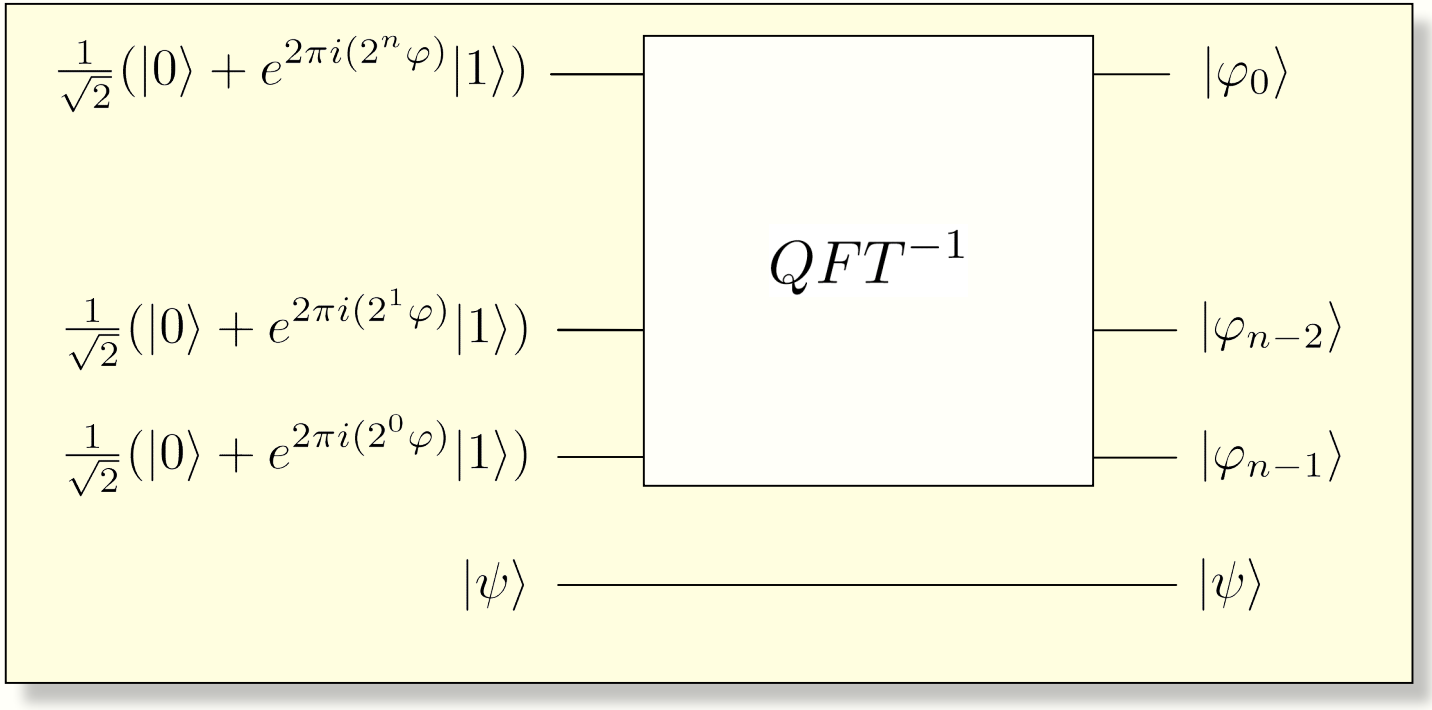
Given: Controlled-U^{2^k} gates, $C_{U^{2^k}}, k \in \{1, \ldots, n\}$

Eigenstate $|\psi\rangle$ of U such that $U|\psi\rangle = e^{2\pi i\varphi}|\psi\rangle$.

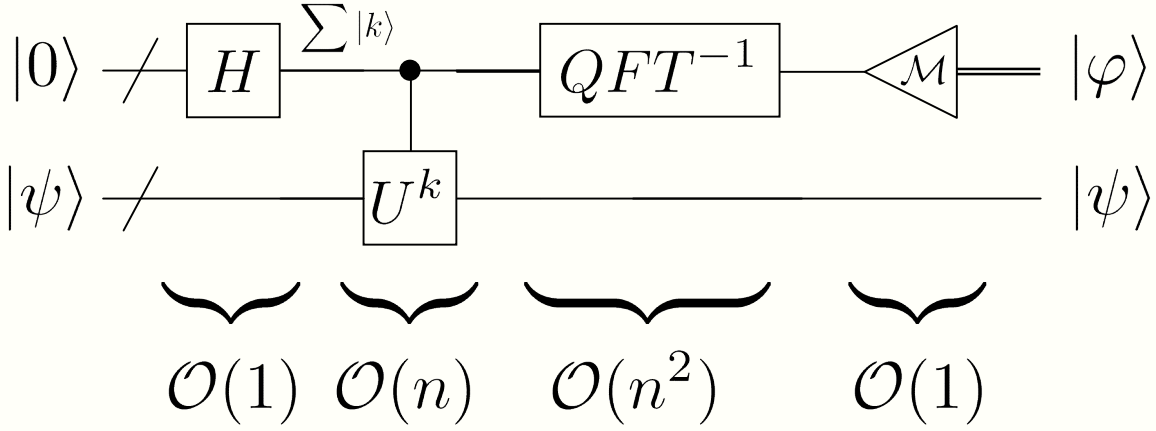
Problem: Estimate φ to n bits of precision.

Solution: Phase 2

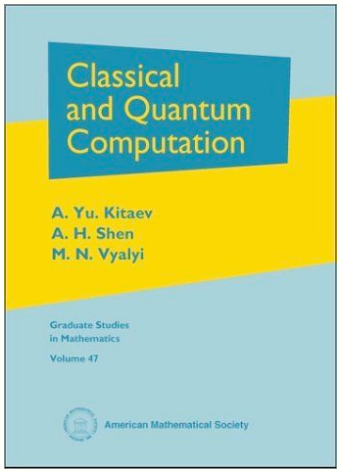
A. Kitaev

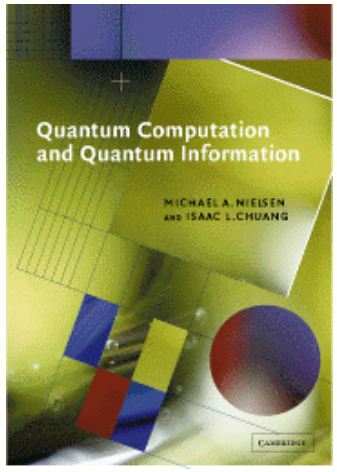


Phase estimation "algorithm": Analysis



Caveat: If φ is not exactly n bits, error analysis is subtle.
For details see:





Factoring

"The problem of distinguishing prime numbers from composite numbers and of ***resolving the latter into their prime factors*** is known to be one of the most important and useful in arithmetic. […] Further, the ***dignity of science itself*** seems to demand that every possible means be explored for the solution of a problem so elegant and so celebrated."

—Carl Friedrich Gauss

Disquisitiones Arithmeticæ, 1801

(translation: A.A. Clarke)

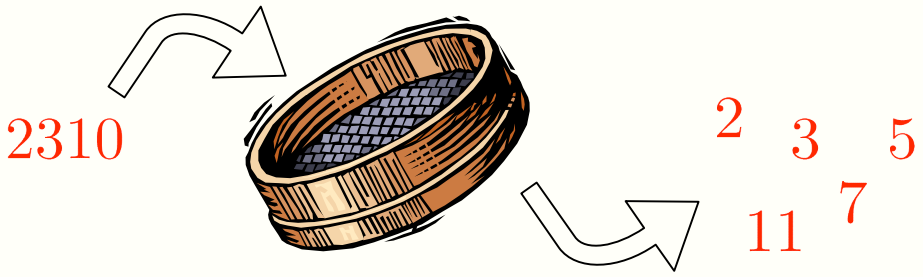
Factoring Problem

Input: Positive integer N.
Output: Positive integer $p > 1$ that divides N, if N is a composite number.

$$n \equiv \log N$$

Best-known classical algorithm:

Number field sieve [Pollard, 1988] $\exp\left[n^{1/3} \cdot \log^{2/3} n \cdot \mathcal{O}(1)\right]$



Best-known quantum algorithm:

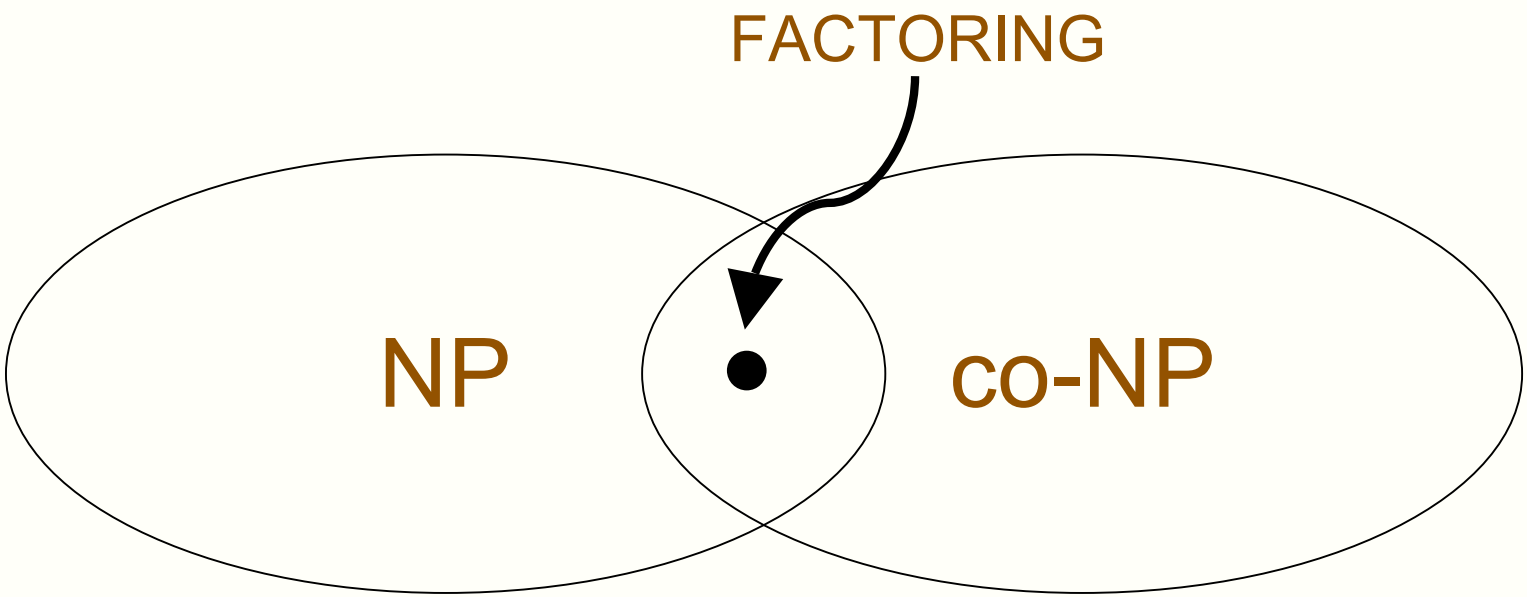
Shor's factoring algorithm [Shor, 1994] $\mathcal{O}(n^3)$

N.B. Very parallelizable!

$\mathcal{O}(n^5 \log^2 n)$-sized circuit with $\mathcal{O}(\log n)$ depth. [Cleve & Watrous, 2000]

Factoring in Theory

FACTORING
Input: Positive integers N and $k < N$.
Output: Does N have a factor less than k ?



Can verify YES or NO efficiently with a witness

[Agarwal, Kayal, Saxena, 2002]

Factoring in Practice

General Purpose Factoring Records - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://www.crypto-world.com/FactorRecords.html

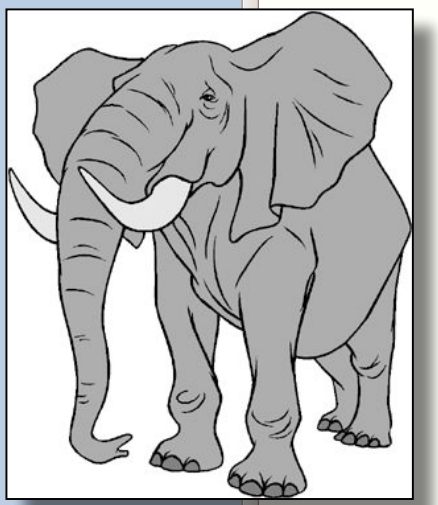
Getting Started Latest Headlines

General Purpose Factoring Records

Below is a chart of general purpose factoring records going back to 1990. By "general purpose", we mean a factoring algorithms whose running time is dependent upon only the size of the number being factored (i.e. not on the size of the prime factors or any particular form of the number).

Sieving is typically the dominant factorization run time in practice. All sieving times below are approximate. Early versions of factoring records estimated time in MIPS years, which is the number of years it would take a computer that operates at one million instructions per second to factor the number. More recently, almost everybody uses Pentiums or AMDs. Thus, we scale some timings to Pentium 1GHz CPU years: the number of years it would take a 1GHz Pentium (or AMD) to complete sieving.

number	digits	date completed	sieving time	algorithm
C116	116	1990	275 MIPS years	mpqs
RSA-120	120	June, 1993	830 MIPS years	mpqs
RSA-129	129	April, 1994	5000 MIPS years	mpqs
RSA-130	130	April, 1996	1000 MIPS years	gnfs
RSA-140	140	February, 1999	2000 MIPS years	gnfs
RSA-155	155	August, 1999	8000 MIPS years	gnfs
C158	158	January, 2002	3.4 Pentium 1GHz CPU years	gnfs
RSA-160	160	March, 2003	2.7 Pentium 1GHz CPU years	gnfs
RSA-576	174	December, 2003	13.2 Pentium 1GHz CPU years	gnfs
C176	176	May, 2005	48.6 Pentium 1GHz CPU years	gnfs
RSA-200	200	May, 2005	121 Pentium 1GHz CPU years [*]	gnfs



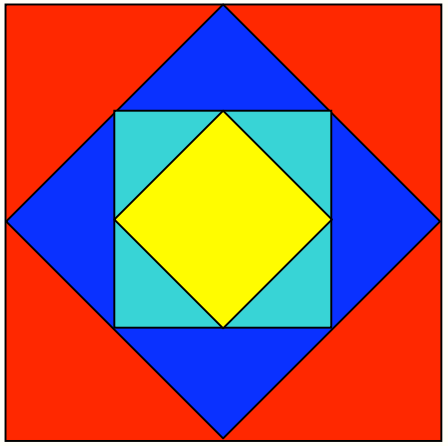
[*] In regard to RSA-200, Thorsten Kleinjung writes: *we spend 25% of the total time for the matrix step. If one considers the total time we spent about 170 CPU years.*

For more factorization results, please see the Factorization Announcements page or Paul Zimmermann's Factor Records page.

Done

Factoring algorithms

Difference-of-squares technique:



Random x, y such that

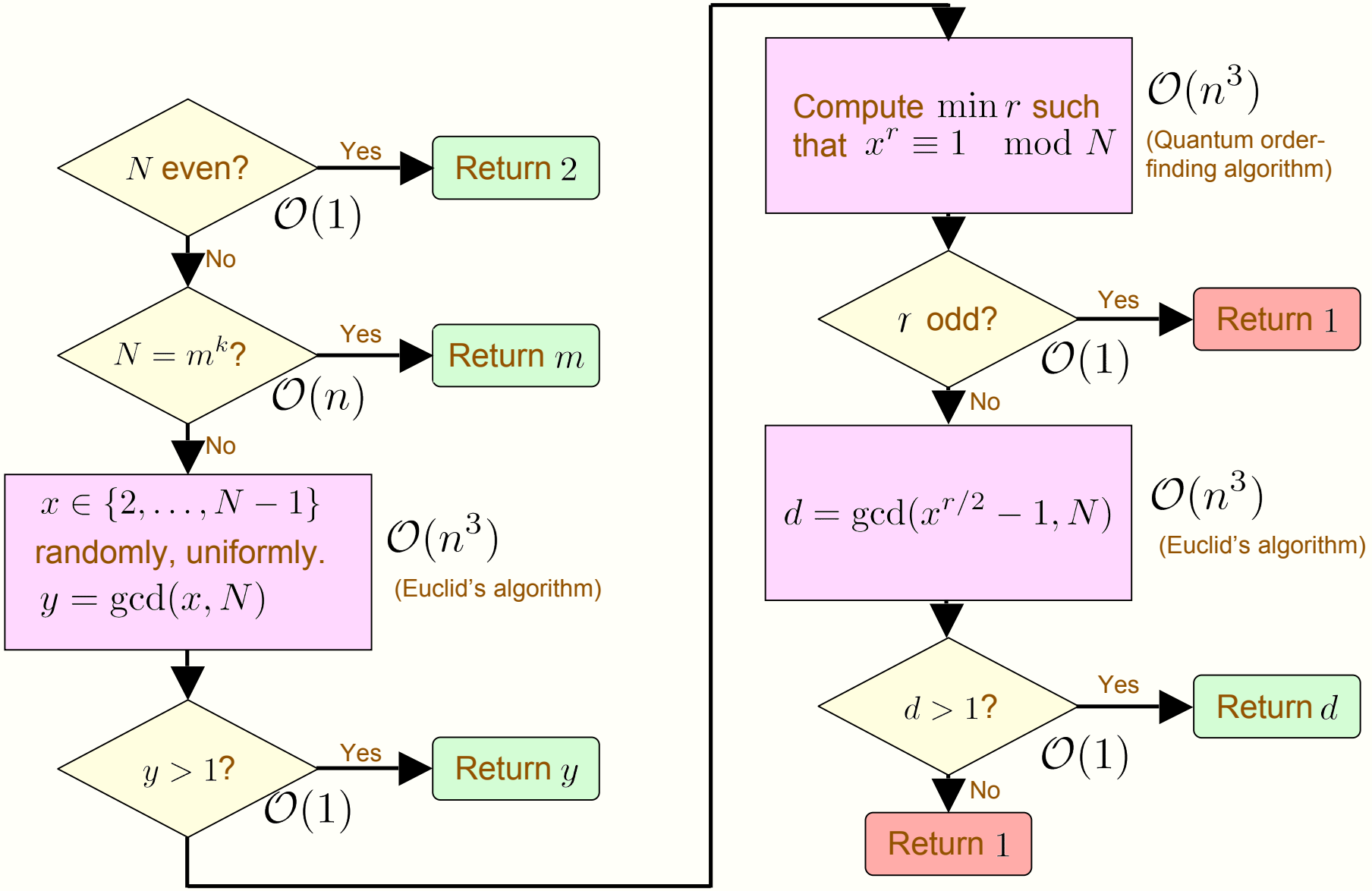
$$x^2 \equiv y^2 \mod N$$

N divides $x^2 - y^2 = (x+y)(x-y)$

Hope that $\gcd(x-y, N) > 1.$

Art: Choosing x, y wisely.

Factoring in five easy steps [Miller, 1976]



Factoring from order finding: Analysis

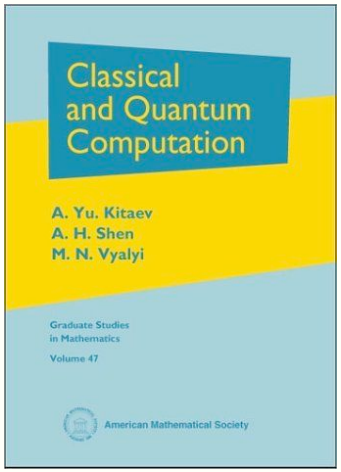
Only way to fail is if order is odd or an even order doesn't yield a nontrivial gcd.

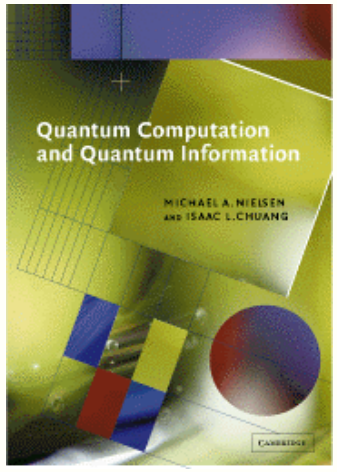
$$N = p_1^{a_1} \cdots p_k^{a_k}$$

$$\Pr[r \text{ is odd or } r \text{ is even and } d = 1] = \left(\frac{1}{2}\right)^{k-1}$$

$$\leq \frac{1}{2}$$

Proof: Uses *Chinese Remainder theorem*. See one of the below for details:





Factoring from order finding: Example

$N = 15 = 3 \cdot 5$ [Vandersypen *et al.*, 2001]

1. N even? No.
2. $N = m^k$? No.
3. $y = \gcd(x, N)$. $x \in \{2, \cancel{3}, 4, \cancel{5}, 6, 7, 8, \cancel{9}, \cancel{10}, 11, \cancel{12}, 13, 14\}$

   $x \in \{2, 4, 7, 8, 11, 13, 14\}$ have $y = 1$.
4. $x^r \equiv 1 \mod N$. $x^r = \{2^4, 4^2, 7^4, 8^4, 11^2, 13^4, 14^2\}$

   All possible orders are even.
5. $d = \gcd(x^{r/2} - 1, N) = \{3, 3, 3, 3, 5, 3, 1\}$.

Every possibility but one yields a nontrivial factor of N.

Order finding from phase estimation

Consider the unitary operator

$$U_{x,N}|y\rangle = |xy \mod N\rangle$$

Some of its eigenstates are of the form

$$|\psi_j\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi i j k/r} |x^k \mod N\rangle,$$

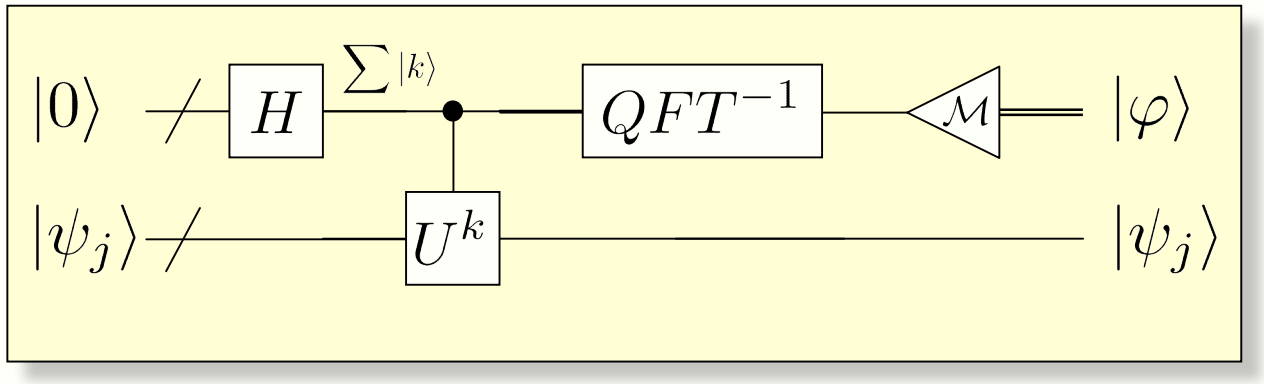
where r is the order of x in $\mathbb{Z}_N$.

Why? Because

$$\begin{aligned} U_{x,N}|\psi_j\rangle &= \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi i j k/r} |x^{k+1} \mod N\rangle \\ &= e^{2\pi i j/r} |\psi_j\rangle \end{aligned}$$

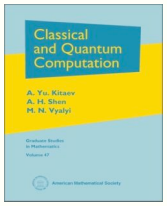
Order finding from phase estimation

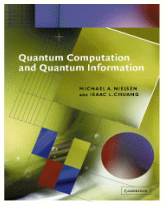
We can construct $C_{U_{x,N}^{2^k}}$ using the $\mathcal{O}(\lceil \log N \rceil^3)$ *modular exponentiation algorithm*. Given $|\psi_j\rangle$, we can then run the phase estimation algorithm to find $\varphi = j/r$ to $2n = 2\lceil \log N \rceil$ bits of precision:



Given this, we can use the $\mathcal{O}(n^3)$ *continued fractions algorithm* to obtain j/r in irreducible form exactly because j and r are bounded by N.

For details on the classical algorithms MEA and CFA, see:





Order finding from phase estimation

Given j/r in irreducible form j'/r' for several values of j, we can find r with high probability: $(r = \mathrm{lcm}(r'_1, r'_2)$ if j'_1 and j'_2 are coprime.)

Any prime p divides $1/p$ of all numbers.

$$\Rightarrow \Pr[p \text{ divides both } j'_1 \text{ and } j'_2] = 1/p^2$$

j'_1 and j'_2 are coprime iff there is no prime p that divides both.

$$\begin{aligned}
\Rightarrow \Pr[j'_1 \text{ and } j'_2 \text{ are coprime}] &= \prod_{\text{prime } p} \left(1 - \frac{1}{p^2}\right) \\
&= \frac{1}{\zeta(2)} \\
&= \frac{6}{\pi^2} \\
&\cong 0.607
\end{aligned}$$

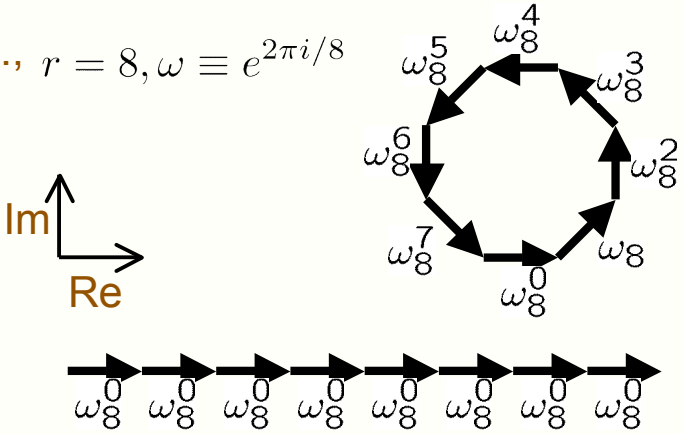
Order finding from phase estimation

We're almost done. All we need is a way to generate eigenstates $|\psi_j\rangle$ at random and we can efficiently factor integers!

Using the identity

$$\sum_{k=0}^{r-1} e^{-2\pi ijk/r} = r\delta_{j0}$$

E.g., $r = 8, \omega \equiv e^{2\pi i/8}$



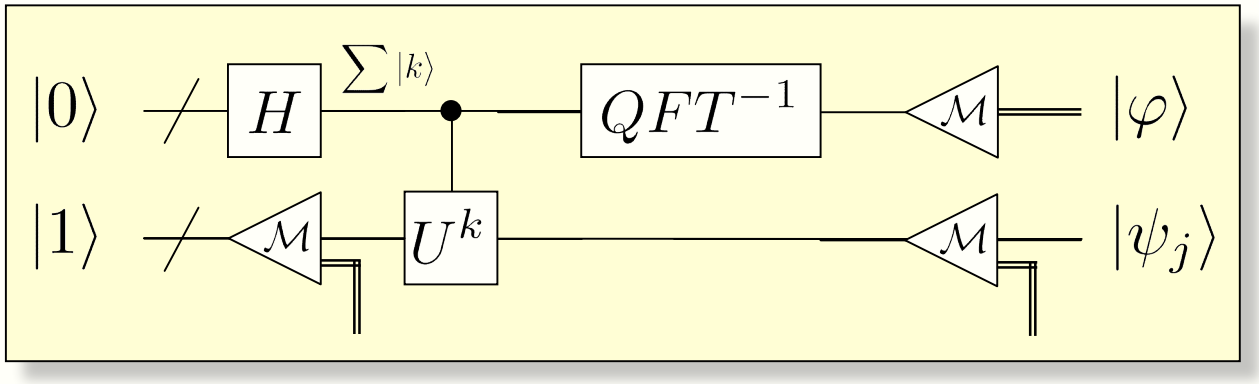
We know that

$$\frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} |\psi_k\rangle = |1\rangle$$

$$|\psi_j\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi ijk/r} |x^k \mod N\rangle,$$

Order finding from phase estimation

We could measure $|1\rangle$ in the $|\psi_j\rangle$ basis and run the phase estimation algorithm, but since this measurement commutes with U^{2^k}, we can measure at the end of the circuit, and indeed, we can omit measuring altogether:



Factoring: Summary

- Quantum Fourier transform ⇒ Phase estimation algorithm

- Phase estimation algorithm +
  Continued fractions algorithm +
  Modular exponentiation algorithm ⇒ Order finding algortihm

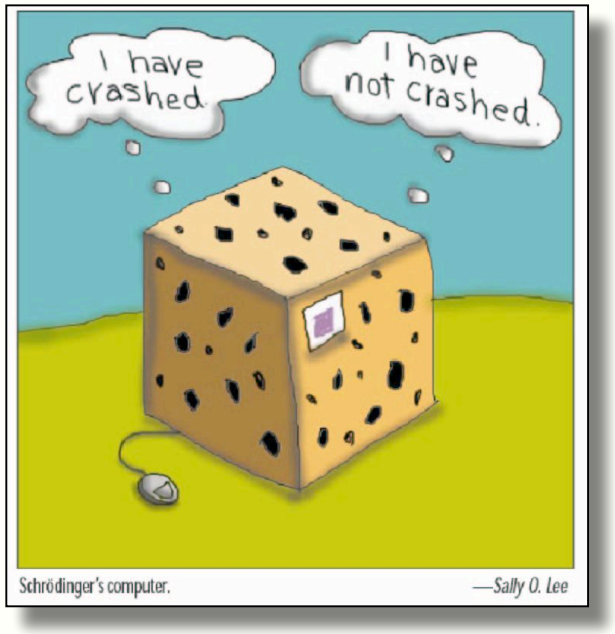
- Order finding algorithm +
  Euclid's algorithm ⇒ Factoring

Tune in next time…

- Nov. 16: No lecture, but….

  Sankar das Sarma: Physics, Chemistry, & Nanosciences Colloquium

  Topic: Spintronic quantum computing (9:15 a.m., building 897, Rms. 1010-1012).



- Nov. 23: Next lecture on Quantum Error Correction by Ivan Deutsch.