QIC 710 / CS 768 / PH 767 / CO 681 / AM 871 Fall 2020

Introduction to
Quantum Information Processing

Richard Cleve

Institute for Quantum Computing

and

Cheriton School of Computer Science

Lecture 11

Algorithms for order-finding and factoring

Richard Cleve 2020

Divisors vs common divisors

Factoring a number x means finding its divisors

Definition: the **common divisors** of two numbers, x and y, are the numbers that divide both of them

Example: the common divisors of 28 and 42 are: $1, 2, 7,$ and 14

Definition: let $\mathbf{gcd}(x, y)$ be the largest common divisor of x and y ($\gcd$ = "greatest common divisor")

Examples: $\gcd(28, 42) = 14 \quad \gcd(16, 21) = 1$

Definition:
x and y are **relatively prime** if $\gcd(x, y) = 1$

Euclid's algorithm (~300 BC)
an efficient algorithm for $\gcd(x, y)$
with gate cost $O(n^2 \log n)$

Euclid
Source: Wikipedia

Fragment from "Elements"
Source: Wikipedia

Richard Cleve 2020

Order-finding problem

For any integer $m \geq 2$ (not necessarily prime), we define two sets:

- $\mathbb{Z}_m = \{0, 1, 2, \ldots, m-1\}$
- $\mathbb{Z}_m^* = \{a \in \mathbb{Z}_m : \gcd(a, m) = 1\}$

Examples: $\mathbb{Z}_{21} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20\}$

$\mathbb{Z}_{21}^* = \{1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20\}$

The powers of 5 ($5^0, 5^1, 5^2, 5^3, \ldots$): $1, 5, 4, 20, 16, 17, 1, 5, 4, 20, 16, 17, 1, 5, 4, 20, 16, \ldots$

Definition: for $a \in \mathbb{Z}_m^*$, we define $\mathrm{ord}_m(a)$ (the order of a modulo m) as the minimum $r > 0$ such that $a^r \bmod m = 1$

For example, $\mathrm{ord}_{21}(5) = 6$ (the periodicity of the above sequence)

Order-finding problem

input: $m \geq 2$ (n-bit number) and $a \in \mathbb{Z}_m^*$

output: $\mathrm{ord}_m(a)$

Why should we care about this problem?

One reason: the factoring problem reduces to it …

Richard Cleve 2020

Factoring ← order-finding

Any efficient algorithm for **order-finding** can be used to **factor** efficiently

The general idea:

Consider the case where $m = pq$ (distinct primes)

Select an $a \in \mathbb{Z}_m^*$ and then compute $r = \mathrm{ord}_m(a)$

Note: $a^r = 1 \pmod m$ means m divides $a^r - 1$

For a **lucky** a, it holds that r is an even number

In that case, $a^r - 1 = (a^{r/2} + 1)(a^{r/2} - 1)$

Then we have: pq divides $(a^{r/2} + 1)(a^{r/2} - 1)$

This can happen three ways:

lucky

1. p divides $a^{r/2} + 1$ & q divides $a^{r/2} - 1$
2. p divides $a^{r/2} - 1$ & q divides $a^{r/2} + 1$
3. p and q both divide $a^{r/2} + 1$
4. p and q both divide $a^{r/2} - 1$ **impossible**

For any **lucky** $a \in \mathbb{Z}_m^*$, it holds that $\gcd\left(a^{r/2} + 1, m\right) = p$ or q

computable with $O(n^2 \log n)$ gates

How do we find a **lucky** $a \in \mathbb{Z}_m^*$?

Answer: randomly choose an $a \in \mathbb{Z}_m^*$ (this succeeds with probability $1/2$)

Richard Cleve 2020

Order-finding algorithm: broad idea

Input: m (an n-bit number) and $a \in \mathbb{Z}_m^*$

Goal: find $r = \mathrm{ord}_m(a)$, which is the minimum $r > 0$ such that $a^r \bmod m = 1$

Let $\begin{cases} U \text{ be an } n\text{-qubit unitary such that } U|b\rangle = |ab \bmod m\rangle \text{ for all } b \in \mathbb{Z}_m^* \\ |\psi_1\rangle = |1\rangle + \omega^{-1}|a\rangle + \omega^{-2}|a^2\rangle + \cdots + \omega^{-(r-1)}|a^{r-1}\rangle = \sum_{j=0}^{r-1} \omega^{-j}|a^j \bmod m\rangle \quad \text{where } \omega = e^{2\pi i/r} \end{cases}$

Claim: $|\psi_1\rangle$ is an eigenvector of U with eigenvalue $e^{2\pi i\left(\frac{1}{r}\right)}$

Proof:

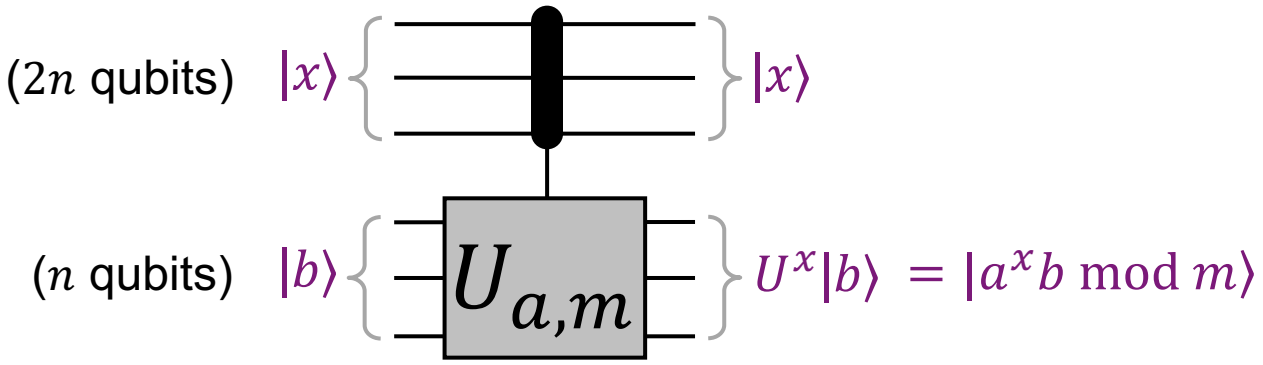
$$\begin{aligned} U|\psi_1\rangle &= |a\rangle + \omega^{-1}|a^2\rangle + \omega^{-2}|a^3\rangle + \cdots + \omega^{-(r-1)}|1\rangle \\ &= \omega\left(\omega^{-1}|a\rangle + \omega^{-2}|a^2\rangle + \omega^{-3}|a^3\rangle + \cdots + \omega^{-r}|1\rangle\right) \\ &= \omega|\psi_1\rangle \end{aligned}$$

We'll try to use the **phase estimation** algorithm to approximate $\frac{1}{r}$ and then calculate r

Richard Cleve 2020

The multiplicity-controlled-U gate

Define $U_{a,m}|b\rangle = |ab \bmod m\rangle$

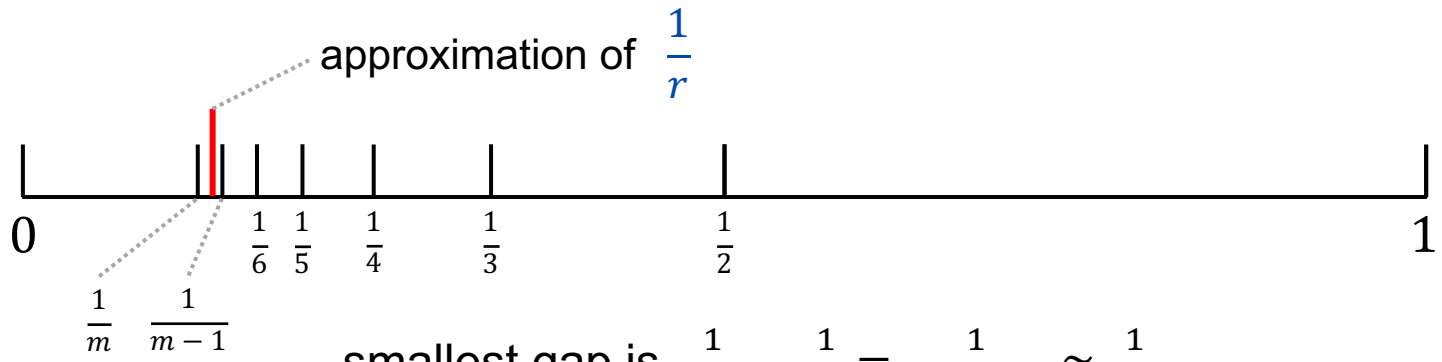


This can be implemented with $O(n^2 \log n)$ gates (using the repeated squaring trick)

Why $2n$ bits for the control?

The eigenvalue of $|\psi_1\rangle$ is $e^{2\pi i\left(\frac{1}{r}\right)}$

A $2n$-bit precision approximation of $\frac{1}{r}$ is necessary and sufficient to deduce r



smallest gap is $\frac{1}{m-1} - \frac{1}{m} = \frac{1}{m(m-1)} \approx \frac{1}{m^2}$

For $b \notin \mathbb{Z}_m^*$, set $U_{a,m}|b\rangle = |b\rangle$

Richard Cleve 2020

Constructing the eigenvector

For phase-estimation, we also need the state

$$|\psi_1\rangle = \sum_{j=0}^{r-1} \omega^{-j} |a^j \bmod m\rangle \quad \text{where } \omega = e^{2\pi i/r}$$

How can we construct this without already knowing what r is?

It's not known how to do this! This is a serious problem!

Instead of constructing $|\psi_1\rangle$, we'll construct **something else** that can be used

Richard Cleve 2020

Random eigenvector

Define

	eigenvalue
$\|\psi_1\rangle = \sum_{j=0}^{r-1} \omega^{-j} \|a^j\rangle$	$e^{2\pi i\left(\frac{1}{r}\right)}$
$\|\psi_2\rangle = \sum_{j=0}^{r-1} \omega^{-2j} \|a^j\rangle$	$e^{2\pi i\left(\frac{2}{r}\right)}$
⋮	⋮
$\|\psi_k\rangle = \sum_{j=0}^{r-1} \omega^{-kj} \|a^j\rangle$	$e^{2\pi i\left(\frac{k}{r}\right)}$
⋮	⋮
$\|\psi_r\rangle = \sum_{j=0}^{r-1} \omega^{-rj} \|a^j\rangle$	$e^{2\pi i\left(\frac{r}{r}\right)}$

Suppose we have a method that generates $(k, |\psi_k\rangle)$ for a uniformly distributed ***random*** $k \in \{1, 2, \ldots, r\}$

Can we extract r from this?

Yes, use phase estimation to approximate $\frac{k}{r}$ and then divide by k

Suppose we have a method to generate $|\psi_k\rangle$ for a random $k \in \{1, 2, \ldots, r\}$ (***not*** revealing k)

Can we extract k and r from this?

Can use phase estimation to approximate $\frac{k}{r}$ — but then figuring out k and r is trickier …

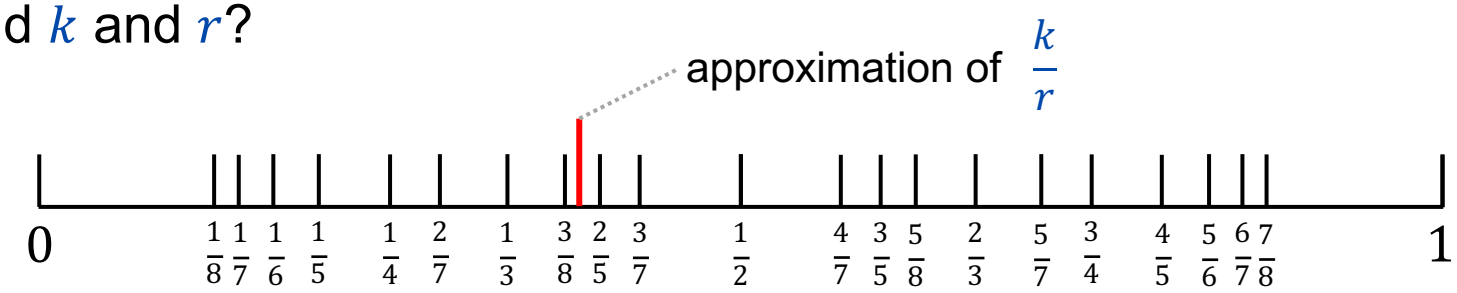
Richard Cleve 2020

Extracting k and r

Scenario: $r \in \{1, 2, \ldots, m\}$ and $k \in \{1, 2, \ldots, r\}$ and we have an approximation of $\frac{k}{r}$

How do we find k and r?

If $m = 8$ then



The smallest gap is $\frac{1}{m-1} - \frac{1}{m} = \frac{1}{m(m-1)} \approx \frac{1}{m^2}$ so $2n$ bits precision is sufficient

How can k and r be found efficiently?

There's an efficient algorithm for finding the number $\frac{k}{r}$ (as a reduced fraction) with $O(n^2 \log n)$ gates

Discovered in the 1600s!

"**continued fractions algorithm**"

Christiaan Huygens

Source: Popular Science Monthly Volume 28

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\ddots + \cfrac{1}{a_n}}}}$$

Richard Cleve 2020

Problem of reduced fractions

The continued fraction algorithm does not find k and r in the cases where they have a common factor

An example of such a bad case is $k = 34$ and $r = 51$ (since $\frac{34}{51} = \frac{2}{3}$)

In our framework, k is chosen randomly (uniformly) from $\{1, 2, \ldots, r\}$

What's the probability that $\gcd(k, r) = 1$?

Answer: it's $|\mathbb{Z}_r^*|/|\mathbb{Z}_r|$, which is known to scale as $\geq \frac{1}{\log\log r} \geq \frac{1}{\log\log m} \geq \frac{1}{\log n}$

Therefore $O(\log n)$ repetitions of the process ensures that the probability of at least one **good pair** (k, r) with $\gcd(k, r) = 1$ is constant

In fact, **two** repetitions $\frac{k_1}{r}$ and $\frac{k_2}{r}$ are enough: for reduced fractions $\frac{k_1'}{r_1'}$ and $\frac{k_2'}{r_2'}$ the probability that $\mathrm{lcm}(r_1', r_2') = r$ is constant

---

$\mathrm{lcm}(x, y)$ is the least common multiple of x and y

Richard Cleve 2020

Status report (so far)

We're solving the order-finding problem using the **phase estimation algorithm**

The multiplicity-controlled-$U_{a,m}$ is straightforward to compute 🙂

Algorithm works using $|\psi_1\rangle$ … but we don't know how to generate that state 😕

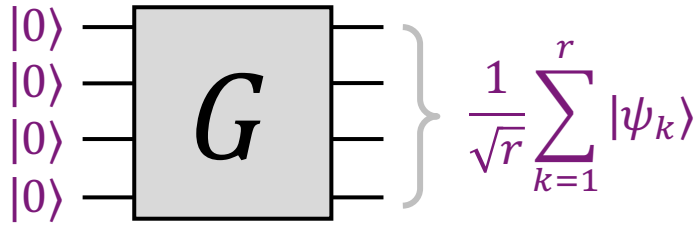
Algorithm works using a **random** $(k, |\psi_k\rangle)$ … but we don't know how to generate that 😕

Algorithm works using a **random** $|\psi_k\rangle$ … but we don't know how to generate that either 🤬

But hang on! We're almost at a solution!

Algorithm works using a **superposition of** $|\psi_k\rangle$ … and we **can** generate that state 😀

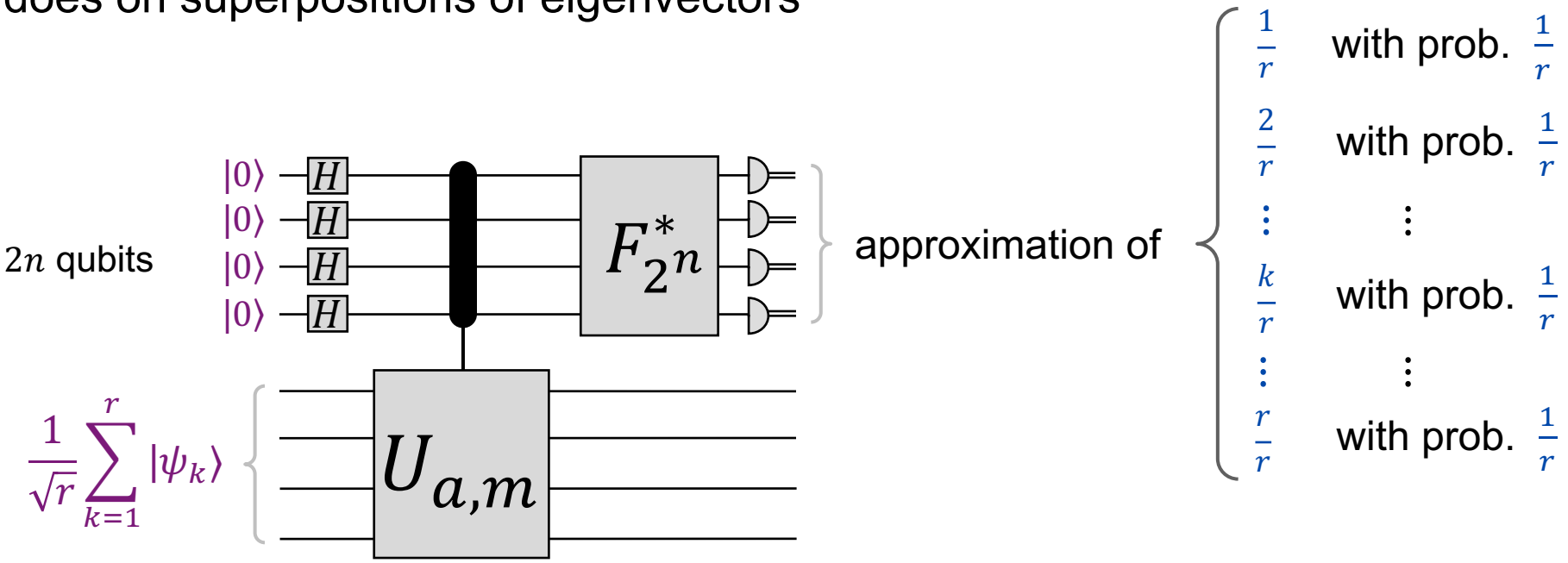
$$\frac{1}{\sqrt{r}}|\psi_1\rangle + \frac{1}{\sqrt{r}}|\psi_2\rangle + \frac{1}{\sqrt{r}}|\psi_3\rangle + \cdots + \frac{1}{\sqrt{r}}|\psi_r\rangle$$



Richard Cleve 2020

Why superposition is OK

Recall what the **phase estimation algorithm** does on superpositions of eigenvectors



Richard Cleve 2020

Generating the superposition

$$\frac{1}{\sqrt{r}}\sum_{k=1}^{r}|\psi_k\rangle = |00\cdots01\rangle$$ Why?

$$\frac{1}{\sqrt{r}}\sum_{k=1}^{r}|\psi_k\rangle = \frac{1}{\sqrt{r}}\left(\frac{1}{\sqrt{r}}|1\rangle + \frac{\omega^{-1}}{\sqrt{r}}|a\rangle + \frac{\omega^{-2}}{\sqrt{r}}|a^2\rangle + \cdots + \frac{\omega^{-(r-1)}}{\sqrt{r}}|a^{r-1}\rangle\right) \leftarrow k=1$$

$$+\frac{1}{\sqrt{r}}\left(\frac{1}{\sqrt{r}}|1\rangle + \frac{\omega^{-2}}{\sqrt{r}}|a\rangle + \frac{\omega^{-4}}{\sqrt{r}}|a^2\rangle + \cdots + \frac{\omega^{-2(r-1)}}{\sqrt{r}}|a^{r-1}\rangle\right) \leftarrow k=2$$

$$\vdots \qquad \vdots \qquad \vdots \qquad \vdots \qquad \vdots$$

$$+\frac{1}{\sqrt{r}}\left(\frac{1}{\sqrt{r}}|1\rangle + \frac{\omega^{-r}}{\sqrt{r}}|a\rangle + \frac{\omega^{-2r}}{\sqrt{r}}|a^2\rangle + \cdots + \frac{\omega^{-r(r-1)}}{\sqrt{r}}|a^{r-1}\rangle\right) \leftarrow k=r$$

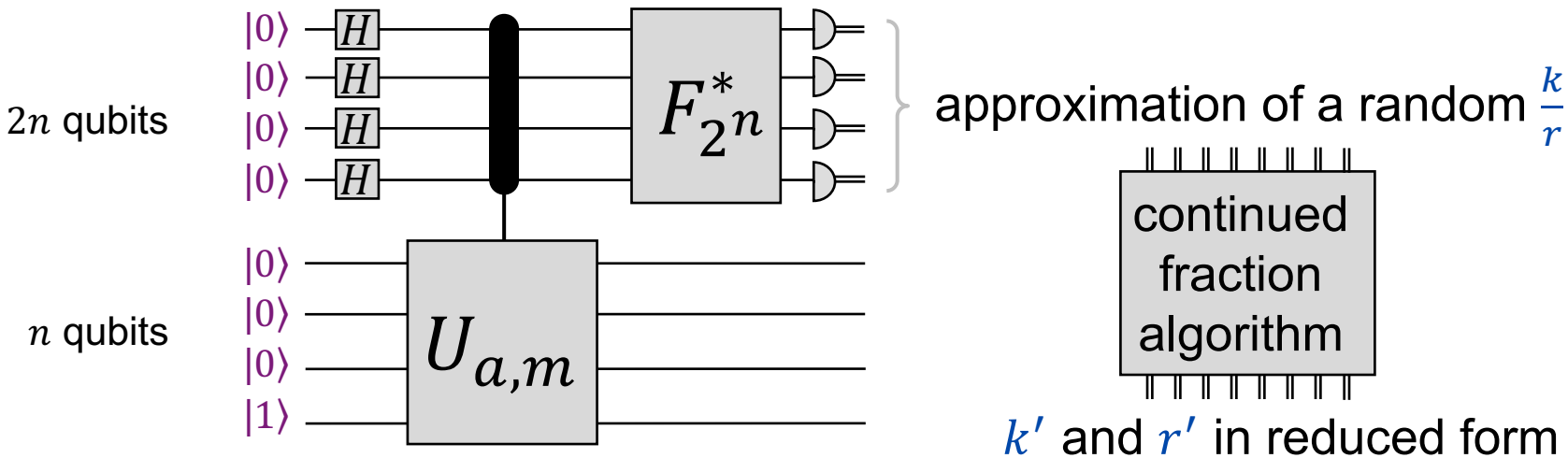
$$= |1\rangle \qquad = 0 \qquad = 0 \qquad = 0$$

$$= |00\cdots01\rangle$$

Richard Cleve 2020

Order-finding algorithm

input: m (an n-bit number) and $a \in \mathbb{Z}_m^*$ **output:** $r = \mathrm{ord}_m(a)$



With constant probability:

the phase estimation algorithm succeeds

and a non-reduced k and r are obtained (with two runs)

The total gate cost is $O(n^2 \log n)$

Richard Cleve 2020

Factoring algorithm

input: m (an n-bit number) **output:** divisor of m

There's a simple classical algorithm for the special case where m is a prime power ($m = p^d$)

Exercise: give an algorithm for the prime power case

In all other cases, the following algorithm works:

Algorithm

choose a random non-zero $a \in \mathbb{Z}_m$

compute $g \leftarrow \gcd(a, m)$

if $g = 1$ **then** (in this case $a \in \mathbb{Z}_m^*$)

compute $r \leftarrow \mathrm{ord}_m(a)$ ..... **quantum part**

compute $h \leftarrow \gcd(a^{r/2} + 1, m)$

output h

if $g > 1$ **then** (in this case g is a divisor)

output g

With constant probability, this algorithm outputs a divisor of m

The total gate cost is $O(n^2 \log n)$

Richard Cleve 2020