Parallel Spooky Pebbling Makes Regev Factoring More Practical

[arXiv:2510.08432]

Greg Kahanamoku-Meyer [1] Seyoon Ragavan [1] Katherine Van Kirk [2]

[1]MIT

[2]Harvard

January 29, 2026

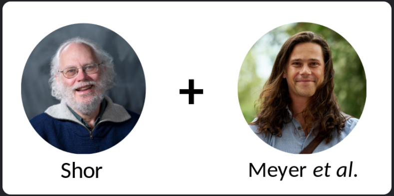
Circuits for factoring general-form numbers

Core of Shor's algorithm

To factor an n-bit number N, biggest cost is modular exponentiation in superposition:

$$f(a, x, N) = a^x \bmod N$$

Circuits for factoring general-form numbers

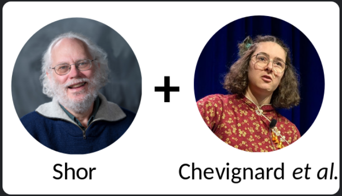


Core of Shor's algorithm

To factor an n-bit number N, biggest cost is modular exponentiation in superposition:

$$f(a, x, N) = a^x \bmod N$$

Circuits for factoring general-form numbers



Core of Shor's algorithm

To factor an n-bit number N, biggest cost is modular exponentiation in superposition:

$$f(a, x, N) = a^x \bmod N$$

Circuits for factoring general-form numbers

Core of Shor's algorithm

To factor an n-bit number N, biggest cost is modular exponentiation in superposition:

$$f(a, x, N) = a^x \bmod N$$

Circuits for factoring general-form n-bit numbers

Shor

Regev

Core of Shor's algorithm

$$f(a, x, N) = a^x \bmod N$$

Core of Regev's algorithm

$$f(\vec{a}, \vec{x}, N) = \prod_i a_i^{x_i} \bmod N$$

Circuits for factoring general-form n-bit numbers

Shor

Regev

Core of Shor's algorithm

$$f(a, x, N) = a^x \text{ mod } N$$

Gates: $\widetilde{O}(n^2)$ (per shot)

Core of Regev's algorithm

$$f(\vec{a}, \vec{x}, N) = \prod_i a_i^{x_i} \text{ mod } N$$

Gates: $\widetilde{O}(n^{3/2})$ (per shot)

Circuits for factoring general-form n-bit numbers

Core of Shor's algorithm

$$f(a, x, N) = a^x \bmod N$$

Gates: $\widetilde{O}(n^2)$ (per shot)
Qubits: $\widetilde{O}(n)$

Core of Regev's algorithm

$$f(\vec{a}, \vec{x}, N) = \prod_i a_i^{x_i} \bmod N$$

Gates: $\widetilde{O}(n^{3/2})$ (per shot)
Qubits: $O(n^{3/2})$

Algorithms for exponentiation

Intuition: how to compute $a^x \bmod N$ when x is a power of two?

Algorithms for exponentiation

Intuition: how to compute $a^x \bmod N$ when x is a power of two?

Repeated squaring: (everything is mod N)

$$a$$

Algorithms for exponentiation

Intuition: how to compute $a^x \bmod N$ when x is a power of two?

Repeated squaring: (everything is mod N)

$$a \to a^2$$

Algorithms for exponentiation

Intuition: how to compute $a^x \bmod N$ when x is a power of two?

Repeated squaring: (everything is mod N)

$$a \to a^2 \to a^4$$

Algorithms for exponentiation

Intuition: how to compute $a^x \bmod N$ when x is a power of two?

Repeated squaring: (everything is mod N)

$$a \to a^2 \to a^4 \to a^8$$

Algorithms for exponentiation

Intuition: how to compute $a^x \bmod N$ when x is a power of two?

Repeated squaring: (everything is mod N)

$$a \to a^2 \to a^4 \to a^8 \to a^{16}$$

Algorithms for exponentiation

Intuition: how to compute $a^x \bmod N$ when x is a power of two?

Repeated squaring: (everything is mod N)

$$a \to a^2 \to a^4 \to a^8 \to a^{16} \to \cdots$$

Algorithms for exponentiation

Intuition: how to compute $a^x \bmod N$ when x is a power of two?

Repeated squaring: (everything is mod N)

$$a \to a^2 \to a^4 \to a^8 \to a^{16} \to \cdots \to a^x$$

Algorithms for exponentiation

Intuition: how to compute $a^x \bmod N$ when x is a power of two?

Repeated squaring: (everything is mod N)

$$a \to a^2 \to a^4 \to a^8 \to a^{16} \to \cdots \to a^x$$

General case: multiply in extra factors of a along the way...

Algorithms for exponentiation

How to compute $|x\rangle\,|0\rangle \rightarrow |x\rangle\,|a^x \bmod N\rangle$ quantumly?

Algorithms for exponentiation

How to compute $|x\rangle|0\rangle \to |x\rangle|a^x \bmod N\rangle$ quantumly?

Repeated squaring?

$$|x\rangle|a\rangle \to |x\rangle|a^2\rangle \to |x\rangle|a^4\rangle \to |x\rangle|a^8\rangle \to |x\rangle|a^{16}\rangle \to \cdots \to |x\rangle|a^x\rangle$$

Algorithms for exponentiation

How to compute $|x\rangle|0\rangle \to |x\rangle|a^x \bmod N\rangle$ quantumly?

Repeated squaring?

$$|x\rangle|a\rangle \to |x\rangle|a^2\rangle \to |x\rangle|a^4\rangle \to |x\rangle|a^8\rangle \to |x\rangle|a^{16}\rangle \to \cdots \to |x\rangle|a^x\rangle$$

Issue: squaring mod N is not reversible!

Algorithms for exponentiation

How to compute $|x\rangle|0\rangle \to |x\rangle|a^x \bmod N\rangle$ quantumly?

Repeated squaring?

$$|x\rangle|a\rangle \to |x\rangle|a^2\rangle \to |x\rangle|a^4\rangle \to |x\rangle|a^8\rangle \to |x\rangle|a^{16}\rangle \to \cdots \to |x\rangle|a^x\rangle$$

Issue: squaring mod N is not reversible!

General problem: How to do $|x\rangle|0\rangle \to |x\rangle|f(x)\rangle$ efficiently when f has many irreversible steps?

Aside: avoiding irreversibility

What if we restructure our computation so each step is reversible?

Aside: avoiding irreversibility

What if we restructure our computation so each step is reversible?

Replace **squaring** with **multiplication by constants**, which is reversible!

Aside: avoiding irreversibility

What if we restructure our computation so each step is reversible?

Shor

To compute $a^x \bmod N$

Replace **squaring** with **multiplication by constants**, which is reversible!

Aside: avoiding irreversibility

What if we restructure our computation so each step is reversible?

Replace **squaring** with **multiplication by constants**, which is reversible!

To compute $a^x \bmod N = \prod_i \left(a^{2^i}\right)^{x_i} \bmod N$:

Aside: avoiding irreversibility

What if we restructure our computation so each step is reversible?


Shor

Replace **squaring** with **multiplication by constants**, which is reversible!

To compute $a^x \bmod N = \prod_i \left(a^{2^i}\right)^{x_i} \bmod N$:

- **classically** precompute $a^2, a^4, a^8, \cdots \pmod{N}$
- Multiply together for all i where bit $x_i = 1$

Aside: avoiding irreversibility

What if we restructure our computation so each step is reversible?

Regev

Regev's factoring speedup comes from multiplied values being **small**

Aside: avoiding irreversibility

What if we restructure our computation so each step is reversible?

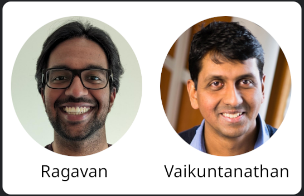
Regev's factoring speedup comes from multiplied values being **small**

😢 $a_j^{2^i}$ mod N in general *not* small

- Shor's rearrangement would **kill improvement** in gate count

Aside: avoiding irreversibility

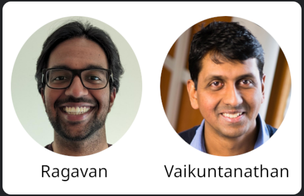
What if we restructure our computation so each step is reversible?



Let's do "repeated squaring,"
but with **Fibonacci number**
exponents instead of powers of 2

Aside: avoiding irreversibility

What if we restructure our computation so each step is reversible?



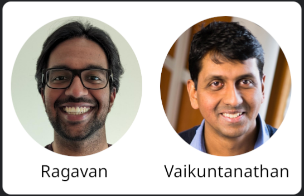
Let's do "repeated squaring," but with **Fibonacci number** exponents instead of powers of 2

😃 "Squaring" is **reversible** in this representation!!

- Maintains Regev's gate count of $\tilde{O}(n^{3/2})$
- Reduces qubit count to $O(n)$

Aside: avoiding irreversibility

What if we restructure our computation so each step is reversible?



Let's do "repeated squaring," but with **Fibonacci number** exponents instead of powers of 2

😃 "Squaring" is **reversible** in this representation!!

- Maintains Regev's gate count of $\tilde{O}(n^{3/2})$
- Reduces qubit count to $O(n)$

😢 Large constant factors → Shor still wins in practice

Aside: avoiding irreversibility

What if we restructure our computation so each step is reversible?

Regev's factoring speedup comes from multiplied values being **small**

Let's see what we can do if we **accept irreversibility** of each step

The general problem

General problem: How to do $|x\rangle|0\rangle \rightarrow |x\rangle|f(x)\rangle$ efficiently when f has many irreversible steps?

Consider an algorithm f with k steps f_i.

The general problem

General problem: How to do $|x\rangle|0\rangle \to |x\rangle|f(x)\rangle$ efficiently when f has many irreversible steps?

Consider an algorithm f with k steps f_i.

$$x$$

The general problem

General problem: How to do $|x\rangle |0\rangle \rightarrow |x\rangle |f(x)\rangle$ efficiently when f has many irreversible steps?

Consider an algorithm f with k steps f_i.

$$f_1(x)$$

The general problem

General problem: How to do $|x\rangle |0\rangle \to |x\rangle |f(x)\rangle$ efficiently when f has many irreversible steps?

Consider an algorithm f with k steps f_i.

$$f_2(f_1(x))$$

The general problem

General problem: How to do $|x\rangle|0\rangle \to |x\rangle|f(x)\rangle$ efficiently when f has many irreversible steps?

Consider an algorithm f with k steps f_i.

$$f_{k-1}(\cdots f_2(f_1(x))\cdots)$$

The general problem

General problem: How to do $|x\rangle|0\rangle \rightarrow |x\rangle|f(x)\rangle$ efficiently when f has many irreversible steps?

Consider an algorithm f with k steps f_i.

$$f(x) := f_k(f_{k-1}(\cdots f_2(f_1(x))\cdots))$$

The general problem

General problem: How to do $|x\rangle|0\rangle \to |x\rangle|f(x)\rangle$ efficiently when f has many irreversible steps?

Consider an algorithm f with k steps f_i.

$$f(x) := f_k(f_{k-1}(\cdots f_2(f_1(x))\cdots))$$

Let $z_i = f_i(\cdots)$. One quantum way to do it:

The general problem

General problem: How to do $|x\rangle|0\rangle \to |x\rangle|f(x)\rangle$ efficiently when f has many irreversible steps?

Consider an algorithm f with k steps f_i.

$$x$$

Let $z_i = f_i(\cdots)$. One quantum way to do it:

$$|x\rangle$$

The general problem

General problem: How to do $|x\rangle|0\rangle \to |x\rangle|f(x)\rangle$ efficiently when f has many irreversible steps?

Consider an algorithm f with k steps f_i.

$$f_1(x)$$

Let $z_i = f_i(\cdots)$. One quantum way to do it:

$$|x\rangle\,|z_1\rangle$$

The general problem

General problem: How to do $|x\rangle|0\rangle \to |x\rangle|f(x)\rangle$ efficiently when f has many irreversible steps?

Consider an algorithm f with k steps f_i.

$$f_2(f_1(x))$$

Let $z_i = f_i(\cdots)$. One quantum way to do it:

$$|x\rangle\ |z_1\rangle\ |z_2\rangle$$

The general problem

General problem: How to do $|x\rangle|0\rangle \to |x\rangle|f(x)\rangle$ efficiently when f has many irreversible steps?

Consider an algorithm f with k steps f_i.

$$f_{k-1}(\cdots f_2(f_1(x))\cdots)$$

Let $z_i = f_i(\cdots)$. One quantum way to do it:

$$|x\rangle\ |z_1\rangle\ |z_2\rangle\ \cdots\ |z_{k-1}\rangle$$

The general problem

General problem: How to do $|x\rangle|0\rangle \rightarrow |x\rangle|f(x)\rangle$ efficiently when f has many irreversible steps?

Consider an algorithm f with k steps f_i.

$$f(x) := f_k(f_{k-1}(\cdots f_2(f_1(x))\cdots))$$

Let $z_i = f_i(\cdots)$. One quantum way to do it:

$$|x\rangle\,|z_1\rangle\,|z_2\rangle\,\cdots\,|z_{k-1}\rangle\,|f(x)\rangle$$

The general problem

General problem: How to do $|x\rangle|0\rangle \to |x\rangle|f(x)\rangle$ efficiently when f has many irreversible steps?

Consider an algorithm f with k steps f_i.

$$f(x) := f_k(f_{k-1}(\cdots f_2(f_1(x))\cdots))$$

Let $z_i = f_i(\cdots)$. One quantum way to do it:

$$|x\rangle\,|z_1\rangle\,|z_2\rangle\,\cdots \qquad |f(x)\rangle$$

The general problem

General problem: How to do $|x\rangle|0\rangle \to |x\rangle|f(x)\rangle$ efficiently when f has many irreversible steps?

Consider an algorithm f with k steps f_i.

$$f(x) := f_k(f_{k-1}(\cdots f_2(f_1(x))\cdots))$$

Let $z_i = f_i(\cdots)$. One quantum way to do it:

$$|x\rangle\,|z_1\rangle\,|z_2\rangle \qquad\qquad |f(x)\rangle$$

The general problem

General problem: How to do $|x\rangle\,|0\rangle \rightarrow |x\rangle\,|f(x)\rangle$ efficiently when f has many irreversible steps?

Consider an algorithm f with k steps f_i.

$$f(x) := f_k(f_{k-1}(\cdots f_2(f_1(x))\cdots))$$

Let $z_i = f_i(\cdots)$. One quantum way to do it:

$$|x\rangle\,|z_1\rangle \qquad\qquad |f(x)\rangle$$

The general problem

General problem: How to do $|x\rangle|0\rangle \to |x\rangle|f(x)\rangle$ efficiently when f has many irreversible steps?

Consider an algorithm f with k steps f_i.

$$f(x) := f_k(f_{k-1}(\cdots f_2(f_1(x))\cdots))$$

😃 Total quantum steps: $2k-1$ (optimal)

Let $z_i = f_i(\cdots)$. One quantum way to do it:

$|x\rangle$ $|f(x)\rangle$

The general problem

General problem: How to do $|x\rangle|0\rangle \to |x\rangle|f(x)\rangle$ efficiently when f has many irreversible steps?

Consider an algorithm f with k steps f_i.

$$f(x) := f_k(f_{k-1}(\cdots f_2(f_1(x))\cdots))$$

Let $z_i = f_i(\cdots)$. One quantum way to do it:

$|x\rangle$ $|f(x)\rangle$

😃 Total quantum steps: $2k-1$ (optimal)

😢 Requires $O(k)$ ancilla registers...

The general problem

General problem: How to do $|x\rangle|0\rangle \to |x\rangle|f(x)\rangle$ efficiently when f has many irreversible steps?

Consider an algorithm f with k steps f_i.

$$f(x) := f_k(f_{k-1}(\cdots f_2(f_1(x))\cdots))$$

Let $z_i = f_i(\cdots)$. One quantum way to do it:

$|x\rangle$ $|f(x)\rangle$

😃 Total quantum steps: $2k - 1$ (optimal)

😢 Requires $O(k)$ ancilla registers...

😵‍💫 **Regev:** $O(n^{3/2})$ qubits

The general problem

General problem: How to do $|x\rangle|0\rangle \rightarrow |x\rangle|f(x)\rangle$ efficiently when f has many irreversible steps?

Consider an algorithm f with k steps f_i.

$$f(x) := f_k(f_{k-1}(\cdots f_2(f_1(x))\cdots))$$

Let $z_i = f_i(\cdots)$. One quantum way to do it:

$|x\rangle$ $|f(x)\rangle$

😃 Total quantum steps: $2k - 1$ (optimal)

😢 Requires $O(k)$ ancilla registers...

😵‍💫 **Regev:** $O(n^{3/2})$ qubits

Is it possible to do better?

Pebble games for reversible computation

Bennett ’89 introduced pebble games:

- Placing a pebble → computing a value (uses new space)

$$|x\rangle \qquad |z_1\rangle \qquad |z_2\rangle \qquad |z_3\rangle \qquad |f(x)\rangle$$

Pebble games for reversible computation

Bennett '89 introduced pebble games:

- Placing a pebble → computing a value (uses new space)
- Removing a pebble → uncomputing that value (frees space)

$|x\rangle \quad |z_1\rangle \quad |z_2\rangle \quad |z_3\rangle \quad |f(x)\rangle$

Pebble games for reversible computation

Bennett '89 introduced pebble games:

- Placing a pebble → computing a value (uses new space)
- Removing a pebble → uncomputing that value (frees space)
- **Rule:** Can only pebble/unpebble if value to the left has a pebble

$$|x\rangle \qquad |z_1\rangle \qquad |z_2\rangle \qquad |z_3\rangle \qquad |f(x)\rangle$$

Pebble games for reversible computation

Bennett '89 introduced pebble games:

- Placing a pebble → computing a value (uses new space)
- Removing a pebble → uncomputing that value (frees space)
- **Rule:** Can only pebble/unpebble if value to the left has a pebble
- **Goal:** place pebble on $f(x)$; leave no pebbles on z_i

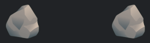
$$|x\rangle \quad |z_1\rangle \quad |z_2\rangle \quad |z_3\rangle \quad |f(x)\rangle$$

Pebble games for reversible computation

Bennett ’89 introduced pebble games:

- Placing a pebble → computing a value (uses new space)
- Removing a pebble → uncomputing that value (frees space)
- **Rule:** Can only pebble/unpebble if value to the left has a pebble
- **Goal:** place pebble on $f(x)$; leave no pebbles on z_i

$|x\rangle \quad |z_1\rangle \quad |z_2\rangle \quad |z_3\rangle \quad |f(x)\rangle$



Pebble games for reversible computation

Bennett '89 introduced pebble games:

- Placing a pebble → computing a value (uses new space)
- Removing a pebble → uncomputing that value (frees space)
- **Rule:** Can only pebble/unpebble if value to the left has a pebble
- **Goal:** place pebble on $f(x)$; leave no pebbles on z_i

$$|x\rangle \quad |z_1\rangle \quad |z_2\rangle \quad |z_3\rangle \quad |f(x)\rangle$$

Pebble games for reversible computation

Bennett ’89 introduced pebble games:

- Placing a pebble → computing a value (uses new space)
- Removing a pebble → uncomputing that value (frees space)
- **Rule:** Can only pebble/unpebble if value to the left has a pebble
- **Goal:** place pebble on $f(x)$; leave no pebbles on z_i

Pebble games for reversible computation

Bennett ’89 introduced pebble games:

- Placing a pebble → computing a value (uses new space)
- Removing a pebble → uncomputing that value (frees space)
- **Rule:** Can only pebble/unpebble if value to the left has a pebble
- **Goal:** place pebble on $f(x)$; leave no pebbles on z_i

Pebble games for reversible computation

Bennett ’89 introduced pebble games:

- Placing a pebble → computing a value (uses new space)
- Removing a pebble → uncomputing that value (frees space)
- **Rule:** Can only pebble/unpebble if value to the left has a pebble
- **Goal:** place pebble on $f(x)$; leave no pebbles on z_i

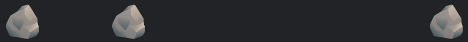
$$|x\rangle \quad |z_1\rangle \quad |z_2\rangle \quad |z_3\rangle \quad |f(x)\rangle$$

Pebble games for reversible computation

Bennett '89 introduced pebble games:

- Placing a pebble → computing a value (uses new space)
- Removing a pebble → uncomputing that value (frees space)
- **Rule:** Can only pebble/unpebble if value to the left has a pebble
- **Goal:** place pebble on $f(x)$; leave no pebbles on z_i

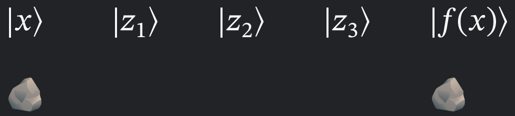
$$|x\rangle \quad |z_1\rangle \quad |z_2\rangle \quad |z_3\rangle \quad |f(x)\rangle$$



Pebble games for reversible computation

Bennett ’89 introduced pebble games:

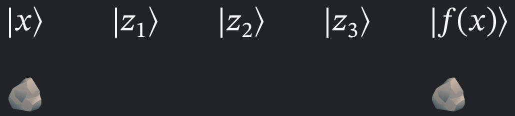
- Placing a pebble → computing a value (uses new space)
- Removing a pebble → uncomputing that value (frees space)
- **Rule:** Can only pebble/unpebble if value to the left has a pebble
- **Goal:** place pebble on $f(x)$; leave no pebbles on z_i



Pebble games for reversible computation

Bennett '89 introduced pebble games:

- Placing a pebble → computing a value (uses new space)
- Removing a pebble → uncomputing that value (frees space)
- **Rule:** Can only pebble/unpebble if value to the left has a pebble
- **Goal:** place pebble on $f(x)$; leave no pebbles on z_i



Space usage (max # pebbles): $O(k)$ registers
Time cost (# of steps): $2k - 1$ steps (optimal)

Using less space: a recursive strategy

1. Pebble from start to $k/2$

$$|x\rangle \quad |z_1\rangle \quad |z_2\rangle \quad |z_3\rangle \quad |f(x)\rangle$$

Using less space: a recursive strategy

1. Pebble from start to $k/2$

$$|x\rangle \quad |z_1\rangle \quad |z_2\rangle \quad |z_3\rangle \quad |f(x)\rangle$$

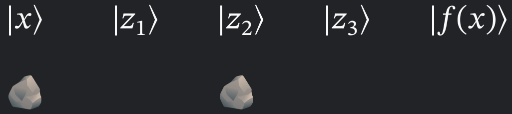
Using less space: a recursive strategy

1. Pebble from start to $k/2$

$$|x\rangle \quad |z_1\rangle \quad |z_2\rangle \quad |z_3\rangle \quad |f(x)\rangle$$

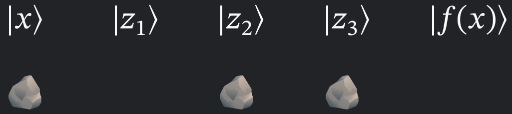
Using less space: a recursive strategy

1. Pebble from start to $k/2$
2. Pebble from $k/2$ to k



Using less space: a recursive strategy

1. Pebble from start to $k/2$
2. Pebble from $k/2$ to k

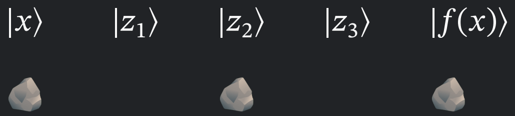


Using less space: a recursive strategy

1. Pebble from start to $k/2$
2. Pebble from $k/2$ to k

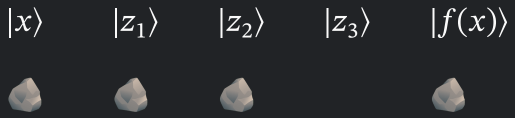
Using less space: a recursive strategy

1. Pebble from start to $k/2$
2. Pebble from $k/2$ to k
3. Remove pebble at $k/2$

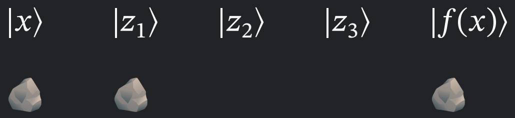


Using less space: a recursive strategy

1. Pebble from start to $k/2$
2. Pebble from $k/2$ to k
3. Remove pebble at $k/2$



1. Pebble from start to $k/2$
2. Pebble from $k/2$ to k
3. Remove pebble at $k/2$



Using less space: a recursive strategy

1. Pebble from start to $k/2$
2. Pebble from $k/2$ to k
3. Remove pebble at $k/2$

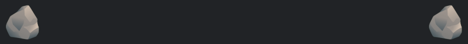
$$|x\rangle \quad |z_1\rangle \quad |z_2\rangle \quad |z_3\rangle \quad |f(x)\rangle$$

Space usage (max # pebbles): $O(\log k)$ registers 😃

Using less space: a recursive strategy

1. Pebble from start to $k/2$
2. Pebble from $k/2$ to k
3. Remove pebble at $k/2$

$$|x\rangle \quad |z_1\rangle \quad |z_2\rangle \quad |z_3\rangle \quad |f(x)\rangle$$



Space usage (max # pebbles): $O(\log k)$ registers 😃

Time cost (# of steps): $T(k) = 3T(k/2)$

Using less space: a recursive strategy

1. Pebble from start to $k/2$
2. Pebble from $k/2$ to k
3. Remove pebble at $k/2$

$$|x\rangle \quad |z_1\rangle \quad |z_2\rangle \quad |z_3\rangle \quad |f(x)\rangle$$

Space usage (max # pebbles): $O(\log k)$ registers 😃

Time cost (# of steps): $O(k^{\log_2 3}) \approx O(k^{1.58\cdots})$ steps 😢

Pebbling, but make it quantum

"Spooky Pebble Games and Irreversible Uncomputation"
algassert.com/post/1905

Pebbling, but make it quantum

"Spooky Pebble Games and Irreversible Uncomputation"
algassert.com/post/1905

Measurement-based uncomputation

Given an intermediate value

$$|z_i\rangle$$

Pebbling, but make it quantum

“Spooky Pebble Games and Irreversible Uncomputation”
algassert.com/post/1905

Measurement-based uncomputation

Given an intermediate value

$$|z_i\rangle$$

what if we apply $H^{\otimes n}$ and then measure it?

Pebbling, but make it quantum

“Spooky Pebble Games and Irreversible Uncomputation”
algassert.com/post/1905

Measurement-based uncomputation

Given an intermediate value

$$|z_i\rangle$$

what if we apply $H^{\otimes n}$ and then measure it? Get phase

$$(-1)^{d \cdot z_i}$$

where d is classically-known measurement outcome.

Pebbling, but make it quantum

"Spooky Pebble Games and Irreversible Uncomputation"
algassert.com/post/1905

Measurement-based uncomputation

Given an intermediate value

$$|z_i\rangle$$

what if we apply $H^{\otimes n}$ and then measure it? Get phase

$$(-1)^{d \cdot z_i}$$

where d is classically-known measurement outcome.

We've turned $|z_i\rangle$ into a ghost!

Spooky pebble games

Rules:

- can only place or remove a pebble if there is a pebble to its left

Spooky pebble games

Rules:

- can only place or remove a pebble if there is a pebble to its left
- can ghost a pebble at any time

Spooky pebble games

Rules:

- can only place or remove a pebble if there is a pebble to its left
- can ghost a pebble at any time—but must exorcise later by placing a pebble again

Spooky pebble games

Rules:

- can only place or remove a pebble if there is a pebble to its left
- can ghost a pebble at any time—but must exorcise later by placing a pebble again

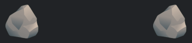
$|x\rangle$

Spooky pebble games

Rules:

- can only place or remove a pebble if there is a pebble to its left
- can ghost a pebble at any time—but must exorcise later by placing a pebble again

$|x\rangle$ $|z_1\rangle$



Spooky pebble games

Rules:

- can only place or remove a pebble if there is a pebble to its left
- can ghost a pebble at any time—but must exorcise later by placing a pebble again

Spooky pebble games

Rules:

- can only place or remove a pebble if there is a pebble to its left
- can ghost a pebble at any time—but must exorcise later by placing a pebble again

Spooky pebble games

Rules:

- can only place or remove a pebble if there is a pebble to its left
- can ghost a pebble at any time—but must exorcise later by placing a pebble again

$|x\rangle$ $\quad (-1)^{d_1 \cdot z_1}$ $\quad |z_2\rangle$

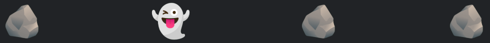
🪨 👻 🪨

Spooky pebble games

Rules:

- can only place or remove a pebble if there is a pebble to its left
- can ghost a pebble at any time—but must exorcise later by placing a pebble again

$$|x\rangle \qquad (-1)^{d_1 \cdot z_1} \qquad |z_2\rangle \qquad |z_3\rangle$$

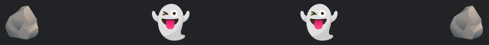


Spooky pebble games

Rules:

- can only place or remove a pebble if there is a pebble to its left
- can ghost a pebble at any time—but must exorcise later by placing a pebble again

$|x\rangle \qquad (-1)^{d_1 \cdot z_1} \qquad (-1)^{d_2 \cdot z_2} \qquad |z_3\rangle$



Spooky pebble games

Rules:

- can only place or remove a pebble if there is a pebble to its left
- can ghost a pebble at any time—but must exorcise later by placing a pebble again

$\lvert x\rangle$	$(-1)^{d_1 \cdot z_1}$	$(-1)^{d_2 \cdot z_2}$	$\lvert z_3\rangle$	$\lvert f(x)\rangle$
🪨	👻	👻	🪨	🪨

Spooky pebble games

Rules:

- can only place or remove a pebble if there is a pebble to its left
- can ghost a pebble at any time—but must exorcise later by placing a pebble again

$\lvert x\rangle$	$(-1)^{d_1 \cdot z_1}$	$(-1)^{d_2 \cdot z_2}$	$(-1)^{d_3 \cdot z_3}$	$\lvert f(x)\rangle$
🪨	👻	👻	👻	🪨

Spooky pebble games

Rules:

- can only place or remove a pebble if there is a pebble to its left
- can ghost a pebble at any time—but must exorcise later by placing a pebble again

$$|x\rangle \quad (-1)^{d_1 \cdot z_1} \quad (-1)^{d_2 \cdot z_2} \quad (-1)^{d_3 \cdot z_3} \quad |f(x)\rangle$$

🪨 👻 👻 👻 🪨

We've been tempted too strongly by the power of dark magic...

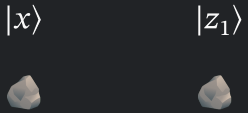
Spooky pebble games

1. Blast straight to $k/2$, leaving ghosts

$|x\rangle$

Spooky pebble games

1. Blast straight to $k/2$, leaving ghosts



Spooky pebble games

1. Blast straight to $k/2$, leaving ghosts

Spooky pebble games

1. Blast straight to $k/2$, leaving ghosts
2. Pebble from $k/2$ to k

$|x\rangle$ $(-1)^{d_1 \cdot z_1}$ $|z_2\rangle$

🪨 👻 🪨

Spooky pebble games

1. Blast straight to $k/2$, leaving ghosts
2. Pebble from $k/2$ to k

$|x\rangle$ $\quad$ $(-1)^{d_1 \cdot z_1}$ $\quad$ $|z_2\rangle$ $\quad$ $|z_3\rangle$

🪨 👻 🪨 🪨

Spooky pebble games

1. Blast straight to $k/2$, leaving ghosts
2. Pebble from $k/2$ to k

$\|x\rangle$	$(-1)^{d_1 \cdot z_1}$	$\|z_2\rangle$	$\|z_3\rangle$	$\|f(x)\rangle$
🪨	👻	🪨	🪨	🪨

Spooky pebble games

1. Blast straight to $k/2$, leaving ghosts
2. Pebble from $k/2$ to k
3. Remove pebble at $k/2$

$|x\rangle$ $(-1)^{d_1 \cdot z_1}$ $|z_2\rangle$ $|f(x)\rangle$

Spooky pebble games

1. Blast straight to $k/2$, leaving ghosts
2. Pebble from $k/2$ to k
3. Remove pebble at $k/2$

$|x\rangle$ $\quad (-1)^{d_1 \cdot z_1} |z_1\rangle \quad |z_2\rangle \quad |f(x)\rangle$

🪨 👻 🪨 🪨 🪨

Spooky pebble games

1. Blast straight to $k/2$, leaving ghosts
2. Pebble from $k/2$ to k
3. Remove pebble at $k/2$

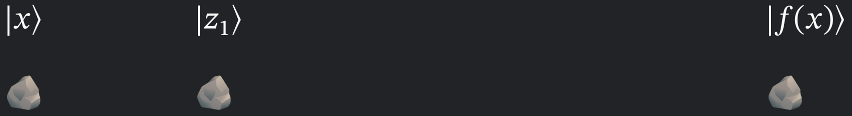
$$|x\rangle \quad (-1)^{d_1 \cdot z_1} |z_1\rangle \quad |z_2\rangle \quad |f(x)\rangle$$

Spooky pebble games

1. Blast straight to $k/2$, leaving ghosts
2. Pebble from $k/2$ to k
3. Remove pebble at $k/2$

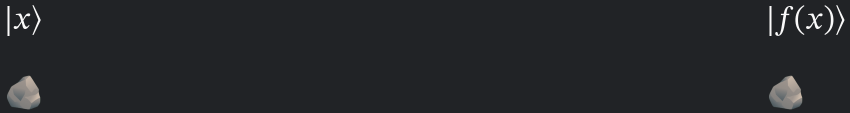
Spooky pebble games

1. Blast straight to $k/2$, leaving ghosts
2. Pebble from $k/2$ to k
3. Remove pebble at $k/2$



Spooky pebble games

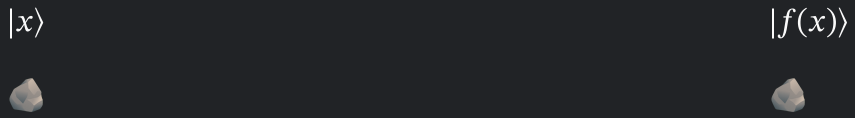
1. Blast straight to $k/2$, leaving ghosts
2. Pebble from $k/2$ to k
3. Remove pebble at $k/2$



Space: $O(\log k)$ pebbles 😃

Spooky pebble games

1. Blast straight to $k/2$, leaving ghosts
2. Pebble from $k/2$ to k
3. Remove pebble at $k/2$

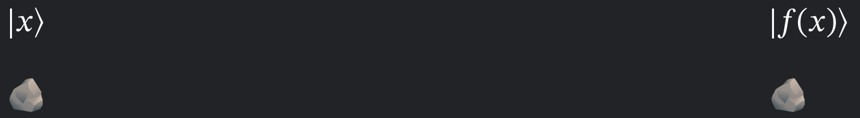


Space: $O(\log k)$ pebbles 😃

Time cost (# steps): $T(k) = O(k) + 2T(k/2)$

Spooky pebble games

1. Blast straight to $k/2$, leaving ghosts
2. Pebble from $k/2$ to k
3. Remove pebble at $k/2$



Space: $O(\log k)$ pebbles 😃

Time cost (# steps): $O(k \log k)$ steps 🤩

Our work: parallel spooky pebble games

Absolutely optimal depth for a length-k pebble game: $2k - 1$ steps

Our work: parallel spooky pebble games

Absolutely optimal depth for a length-k pebble game: $2k - 1$ steps

Without parallelism, this is **only** achieved by trivial $O(k)$-space strategy

Our work: parallel spooky pebble games

Absolutely optimal depth for a length-k pebble game: $2k - 1$ steps

Without parallelism, this is **only** achieved by trivial $O(k)$-space strategy

Can we achieve depth $2k - 1$ with less space, using parallelism?

An optimal parallel spooky pebble game for $k = 12$

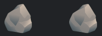
$|x\rangle \quad |z_1\rangle \quad |z_2\rangle \quad |z_3\rangle \quad |z_4\rangle \quad |z_5\rangle \quad |z_6\rangle \quad |z_7\rangle \quad |z_8\rangle \quad |z_9\rangle \quad |z_{10}\rangle \quad |z_{11}\rangle \quad |f(x)\rangle$

Step number: 0

Max. pebble count: 1

An optimal parallel spooky pebble game for $k = 12$

$|x\rangle \quad |z_1\rangle \quad |z_2\rangle \quad |z_3\rangle \quad |z_4\rangle \quad |z_5\rangle \quad |z_6\rangle \quad |z_7\rangle \quad |z_8\rangle \quad |z_9\rangle \quad |z_{10}\rangle \quad |z_{11}\rangle \quad |f(x)\rangle$



Step number: 1

Max. pebble count: 2

An optimal parallel spooky pebble game for $k = 12$

$|x\rangle$ $|z_1\rangle$ $|z_2\rangle$ $|z_3\rangle$ $|z_4\rangle$ $|z_5\rangle$ $|z_6\rangle$ $|z_7\rangle$ $|z_8\rangle$ $|z_9\rangle$ $|z_{10}\rangle$ $|z_{11}\rangle$ $|f(x)\rangle$

🪨 👻 🪨

Step number: 2

Max. pebble count: 2

An optimal parallel spooky pebble game for $k = 12$

$|x\rangle$ $|z_1\rangle$ $|z_2\rangle$ $|z_3\rangle$ $|z_4\rangle$ $|z_5\rangle$ $|z_6\rangle$ $|z_7\rangle$ $|z_8\rangle$ $|z_9\rangle$ $|z_{10}\rangle$ $|z_{11}\rangle$ $|f(x)\rangle$

🪨 👻 👻 🪨

Step number: 3

Max. pebble count: 2

An optimal parallel spooky pebble game for $k = 12$

$|x\rangle$ $|z_1\rangle$ $|z_2\rangle$ $|z_3\rangle$ $|z_4\rangle$ $|z_5\rangle$ $|z_6\rangle$ $|z_7\rangle$ $|z_8\rangle$ $|z_9\rangle$ $|z_{10}\rangle$ $|z_{11}\rangle$ $|f(x)\rangle$

Step number: 4

Max. pebble count: 2

An optimal parallel spooky pebble game for $k = 12$

$\|x\rangle$	$\|z_1\rangle$	$\|z_2\rangle$	$\|z_3\rangle$	$\|z_4\rangle$	$\|z_5\rangle$	$\|z_6\rangle$	$\|z_7\rangle$	$\|z_8\rangle$	$\|z_9\rangle$	$\|z_{10}\rangle$	$\|z_{11}\rangle$	$\|f(x)\rangle$
🪨	👻	👻	👻	🪨	🪨							

Step number: 5

Max. pebble count: 3

An optimal parallel spooky pebble game for $k = 12$

$\|x\rangle$	$\|z_1\rangle$	$\|z_2\rangle$	$\|z_3\rangle$	$\|z_4\rangle$	$\|z_5\rangle$	$\|z_6\rangle$	$\|z_7\rangle$	$\|z_8\rangle$	$\|z_9\rangle$	$\|z_{10}\rangle$	$\|z_{11}\rangle$	$\|f(x)\rangle$
🪨	👻	👻	👻	🪨	👻	🪨						

Step number: 6

Max. pebble count: 3

An optimal parallel spooky pebble game for $k = 12$

Step number: 7

Max. pebble count: 3

An optimal parallel spooky pebble game for $k = 12$

$|x\rangle \quad |z_1\rangle \quad |z_2\rangle \quad |z_3\rangle \quad |z_4\rangle \quad |z_5\rangle \quad |z_6\rangle \quad |z_7\rangle \quad |z_8\rangle \quad |z_9\rangle \quad |z_{10}\rangle \quad |z_{11}\rangle \quad |f(x)\rangle$

Step number: 8

Max. pebble count: 4

An optimal parallel spooky pebble game for $k = 12$

$|x\rangle$ $|z_1\rangle$ $|z_2\rangle$ $|z_3\rangle$ $|z_4\rangle$ $|z_5\rangle$ $|z_6\rangle$ $|z_7\rangle$ $|z_8\rangle$ $|z_9\rangle$ $|z_{10}\rangle$ $|z_{11}\rangle$ $|f(x)\rangle$

Step number: 9

Max. pebble count: 4

An optimal parallel spooky pebble game for $k = 12$

$\|x\rangle$	$\|z_1\rangle$	$\|z_2\rangle$	$\|z_3\rangle$	$\|z_4\rangle$	$\|z_5\rangle$	$\|z_6\rangle$	$\|z_7\rangle$	$\|z_8\rangle$	$\|z_9\rangle$	$\|z_{10}\rangle$	$\|z_{11}\rangle$	$\|f(x)\rangle$
🪨	👻	👻	👻	🪨	👻	👻	🪨	👻	🪨	🪨		

Step number: 10

Max. pebble count: 5

An optimal parallel spooky pebble game for $k = 12$

$\|x\rangle$	$\|z_1\rangle$	$\|z_2\rangle$	$\|z_3\rangle$	$\|z_4\rangle$	$\|z_5\rangle$	$\|z_6\rangle$	$\|z_7\rangle$	$\|z_8\rangle$	$\|z_9\rangle$	$\|z_{10}\rangle$	$\|z_{11}\rangle$	$\|f(x)\rangle$
🪨	👻	👻	👻	🪨	👻	👻	🪨	👻	🪨	🪨	🪨	

Step number: 11

Max. pebble count: 6

An optimal parallel spooky pebble game for $k = 12$

$$|x\rangle \quad |z_1\rangle \quad |z_2\rangle \quad |z_3\rangle \quad |z_4\rangle \quad |z_5\rangle \quad |z_6\rangle \quad |z_7\rangle \quad |z_8\rangle \quad |z_9\rangle \quad |z_{10}\rangle \quad |z_{11}\rangle \quad |f(x)\rangle$$

Step number: 12

Max. pebble count: 7

An optimal parallel spooky pebble game for $k = 12$

$$|x\rangle \quad |z_1\rangle \quad |z_2\rangle \quad |z_3\rangle \quad |z_4\rangle \quad |z_5\rangle \quad |z_6\rangle \quad |z_7\rangle \quad |z_8\rangle \quad |z_9\rangle \quad |z_{10}\rangle \quad |z_{11}\rangle \quad |f(x)\rangle$$

Step number: 13

Max. pebble count: 7

An optimal parallel spooky pebble game for $k = 12$

$\lvert x\rangle$	$\lvert z_1\rangle$	$\lvert z_2\rangle$	$\lvert z_3\rangle$	$\lvert z_4\rangle$	$\lvert z_5\rangle$	$\lvert z_6\rangle$	$\lvert z_7\rangle$	$\lvert z_8\rangle$	$\lvert z_9\rangle$	$\lvert z_{10}\rangle$	$\lvert z_{11}\rangle$	$\lvert f(x)\rangle$
🪨	👻	👻	👻	🪨	👻	👻	🪨	🪨	🪨			🪨

Step number: 14

Max. pebble count: 7

An optimal parallel spooky pebble game for $k = 12$

$\vert x\rangle$	$\vert z_1\rangle$	$\vert z_2\rangle$	$\vert z_3\rangle$	$\vert z_4\rangle$	$\vert z_5\rangle$	$\vert z_6\rangle$	$\vert z_7\rangle$	$\vert z_8\rangle$	$\vert z_9\rangle$	$\vert z_{10}\rangle$	$\vert z_{11}\rangle$	$\vert f(x)\rangle$
🪨	👻	👻	👻	🪨	🪨	👻	🪨	🪨				🪨

Step number: 15

Max. pebble count: 7

An optimal parallel spooky pebble game for $k = 12$

$|x\rangle$ $|z_1\rangle$ $|z_2\rangle$ $|z_3\rangle$ $|z_4\rangle$ $|z_5\rangle$ $|z_6\rangle$ $|z_7\rangle$ $|z_8\rangle$ $|z_9\rangle$ $|z_{10}\rangle$ $|z_{11}\rangle$ $|f(x)\rangle$

Step number: 16

Max. pebble count: 7

An optimal parallel spooky pebble game for $k = 12$

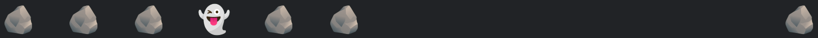
$|x\rangle \quad |z_1\rangle \quad |z_2\rangle \quad |z_3\rangle \quad |z_4\rangle \quad |z_5\rangle \quad |z_6\rangle \quad |z_7\rangle \quad |z_8\rangle \quad |z_9\rangle \quad |z_{10}\rangle \quad |z_{11}\rangle \quad |f(x)\rangle$

Step number: 17

Max. pebble count: 7

An optimal parallel spooky pebble game for $k = 12$

$|x\rangle$ $|z_1\rangle$ $|z_2\rangle$ $|z_3\rangle$ $|z_4\rangle$ $|z_5\rangle$ $|z_6\rangle$ $|z_7\rangle$ $|z_8\rangle$ $|z_9\rangle$ $|z_{10}\rangle$ $|z_{11}\rangle$ $|f(x)\rangle$

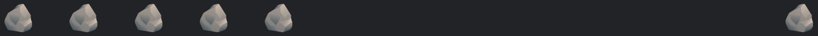


Step number: 18

Max. pebble count: 7

An optimal parallel spooky pebble game for $k = 12$

$|x\rangle$ $|z_1\rangle$ $|z_2\rangle$ $|z_3\rangle$ $|z_4\rangle$ $|z_5\rangle$ $|z_6\rangle$ $|z_7\rangle$ $|z_8\rangle$ $|z_9\rangle$ $|z_{10}\rangle$ $|z_{11}\rangle$ $|f(x)\rangle$

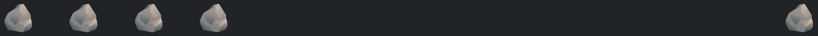


Step number: 19

Max. pebble count: 7

An optimal parallel spooky pebble game for $k = 12$

$|x\rangle$ $|z_1\rangle$ $|z_2\rangle$ $|z_3\rangle$ $|z_4\rangle$ $|z_5\rangle$ $|z_6\rangle$ $|z_7\rangle$ $|z_8\rangle$ $|z_9\rangle$ $|z_{10}\rangle$ $|z_{11}\rangle$ $|f(x)\rangle$

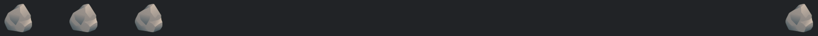


Step number: 20

Max. pebble count: 7

An optimal parallel spooky pebble game for $k = 12$

$|x\rangle$ $|z_1\rangle$ $|z_2\rangle$ $|z_3\rangle$ $|z_4\rangle$ $|z_5\rangle$ $|z_6\rangle$ $|z_7\rangle$ $|z_8\rangle$ $|z_9\rangle$ $|z_{10}\rangle$ $|z_{11}\rangle$ $|f(x)\rangle$

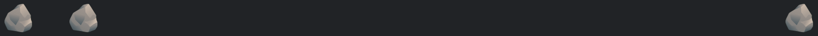


Step number: 21

Max. pebble count: 7

An optimal parallel spooky pebble game for $k = 12$

$|x\rangle$ $|z_1\rangle$ $|z_2\rangle$ $|z_3\rangle$ $|z_4\rangle$ $|z_5\rangle$ $|z_6\rangle$ $|z_7\rangle$ $|z_8\rangle$ $|z_9\rangle$ $|z_{10}\rangle$ $|z_{11}\rangle$ $|f(x)\rangle$



Step number: 22

Max. pebble count: 7

$|x\rangle \quad |z_1\rangle \quad |z_2\rangle \quad |z_3\rangle \quad |z_4\rangle \quad |z_5\rangle \quad |z_6\rangle \quad |z_7\rangle \quad |z_8\rangle \quad |z_9\rangle \quad |z_{10}\rangle \quad |z_{11}\rangle \quad |f(x)\rangle$

Step number: 23

Max. pebble count: 7

An optimal parallel spooky pebble game for $k = 12$

$|x\rangle$ $|z_1\rangle$ $|z_2\rangle$ $|z_3\rangle$ $|z_4\rangle$ $|z_5\rangle$ $|z_6\rangle$ $|z_7\rangle$ $|z_8\rangle$ $|z_9\rangle$ $|z_{10}\rangle$ $|z_{11}\rangle$ $|f(x)\rangle$

🪨 🪨

Step number: $23 = 2k - 1$, which is optimal 🤩

Max. pebble count: 7

Our results

Explicit construction

Our results

Explicit construction

- Achieves optimal depth $2k-1$

Our results

Explicit construction

- Achieves optimal depth $2k - 1$
- Our parallel construction uses only $2.47 \log k$ pebbles

Explicit construction

- Achieves optimal depth $2k - 1$
- Our parallel construction uses only $2.47 \log k$ pebbles
  - **Recall**: achieving optimal depth without parallelism required $O(k)$ pebbles

Our results

Explicit construction

- Achieves optimal depth $2k - 1$
- Our parallel construction uses only $2.47 \log k$ pebbles
  - **Recall**: achieving optimal depth without parallelism required $O(k)$ pebbles

Automated search

Our results

Explicit construction

- Achieves optimal depth $2k - 1$
- Our parallel construction uses only $2.47 \log k$ pebbles
  - **Recall**: achieving optimal depth without parallelism required $O(k)$ pebbles

Automated search

- Highly optimized **A* search** written in Julia

Our results

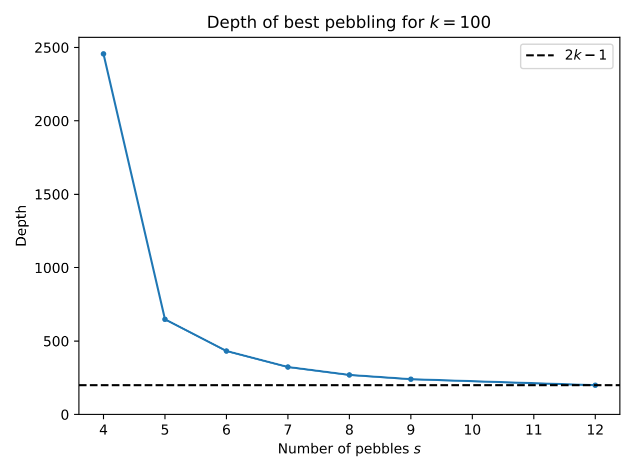
Explicit construction

- Achieves optimal depth $2k - 1$
- Our parallel construction uses only $2.47 \log k$ pebbles
  - **Recall**: achieving optimal depth without parallelism required $O(k)$ pebbles

Automated search

- Highly optimized **A* search** written in Julia
- Finds lowest-depth solution for *any* fixed number of pebbles s and length k

Numerical results



Factoring results

For factoring 4096-bit RSA:

- all depths counted in n-bit multiplications
- for previous estimates see: Ekerå + Gärtner, arXiv:2405.14381

Factoring results

For factoring 4096-bit RSA:

- all depths counted in n-bit multiplications
- for previous estimates see: Ekerå + Gärtner, arXiv:2405.14381

Circuit	Depth (mults.)	Qubits
Previous Regev + space saving (Ragavan et al. ’24)	700	$\sim 13n$

Factoring results

For factoring 4096-bit RSA:

- all depths counted in n-bit multiplications
- for previous estimates see: Ekerå + Gärtner, arXiv:2405.14381

Circuit	Depth (mults.)	Qubits
Previous Regev + space saving (Ragavan et al. ’24)	700	$\sim 13n$
Standard Shor (e.g. Gidney + Ekerå ’19)	444	$2n - 3n$

Factoring results

For factoring 4096-bit RSA:

- all depths counted in n-bit multiplications
- for previous estimates see: Ekerå + Gärtner, arXiv:2405.14381

Circuit	Depth (mults.)	Qubits
Previous Regev + space saving (Ragavan et al. '24)	700	$\sim 13n$
Standard Shor (e.g. Gidney + Ekerå '19)	444	$2n - 3n$
Our results (more pebbles)	380	$14n$

Factoring results

For factoring 4096-bit RSA:

- all depths counted in n-bit multiplications
- for previous estimates see: Ekerå + Gärtner, arXiv:2405.14381

Circuit	Depth (mults.)	Qubits
Previous Regev + space saving (Ragavan et al. ’24)	700	$\sim 13n$
Standard Shor (e.g. Gidney + Ekerå ’19)	444	$2n - 3n$
Our results (more pebbles)	380	$14n$
Our results (fewer pebbles)	450	$7n$

Factoring results

For factoring 4096-bit RSA:

- all depths counted in n-bit multiplications
- for previous estimates see: Ekerå + Gärtner, arXiv:2405.14381

Circuit	Depth (mults.)	Qubits
Previous Regev + space saving (Ragavan et al. '24)	700	$\sim 13n$
Standard Shor (e.g. Gidney + Ekerå '19)	444	$2n - 3n$
Our results (more pebbles)	380	$14n$
Our results (fewer pebbles)	450	$7n$

Note: this is shamelessly focusing on **depth** our best metric...

Takeaways

- Is this how we’ll factor the first cryptographic-size integers?

Takeaways

- Is this how we'll factor the first cryptographic-size integers?
  - **No.**

Takeaways

- Is this how we'll factor the first cryptographic-size integers?
  - **No.**
  - Qubit count, not depth, seems most important in *near-term*

Takeaways

- Is this how we'll factor the first cryptographic-size integers?
  - **No.**
  - Qubit count, not depth, seems most important in *near-term*
- "This paper creates **slack**"

Takeaways

- Is this how we'll factor the first cryptographic-size integers?
  - **No.**
  - Qubit count, not depth, seems most important in *near-term*
- "This paper creates **slack**"
- Efficiently evaluating sequential algorithms is **widely applicable**

Takeaways

- Is this how we'll factor the first cryptographic-size integers?
  - **No.**
  - Qubit count, not depth, seems most important in *near-term*
- "This paper creates **slack**"
- Efficiently evaluating sequential algorithms is **widely applicable**
- **Generalization:** parallel spooky pebbling on arbitrary graphs

INTELLIGENCE COMMUNITY
POSTDOCTORAL RESEARCH
FELLOWSHIP PROGRAM

VE RI TAS

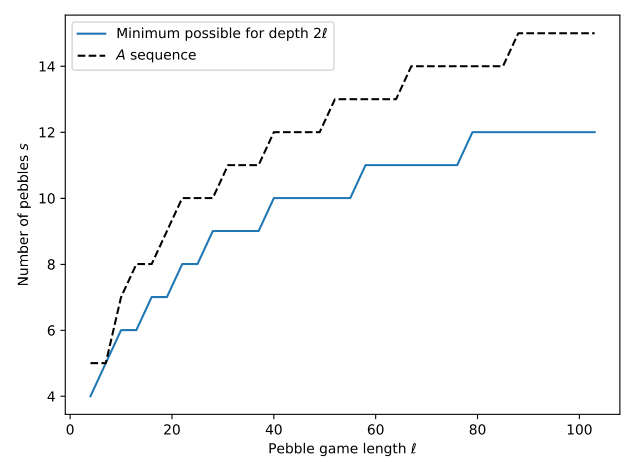
Hertz Foundation

Thank you!

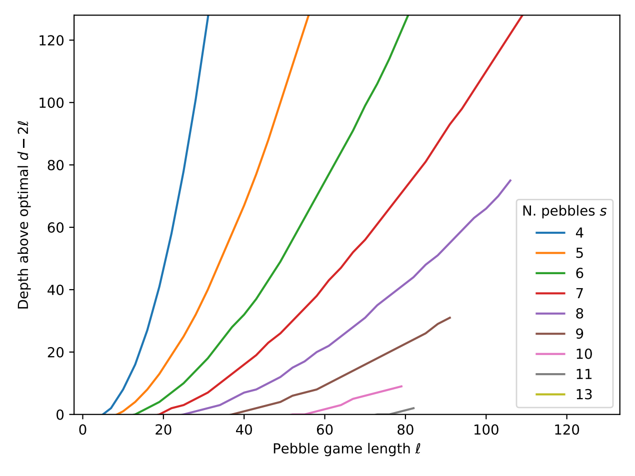
arXiv:2510.08432

Backup

Numerical results



Numerical results



Making Shor reversible

Break up x into its individual bits x_i:

$a^x \bmod N$

Making Shor reversible

Break up x into its individual bits x_i:

$$a^x \bmod N = a^{\sum_i 2^i x_i} \bmod N$$

Making Shor reversible

Break up x into its individual bits x_i:

$$a^x \bmod N = a^{\sum_i 2^i x_i} \bmod N$$
$$= \prod_i a^{2^i x_i} \bmod N$$

Making Shor reversible

Break up x into its individual bits x_i:

$$\begin{aligned} a^x \bmod N &= a^{\sum_i 2^i x_i} \bmod N \\ &= \prod_i a^{2^i x_i} \bmod N \\ &= \prod_i \left(a^{2^i}\right)^{x_i} \bmod N \end{aligned}$$

Making Shor reversible

Break up x into its individual bits x_i:

$$
\begin{aligned}
a^x \bmod N &= a^{\sum_i 2^i x_i} \bmod N \\
&= \prod_i a^{2^i x_i} \bmod N \\
&= \prod_i \left(a^{2^i}\right)^{x_i} \bmod N \\
&= \prod_i c_i^{x_i} \bmod N
\end{aligned}
$$

where classical $c_i = a^{2^i} \bmod N$

Making Shor reversible

Break up x into its individual bits x_i:

$$a^x \text{ mod } N = a^{\sum_i 2^i x_i} \text{ mod } N$$
$$= \prod_i a^{2^i x_i} \text{ mod } N$$
$$= \prod_i \left(a^{2^i}\right)^{x_i} \text{ mod } N$$
$$= \prod_i c_i^{x_i} \text{ mod } N$$

$|1\rangle$

where classical $c_i = a^{2^i} \text{ mod } N$

Making Shor reversible

Break up x into its individual bits x_i:

$$
\begin{aligned}
a^x \bmod N &= a^{\sum_i 2^i x_i} \bmod N \\
&= \prod_i a^{2^i x_i} \bmod N \\
&= \prod_i \left(a^{2^i}\right)^{x_i} \bmod N \\
&= \prod_i c_i^{x_i} \bmod N
\end{aligned}
$$

$$|1\rangle \rightarrow |c_0^{x_0}\rangle$$

where classical $c_i = a^{2^i} \bmod N$

Making Shor reversible

Break up x into its individual bits x_i:

$$\begin{aligned} a^x \bmod N &= a^{\sum_i 2^i x_i} \bmod N \\ &= \prod_i a^{2^i x_i} \bmod N \\ &= \prod_i \left(a^{2^i}\right)^{x_i} \bmod N \\ &= \prod_i c_i^{x_i} \bmod N \end{aligned}$$

$$|1\rangle \rightarrow |c_0^{x_0}\rangle \rightarrow |c_1^{x_1} c_0^{x_0}\rangle$$

where classical $c_i = a^{2^i} \bmod N$

Making Shor reversible

Break up x into its individual bits x_i:

$$
\begin{aligned}
a^x \bmod N &= a^{\sum_i 2^i x_i} \bmod N \\
&= \prod_i a^{2^i x_i} \bmod N \\
&= \prod_i \left(a^{2^i}\right)^{x_i} \bmod N \\
&= \prod_i c_i^{x_i} \bmod N
\end{aligned}
$$

where classical $c_i = a^{2^i} \bmod N$

$$|1\rangle \rightarrow |c_0^{x_0}\rangle \rightarrow |c_1^{x_1} c_0^{x_0}\rangle \rightarrow |c_2^{x_2} c_1^{x_1} c_0^{x_0}\rangle$$

Making Shor reversible

Break up x into its individual bits x_i:

$$\begin{aligned} a^x \bmod N &= a^{\sum_i 2^i x_i} \bmod N \\ &= \prod_i a^{2^i x_i} \bmod N \\ &= \prod_i \left(a^{2^i}\right)^{x_i} \bmod N \\ &= \prod_i c_i^{x_i} \bmod N \end{aligned}$$

where classical $c_i = a^{2^i} \bmod N$

$$|1\rangle \to |c_0^{x_0}\rangle \to |c_1^{x_1} c_0^{x_0}\rangle \to |c_2^{x_2} c_1^{x_1} c_0^{x_0}\rangle \to \cdots$$

Making Shor reversible

Break up x into its individual bits x_i:

$$
\begin{aligned}
a^x \bmod N &= a^{\sum_i 2^i x_i} \bmod N \\
&= \prod_i a^{2^i x_i} \bmod N \\
&= \prod_i \left(a^{2^i}\right)^{x_i} \bmod N \\
&= \prod_i c_i^{x_i} \bmod N
\end{aligned}
$$

where classical $c_i = a^{2^i} \bmod N$

$$|1\rangle \to |c_0^{x_0}\rangle \to |c_1^{x_1} c_0^{x_0}\rangle \to |c_2^{x_2} c_1^{x_1} c_0^{x_0}\rangle \to \cdots$$

Each iteration is a controlled multiplication by classical c_i—which is reversible!